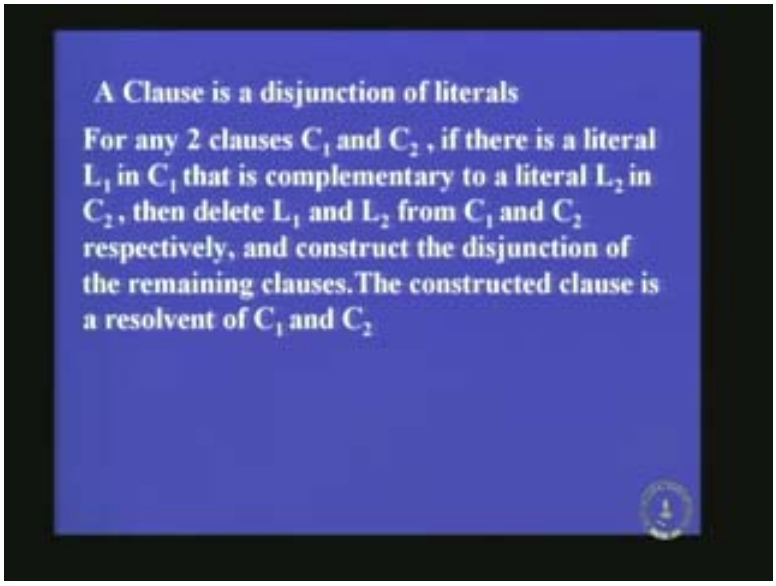
Discrete Mathematical Structures
Dr. Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture - 6
Resolution Principles and Application to Prolog

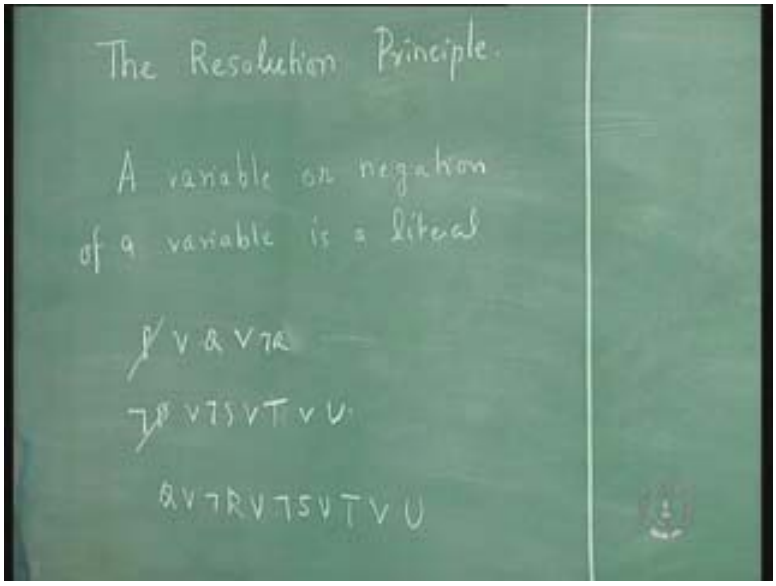
We have seen the rules of inferences in deduction and how we can use them to find out whether an argument is valid or not. We have also seen what is meant by a conjunctive normal form and a disjunctive normal form. And we have also seen how to bring a well formed formula of propositional logic to conjunctive normal form or disjunctive normal form. Now we shall see how to use the resolution principle to find out whether an argument is correct or not so we shall concentrate on the resolution principle. A variable or negation of a variable is called a literal. And you know that a disjunction of literals is called a sum and the conjunction of literals is called a product. A clause is also a disjunction of literals, it is a sum.

(Refer Slide Time: 02:26)



Now what do you mean by resolution? From any two clauses C_1 and C_2 if there is a literal L_1 in C_1 that is complimentary to a literal L_2 in C_2 then delete L_1 and L_2 from C_1 and C_2 respectively and construct the disjunction of the remaining clauses. The constructed clause is the resolvent of C_1 and C_2. This is the definition of a resolvent as to what is meant by resolvent. It is like this: Suppose you have two clauses may be I shall fill this with some Q NOT R something like that NOT S OR P OR T OR U.

(Refer Slide Time: 04:18)

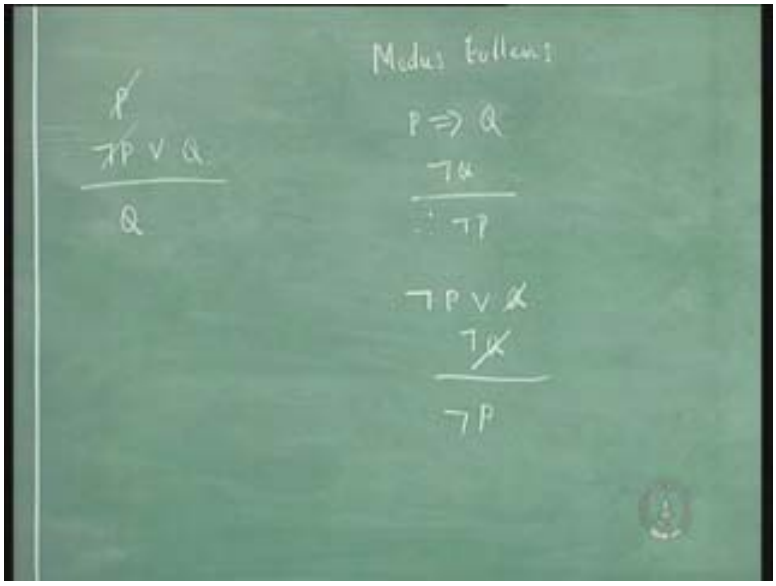


This is a clause, it is a disjunction of literals and is called a clause. This is also a clause, this is a literal this is a literal this is a literal and this is a literal. Now in this clause you have a variable and in this clause you have the negation of the variable. So this literal and this literal are complementary to each other. Then the resolvent of these two clauses is obtained by deleting these two and getting the disjunction of these two. That is Q OR NOT R OR NOT S OR T OR U combine these two this is the resolvent of this and this.

