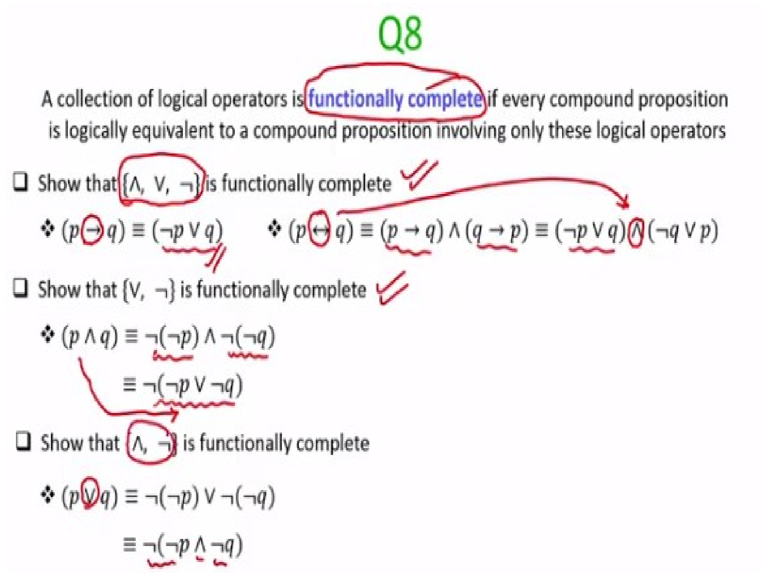
Discrete Mathematics
Prof. Ashish Choudhury
Department of Mathematics and Statistics
International Institute of Information Technology, Bangalore

Lecture -07
Tutorial 1: Part II

Hello everyone, welcome to the second part of the first tutorial.

(Refer Slide Time: 00:24)



So we will start with question 8, in question 8 we are defining a functionally complete set of logical operators, if you are given a set of logical operators, we say it is functionally complete, if every compound proposition can be converted into a logically equivalent proposition involving only the logical operators, that is given in your collection. So the first part of question 8 asks you the following.

So here we want to prove that the set of these three operators is functionally complete. That means any compound proposition you can represent just by using these three operators, so how do we prove this? Well if I am given a proposition which indeed involves only these three operators, I do not have to do anything. But what about a compound proposition where I have an occurrence of implication?

In that case what I can do is I can use this logical identity that $p \rightarrow q$ is logically equivalent to negation of p disjunction q and I can substitute p → q by this RHS expression. And by applying this rule repeatedly wherever I have an occurrence of implication I can remove those implications and I will now have an equivalent formula where everything is represented only in terms of conjunction, disjunction and negation.

What if my expression has bi implication (↔) symbol? I do not have to worry, what I have to do is I can use the identity that the bi implication is nothing but the conjunction of two individual implications and I know that each individual implication can be replaced by these two sub expressions. So now you can see that my original expression is converted into an expression where every operator is either conjunction, disjunction and negation.

So that shows that if you have these three operators namely a conjunction, disjunction and negation, you can represent any statement, any compound proposition and hence this is a functionally complete set of logical operators. Now the second part of the question says that I do not need both conjunction and disjunction to be there. It is sufficient if I just have a disjunction and negation operator and I can represent every statement.

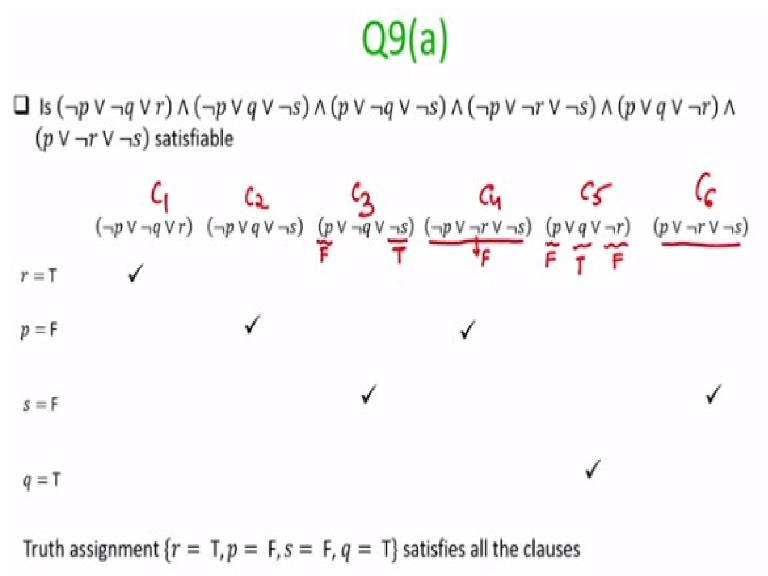
So what I have to do is from the first part of the question, I know that if you have an occurrence of implication you can represent them by just using negation and disjunction, this is what we have shown. What we have to now worry about is how do I represent even a conjunction, namely a proposition where conjunction is involved by an equivalent proposition where I have just occurrences of disjunction and negation.

And this is how we can prove that imagine you have an expression of the form conjunction of p and q. This is logically equivalent to negation of negation of p, conjunction negation of negation of q and then by De Morgan's law, this is nothing but equivalent to negation of this entire expression, namely disjunction of ¬ p and ¬ q. So now you see that even if you have an occurrence of 'and', in your expression you can substitute that expression by another expression where you have only occurrences of negation and disjunction.