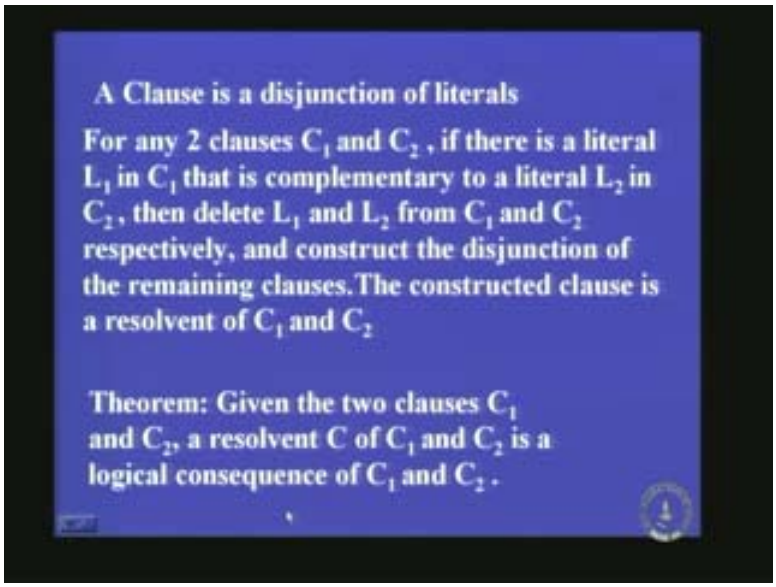
Now for example, Modus ponens you know that P and P implies Q is Q. If I express this in logical form P OR Q I can write as NOT P OR Q so what is the resolvent of these two? You have to remove this and you end up with Q. And similarly Modus tollens you have P implies Q and NOT Q from which you conclude NOT P.

Look at it in clause form; you can write P implies Q as NOT P OR Q and then you have NOT Q take the resolvent of this you will get NOT P. This is just to show how the resolvent is connected to the conclusion.

(Refer Slide Time: 05:15)



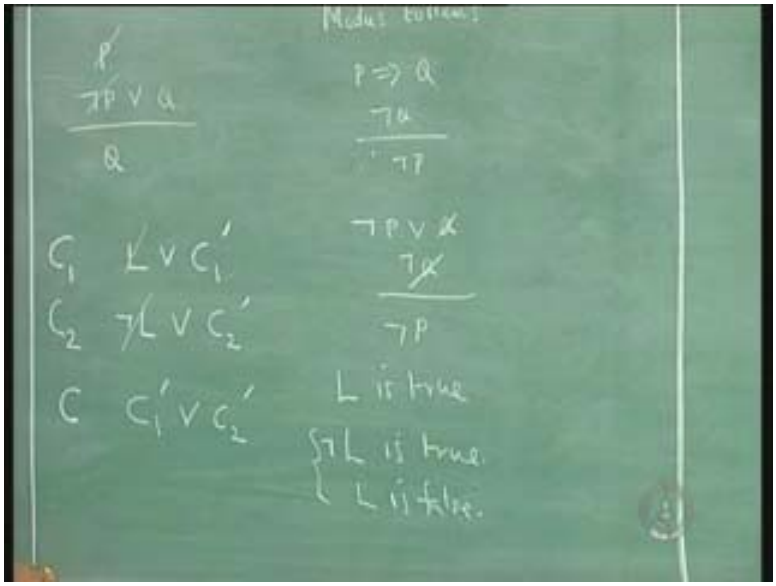
(Refer Slide Time: 05:38)



We will see the next one. Given two clauses C_1 and C_2 a resolvent C_1 and C_2 is the logical consequence of C_1 and C_2. In these two examples we have seen that there are two clauses given, the resolvent of them is the logical consequence of the two that you can see. Here again you can see that the resolvent of these two is the logical consequence. It is a theorem and in general how do you prove this theorem? So I have a clause C_1 in which I have a literal L and remaining clauses I can put as C1 prime and I have another clause C_2 which contains NOT of L and then the remaining clauses I can put as C_2 dash.

Now the resolvent C of C_1 and C_2 is, you remove these two and find the disjunction of the remaining portions, this is the resolvent of this. Now we have to show that this is the logical consequence of this. Now you must remember that this is a sum and this is also sum only having Ors in full. And what do you mean by the conclusion or the consequence? If the premises are true the conclusion must be true this is what we want to show. Now the premises are true then there are two possibilities either L is true or NOT L is true L is false that is this is equivalent to saying L is false. Either L is false or L is true.

(Refer Slide Time: 7:37)

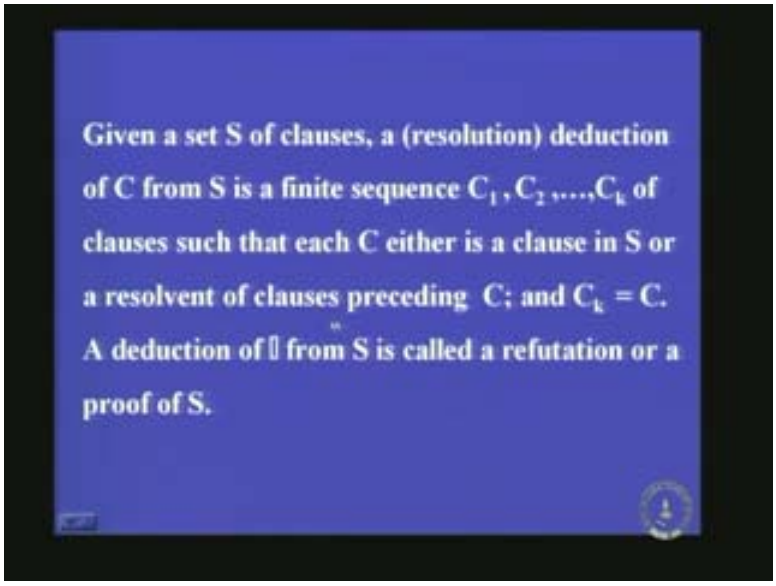


Suppose L is false in that case look at the C_1 if this has to be true then C_1 dash has to be true. That means C_1 prime or C_2 prime will be true. If L is true then NOT L will be false. So in the second clause this is false so in order that the second clause should be true this must be true. And if this is true the whole thing is true because everything is only disjunction.

So you see that when you remove this L and NOT L from two clauses and construct a disjunction of C_1 prime and C_2 prime either C_1 prime will be true or C2 prime will be true depending upon whether L is false or L is true. And L has to be either false or it has to be true, one of them should happen. So ultimately the clause C_1 prime OR C_2 prime will be true. So, whenever these two clauses are true C will be true. That is why C is the resolvent of C_1 and C_2. And this is the major result using which we find out whether an argument is valid or not.

So what is the resolution principle? This is the resolution principle; given a set S of clauses a resolution or deduction of C from S is a finite sequence C_1 C_2 C_k of clauses such that each C is either a clause in S or resolvent of clauses preceding C and C_k is equal to C.

(Refer Slide Time: 09:55)



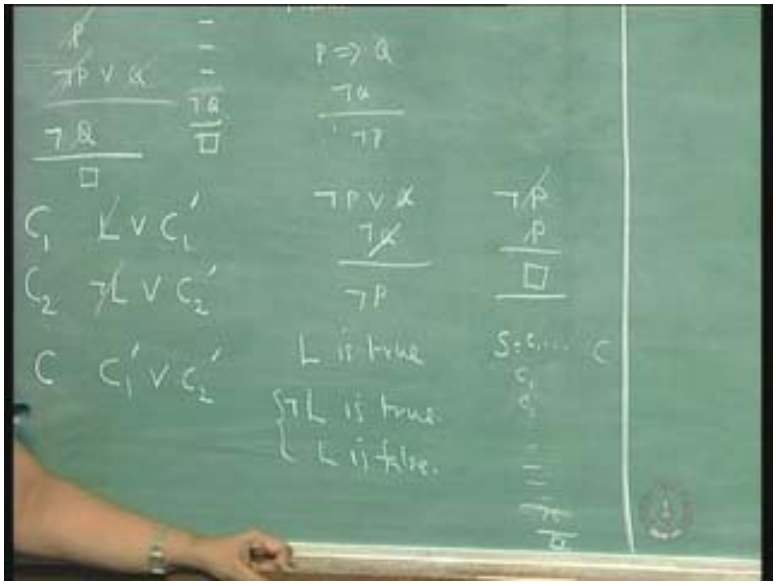
A deduction of the empty clause from S is called a refutation or a proof of S. So look at this example which you have already considered. This is a clause I mean it contains only a single literal, this is also a clause and the resolvent of these two is Q which is a logical consequence. So here from some clauses you are getting more and more clauses and finally you derive the conclusion.

Now if you want to look at it as a refutation or a proof instead of considering like this in the final conclusion you take the negation of the conclusion and then you derive the empty clause from that.

So here instead of taking these two you also take the negation of the conclusion and you can see that if you find the resolvent the resolvent of this will be Q and if you resolve these two clauses you will get the empty clause. This is called the proof or the refutation. And in this case it will be like this; the resolvent of these two is NOT P and you have to take the negation of the conclusion which is P and so you get the empty clause from that this is called a refutation or a proof.

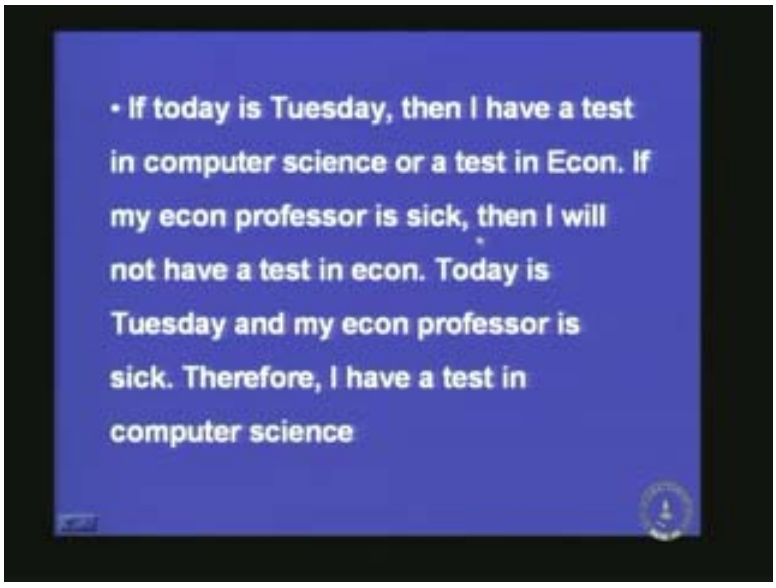
I will once again repeat this; that is you have a collection of some set of clauses C_1 C_2 etc and you want to derive a conclusion from that, what you have to do is try to resolve some of them and get more and more clauses, make use of the resolution principle and finally whatever you want to conclude you take the negation of that and then arrive at the empty clause.

(Refer Slide Time: 12:03)



If this works out correctly then the argument is correct and if it does not workout correctly then the argument is not correct. We have considered some examples to find whether an argument is valid or not. Let us take the same example and work them out using resolution principle. This principle is called the resolution principle. So let us take the second and third example which we already considered. Let us take third one first. If today is Tuesday then I have a test in computer science or a test in Economics. If my Economics professor is sick then I will not have a test in Economics.

(Refer Slide Time: 12:53)



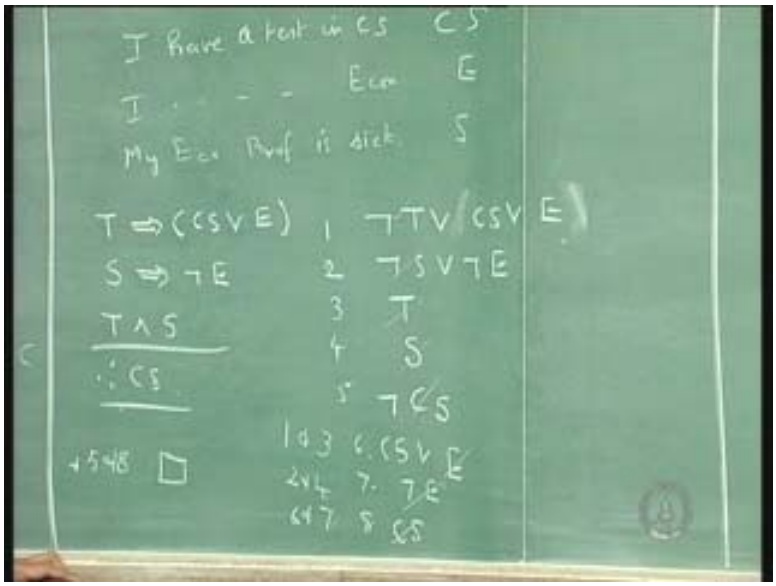
Today is Tuesday and my Economics professor is sick therefore I have test in Computer Science. We know that this argument is correct. Let us see how to prove it using resolution principle. So today is Tuesday T, I have a test in Computer Science CS, I have a test in Economics E, my Economics professor is sick. So how do you write the arguments? If today is Tuesday then I have a test in Computer Science or Economics. If my Economics professor is sick then I will not have a test in Economics. Today is Tuesday and my Economics professor is sick therefore I have a test in Computer Science. And we have seen that using modus ponens etc and we can prove this.

Now let us try to write them in clause form. So the first one you will write as NOT T OR CS OR E. P implies Q can be written as NOT P OR Q. And because OR is associative you can even write it without the parenthesis, this is one clause. Similarly, write this in clause form it will be NOT S OR NOT E. And the last one is a conjunction T AND S.

Actually you have to take it as two clauses one clause is just T having only one literal another clause is just having one literal S. So you have to split it like T and S. And the conclusion is CS to derive the empty clause you have to take the negation of the conclusion so you take NOT S and try to use the resolution principle.

Now I will write it as 1 2 3 4 5 from 1 and 3 this T will cancel with this and you will get CS OR E. And from 2 and 4 this will cancel with this and you will get NOT E and from 6 and 7 you will cancel this and you will get CS and from 5 and 8 CS and CS will get and you derive the empty clause. So this argument is a correct argument because we are able to derive the empty clause from that, this is called the proof or a refutation.

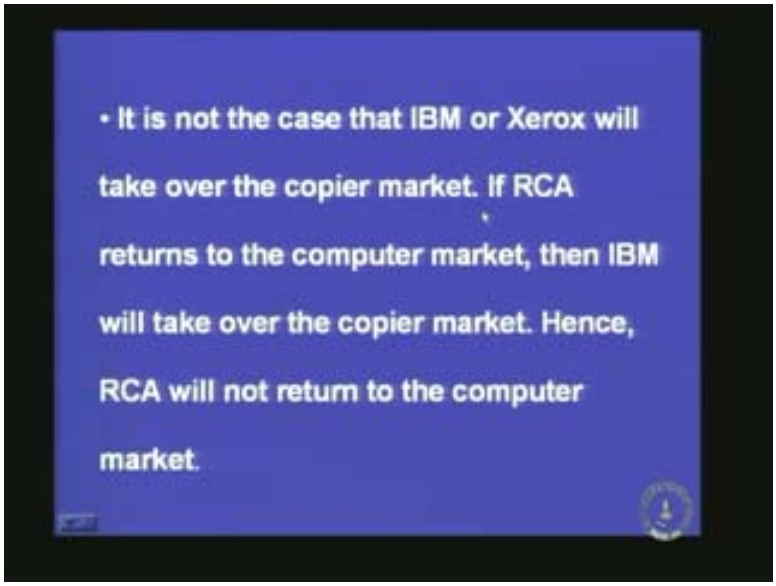
(Refer Slide Time: 16:11)



You can see that instead of the usual procedure of deduction if we use this, this is much simpler and this can be automated. And this is the one which is used in the logic programming language PROLOG. Now let me take the second example and we shall see

how to use the resolution principle for that. It is not the case that IBM or Xerox will take over the copier market.

(Refer Slide Time: 16:48)



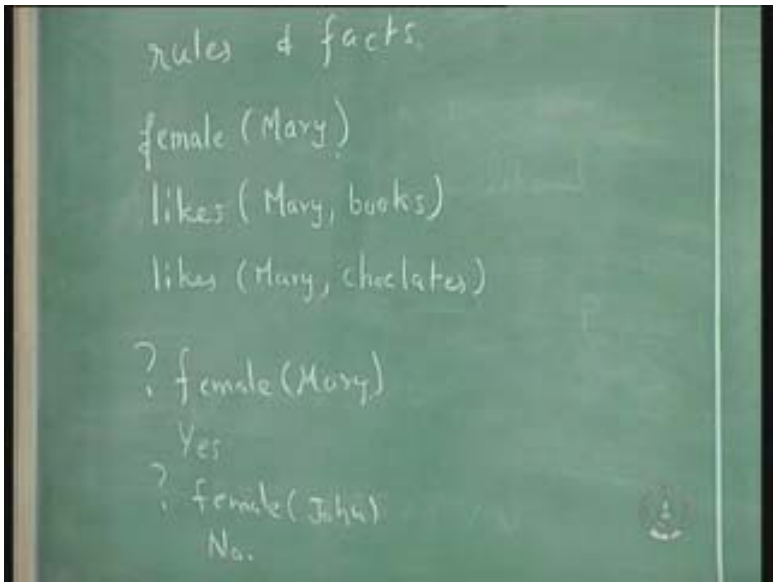
If RCA returns to the computer market then IBM will take over the copier market. Hence RCA will not return to the copier market. So let us write it down. IBM takes over copier market I, Xerox takes over the copier market X, RCA returns to computer market R. So it can be written like this; it is not the case that IBM or Xerox will take over the copier market.

If RCA returns to the computer market then IBM will take over the copier market. Therefore RCA will not return to the computer market, this is the argument. We have seen that this argument is correct. Try to write it in clause form, the first one using De Morgan primes laws can be written as NOT I and NOT X this is not a clause because it is a conjunction. So, as a clause you have to write it as NOT I NOT X, separate them and write it. This you can write as NOT R OR I NOT R OR I and you have to take the negation of the conclusion so that is R and from this you must be able to derive the empty clause. So consider this as 1 2 3 4 so from 3 and 4 you get I, cancel this and you get I, call it as 5 then from 1 and 5 you get the empty clause, this will cancel with this. You have not made use of the second one that is fine. It is not necessary you should make use of every clause in the argument. Even without making use of something you are able to derive the empty clause so that is fine and the argument is correct. This is how the resolution principle works. This is very much used in the logic programming language PROLOG. It is used in artificial intelligence applications. So let us see what the logic programming language PROLOG is and how the resolution principle is used in that.

The logic programming language PROLOG has some rules and facts or I should rather say facts and rules. What are facts? Facts are something written like this;

Like female (Mary) likes (Mary, books) likes (Mary chocolates) like that. This will stand for the fact Mary is a female and this will stand for the fact Mary likes books and this will stand for the fact Mary likes chocolates and so on. So actually you give a lot of facts to the system and then you also have some rules. Making use of that if you ask some questions it will reply.

(Refer Slide Time: 21:35)

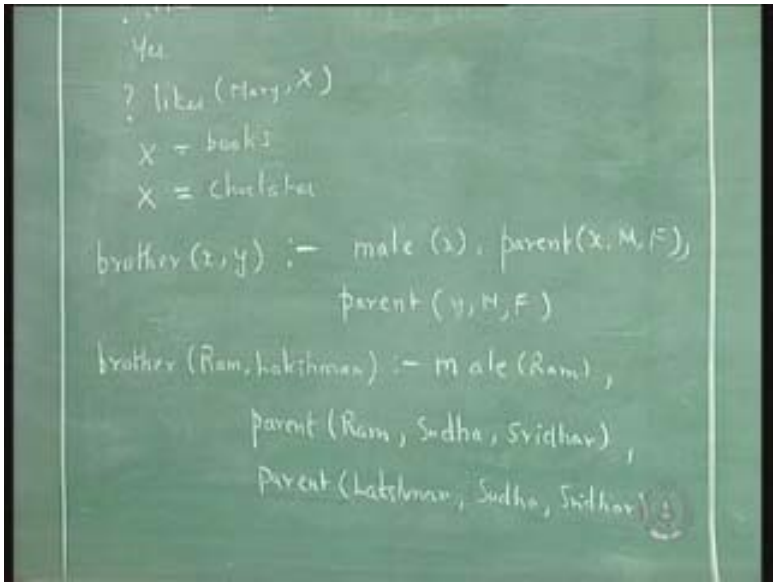


For example, in this case if you ask a question is female (Mary), the question is, is Mary a female? Then it will come out with the answer yes. Or, if you ask something like is female (John) or something like that it will say no. You can also ask questions like this, likes (Mary, books) which you mean does Mary likes books? It will say yes. Or if you ask questions like likes (Mary, x) what does Mary like? Then it will say x is equal to books. It depends on which one you have given as the first one. So if you are given this before this fact then it will say this. Then if you press a semicolon or something or sometimes depending upon the implementation of the language, next if you ask again it will say x is equal to chocolates. And then afterwards if you press it will say no, nothing, this is that. Like that you can ask some questions, given some fact you can ask questions and get the answer.

Now, there are some rules also something like you know like brother (x, y), if male x, parent x M F, parent Y M F, this should be interpreted as, when do you say that the x is the brother of y? When x is a male and also x and y have the same mother and the same father. Parents of x are M and F M is the mother F is the father. And similarly y also has the same mother and the same father. Then you say that x is the brother of y, this is called a rule. This is a general rule and you can instantiate it. When you instantiate it, it will be like this; brother Ram, Lakshman you must give the same value for x here so male Ram, this comma usually represents AND in the logic, parent Ram. Now here there is a Ram, mother is Sudha, Sridhar, parent Lakshman and so on. They have the same parents Ram

and Lakshman and Ram is a male. So, this when you instantiate you get something like this.

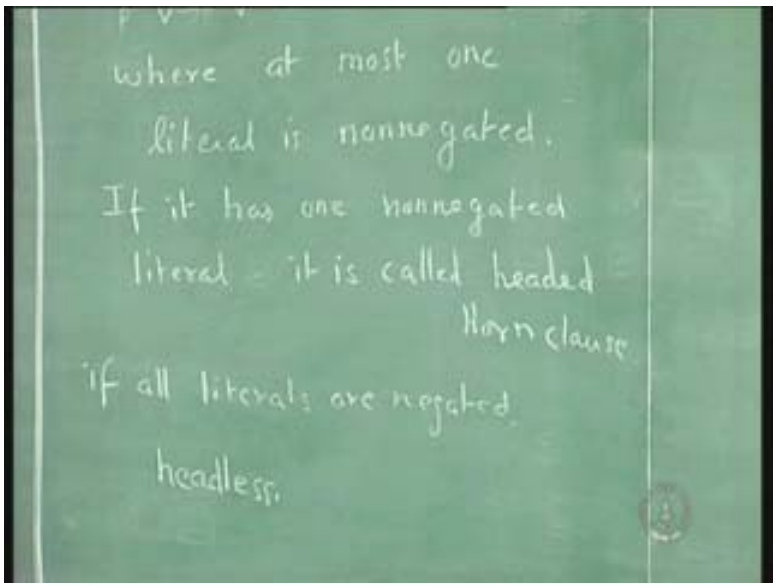
(Refer Slide Time: 25:42)



Now a horn clause is like this, a horn clause is again a disjunction of literals but all of them will be negated except one or either all of them will be negated or utmost one may be non-negated literal. So it is a disjunction of literals where utmost one literal is non-negated. All of them can be negated also. If a horn clause has one non-negated literal it is called headed horn clause. If all literals are negated it is said to be headless, headless horn clause.

Now, in the PROLOG sense this is looked at as a clause and this portion is the head and if you derive something from some facts and rules if you have something like this, this is hard headless clause. Now the if then format can be read like this. Look at this it can be read like this; if x is a male and parent of x are M and F and parent of y are M and F then x is the brother of y.

(Refer Slide Time: 27:34)



So this is the if portion and this is the then portion. You have to read it like that. A single fact like likes (Mary, books) something like that is just one literal and it is a head headed horn clause with just one literal. So a fact is a headed horn clause with just one literal. Let me take one example and show how this works. I will write both in the if then manner and also in the PROLOG manner and see how they correspond with each other.

(Refer Slide Time: 41:26)

So first we will write like this; Aunt x, y; x is the aunt of y, it represents x is the aunt of y. In that case female x female x, sister x, z x is the sister of z and parent z, y. So, if x is a female and x is the sister of z and z is a parent of y then x is the aunt of y. If you want to

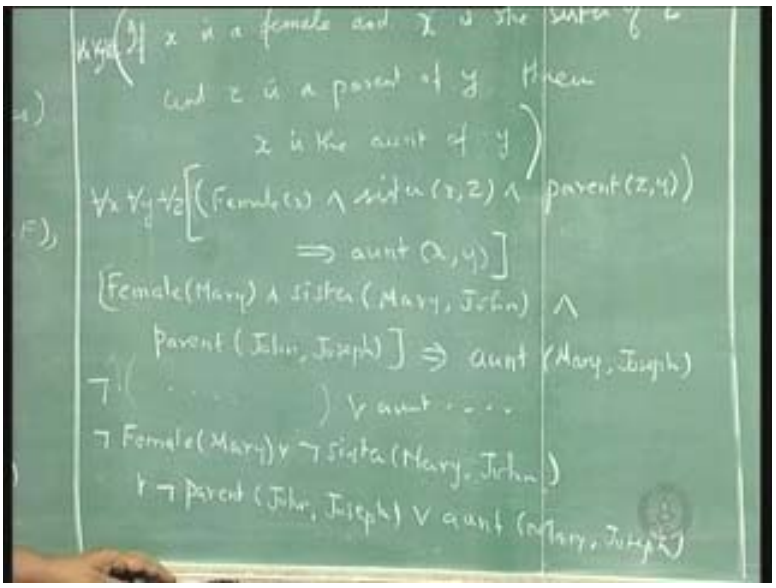
write this in the if then else notation I will write like this here; if x is a female and x is the sister of z and z is the parent of y then x is the aunt of y. Now, when you say it as a rule in general when you write it would mean for all of x for all of y for all of z how would you write it as a logical notation? For all of x for all of y for all of z female x AND sister x, z AND parent z, y implies aunt x, y.

Then suppose you have these facts, you have the facts female Mary, Sister Mary, John, parent John, Joseph. Then from this you can conclude Aunt Mary, Joseph. Now here, in the PROLOG how it will do is you have these facts you have this rule and this is your question, suppose you are asking is Mary the Aunt of Joseph? Then it will say yes. And in order to derive that it will take this and it has to satisfy this one. And this is called the goal and the PROLOG interpreter will try to find out that whether this goal is satisfied. And for that when it searches Aunt it finds a rule like this. So it will try to instantiate Aunt x, y that particular rule it will try to instantiate.

We have already seen what is meant by Universal instantiation, Universal generalization, Existential instantiation etc. So when you try to instantiate and take x to be Mary and y to be Joseph it can be something like this; Aunt Mary Joseph then female Mary, Sister Mary z parent z Joseph and it will try to find out whether there is a fact satisfying Sister Mary and so on and it will find this and it will also find this. So you have z taking z to be John this will be satisfied and when you instantiate you get this.

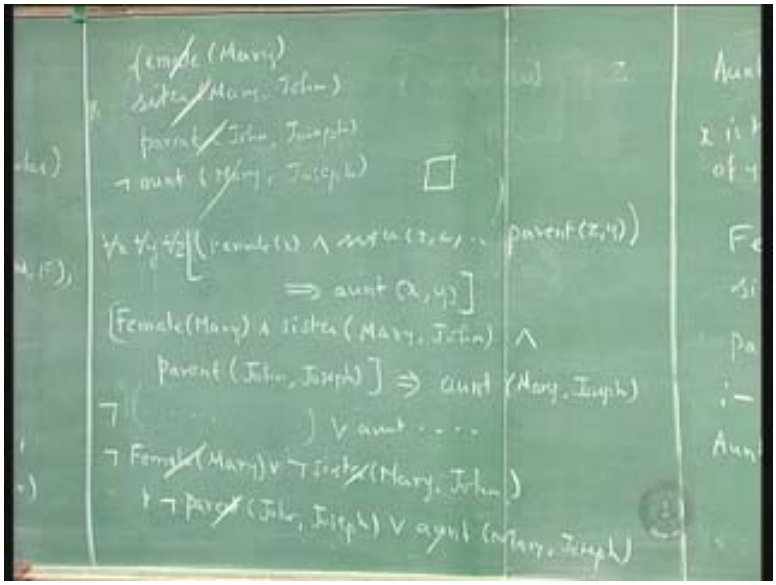
Now let us write it here, if you instantiate with x and y and z you will find that you can write something like that female Mary AND Sister Mary, John AND parent John Joseph implies Aunt Mary Joseph. Now write it in clause form, when you write it in clause form you will have NOT of the whole thing OR Aunt etc, and NOT of this whole thing you have two ANDs here using De Morgan primes laws this will become a clause like this; NOT female Mary OR NOT sister Mary, john OR NOT parent John, Joseph OR Aunt Mary, Joseph.

(Refer Slide Time: 37:31)



So I will try to rub this half because we have already seen that and we have these facts. So this is a clause it is a headed clause with this.

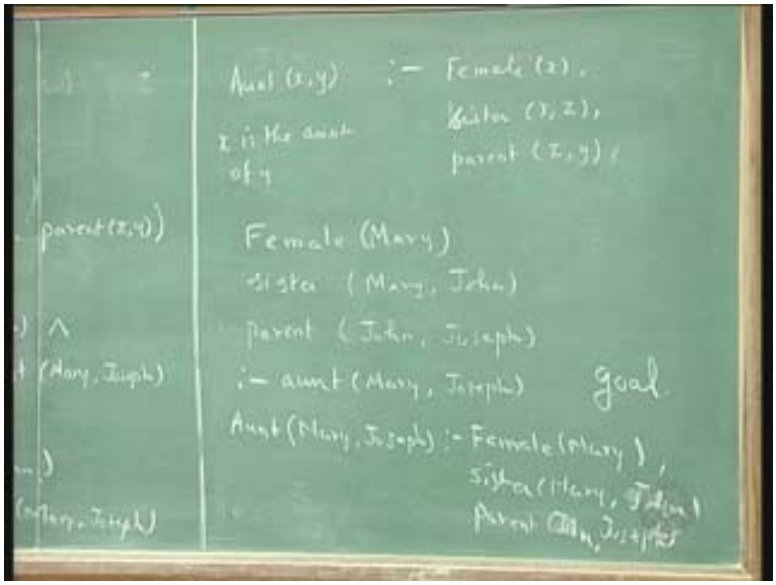
(Refer Slide Time: 39:50)



Now you have facts like female Mary, then you also have fact like sister Mary, John and parent John, Joseph. Now what you want to conclude is Aunt Mary, Joseph. This is what you want to conclude. So you have to take the negation of the conclusion that is NOT Aunt Mary, Joseph and from this you must derive the empty clause using resolution. We can very easily see that take this clause which has got four parts four literals this, this, this and this. And when you try to resolve using this try to use this first when you resolve

you will get the remaining three clauses. Then when you try to resolve this one with the second one this will get cancelled and you will have these two. Then when you try to resolve this with the third one this will get cancelled and you end up with this. But you have taken the negation of that so when you try to resolve this and this, this will get cancelled and you will end up with an empty clause. So you conclude that Mary is the Aunt of Joseph. Or you derive this conclusion Aunt Mary, Joseph. This is a general rule and when you instantiate you get this. This is looked as a headed horn clause, this is a head and this is a remaining portion. And each one of this is a fact and that is also called a headed horn clause there is no negation part it has only one literal and it is a headed horn clause. And the goal or the conclusion is this, you want to derive this from this and this and this is a headless horn clause.

(Refer Slide Time: 41:26)



So when you try to do this the solution, from this you see that you will be able to get this. So in order to have an argument or some sort of a goal satisfied the goal is written in this and that will be the only headless clause in that argument you need to have only one headless clause and the rest of them will be headed horn clause like this. It could be an instantiation of a rule like this where we have the head and the other portion. Each fact is a headed horn clause where there is no negated literal and the last conclusion which we derive from this is a headless horn clause.

Here also you look at this. See this, the first one you wrote it as a clause in the OR form. You see that this is negated this is negated this is negated but this is not negated it's a headed horn clause where this is the only non-negated one the rest of them are negated. Each one of them is a horn clause, this is headed this is headed this is headed but this is not headed and with this one headless horn clause you will be able to use resolution and derive the empty clause.

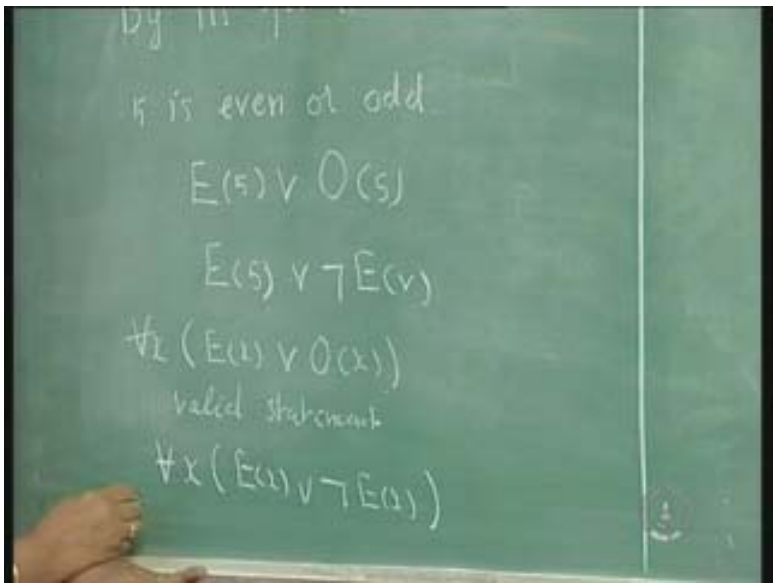
This is the resolution principle used in PROLOG and it is very automatic and it is a very good principle if you want to find out whether an argument is valid or not using Modus ponens, disjunctive syllogism, hypothetical syllogism etc and you have to use your own idea and do it. But this process is rather automated. The resolution principle is rather automated and you can program it. That is why this sort of an idea is used in PROLOG interpreters and you can understand the principle behind it.

One more thing is, actually in the PROLOG this is the goal, you want to find out whether Mary is the Aunt of Joseph so this is the goal to be satisfied. And you want to find out whether this goal will be satisfied for that you will try to find out whether there is a rule where you can instantiate or whether there are corresponding facts which will satisfy this one and so on. So, starting from the goal it will try to find out whether it is possible to derive um the entire thing starting from rules and facts and this process is called backward chain. That is, you start from the conclusion and see whether the conclusion can be derived from the facts and proofs. A forward chaining method you start with axioms and then try to use the rules of inference and derive at the conclusion that is the forward chaining method but here backward chaining method is used.

Next topic which we will consider is methods of proof. In schools and in earlier days you would have heard of theorems, proofs etc and you know what is meant by a proof. But now we shall see the same idea from the logical point of view. How a theorem looks like a logical statement and what sort of a logical idea you use in proving a theorem. There are several ways of proving it. All these things you already know but still we are going to look at these methods of proof from the logical point of view. So you know that starting from some axioms rules of inference you prove a result and that is called a theorem.

Now how theorems are proved? There are several ways in which we can prove theorems. One is by its form, the form of a theorem or the statement may look like a tautology and because it is a tautology it is proved. A tautology is always true so it will be proved. For example, something like this; 5 is even or odd I can say, this I can write in logical notation E(5) OR O(5) or in essence you can write it as E(5) OR NOT of E(5). This is P OR NOT P which is a tautology so it will be always true. This when you represent as a proposition in general you can have something like; for all of x where x is an integer E of x OR O of x every integer is even or odd. And this is a valid statement because you it is equivalent to saying for all of x E of x OR NOT E of x and this is a valid one, always true.

(Refer Slide Time: 46:27)

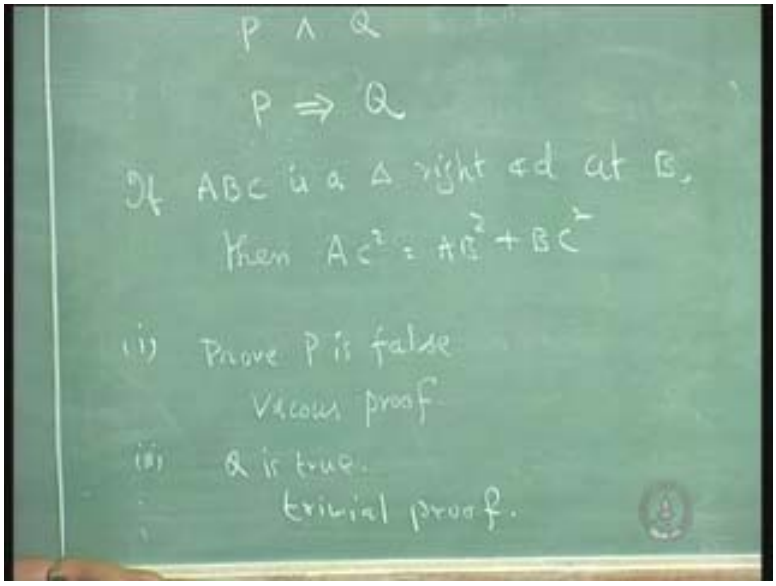


So because of the form or the way in which the statement is made because it is a tautology or a valid statement of the predicate calculus it holds, the result holds, the theorem holds. And this is one method of proof. So when the statement of the theorem is in the form of the tautology or a valid well formed formula of predicate logic it is a theorem. You need not have to prove something beyond that just by its form it is true and so it is a theorem.

We have heard of several types of theorems, proofs etc, and most of the theorems you may have statements like P AND Q and so on. If the statement is of the form P AND Q you have to prove the both P and Q. If it is of the P OR Q if you are able to prove one of them it will be true and so on. And in general most of the theorems the statement will be of the form P implies Q. If then, if you look at the statements, for example if ABC is a triangle right angled at B then AC square is equal to AB square plus BC square, this is Pythagoras theorem. And it is of the form if then.

Now if the statement of the theorem is of the form if then what sort of methods you will use to prove this? Now this is an implication P implies Q is an implication. We know that the implication is true if P is false or Q is true and it is false only when P is true and Q is false. Now here sometimes you will be able to prove very easily that P is false statement of the theorem P implies Q. Sometimes you may be able to prove that P is false or you may realize that P is false. In that case the implication will hold, P implies Q will automatically hold. Such a proof is called a Vacous proof. You need not have to prove anything else you just show P is false the implication will hold.

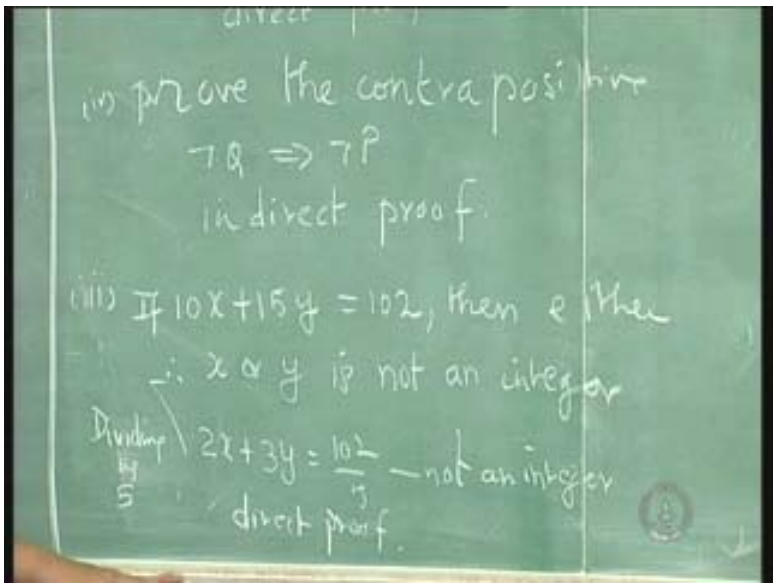
(Refer Slide Time: 50:24)



Usually in some theorems as a particular case you may have to consider the K, some n is equal to 0 it may hold for everything else but may hold for N is equal to 0 you may prove that this sort of proof you have to use and so on. And sometimes you may be able to very easily prove Q is true. Whatever may be P you may be able to prove Q is true. In that case the implication P implies Q will be true and such a proof is called a trivial proof. And sometimes you assume P and then prove Q this is called direct proof, this is third one. And sometimes you will prove the contrapositive, the fourth one, prove the contrapositive NOT Q implies NOT P and this is called indirect proof.

All these four things will be used in some case or the other. These two actually will be used at particular cases some particular cases we have what is meant by proof by cases in that for specific cases may use vacous and trivial proof. But generally mostly we use direct and indirect proofs. Let us take the direct proof; 10x plus 15y is equal to 102, then either x or y is not an integer. You can easily see that dividing by 5 this will be 2x plus 3y dividing by 5 2x plus 5y is equal to 102 and 2 by 5 which is not an integer. Therefore either x or y is not an integer. This sort of a proof is called direct proof.

(Refer Slide Time: 53:26)



Let us see later what is meant by an indirect proof let us give some examples later. We shall also see some more methods of proof by contradiction, proof by cases and so on. There is also something called existence proof, proof by induction and so on. So we shall consider some of them in the next lecture.