And which shows that just your negation operator and disjunction operator are functionally complete. You can represent any statement. The third part now says that you have to show that only the negation operator and the conjunction operator are functionally complete and here we have to show how we can represent a disjunction in terms of conjunction and negation. And again, we have to do similar work which we have done for the previous part, I can represent any disjunction in this form where I have just occurrences of negations and conjunction.

So that shows that just two operators either conjunction along with a negation or disjunction along with negation is sufficient to represent any expression that you are interested in.

(Refer Slide Time: 05:14)



Question 9 asks you to show whether this long expression long compound proposition is satisfiable or not. So what I do here is this expression is already in its conjunctive normal form and what I have done here is I have written down the various clauses that are involved in this compound proposition. And I have to worry and I have to think that how is it possible that I can simultaneously satisfy all these six clauses.

Well, if this expression is satisfiable then there might be many truth assignments which can satisfy all the six clauses, our goal will be to find at least one of them, so let us try to do that. So what I do here is if I ensure that r is true then that will ensure that my clause C_1 will be true, I do not worry what is p and q. If I ensure r is true the disjunction of r with everything will be overall true.

And if I assume r to be true, what happens here is if I go to clause number 4 here, if r is true, then negation of r will be false. Negation of r will be false then what I have to do is in order to satisfy clause number 4, I have to make negation of p to be true or negation of s to be true. So let me make negation of p to be true for that I have to ensure that p is false because if I ensure p is false negation of p will become true then overall this expression C_4 will become true; that is why I have put tick mark here; that means this clause will be satisfied now.

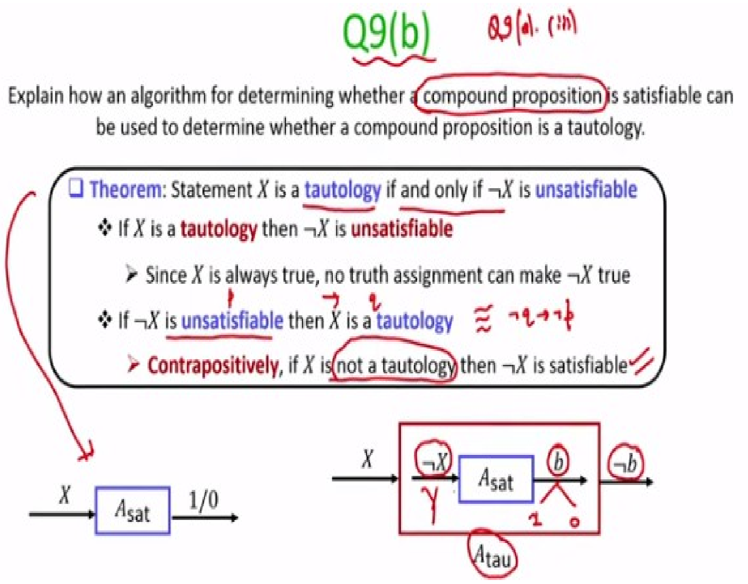
And due to the same truth assignment p equal to false the clause C_2 also will be satisfied because I have an occurrence of $\neg$ p. That means with r equal to true and p equal to false, I will be able to satisfy clause number C_1, clause number C_2, clause number C_4. Now my negation of r will be false because I am assuming here r is true and I am assuming p to be false, then to satisfy clause number 6, I am left with only one option.

Namely I have to ensure negation of s is false, then only clause 6 can be satisfied and negation of s is false means, so what I am doing here is I am not trying to satisfy clause 6 as of now. I am trying to satisfy clause number C_3 first and to satisfy clause number C_3 what I observe here is that p is already taking the value false here and q is not taking any value as of now, I have not assigned any value to q.

But what I observe here is that if I ensure that negation of s is true, then clause C_3 will be satisfied and negation of s is true means s is false. And if I ensure s is false, my clause 6 also gets satisfied because I have an occurrence of $\neg$ s, I do not have to worry what is p and $\neg$ r at all. That means assigning these values to r, p and s, I am able to satisfy all the clauses except clause C_5.

Clause C_5 is not yet satisfied because I have assigned p to be false and I have assigned r to be true, so the negation of r are also false. So the only way I can satisfy clause number 5 is I give the value true to q in clause number C_5 and that means I have found at least 1 truth assignment which can satisfy all these 6 clauses. So now you can see that if your expression is in its conjunctive normal form, you can run this mental algorithm, you can try to individually satisfy each clause at a time and try to come up with a truth assignment which can satisfy all the clauses.

(Refer Slide Time: 09:51)



So in question 9a, there are some other expressions also which are given to you and you have to verify whether they are satisfiable or not, I am leaving that for you. So let me go, sorry this is not 9b, this is question number 9a.3, sorry for this numbering. So the question basically asks you to show the following: it says that imagine you are given an algorithm which can check whether a compound proposition is satisfiable or not.

You do not have to worry about the details of that algorithm. Imagine a box is given to you, you feed some compound proposition to that box and it gives you and yes no answer. Now using that algorithmic box, you have to come up with another algorithm which should tell you whether any input that you feed to that algorithm is a tautology or not. So we first prove a very simple fact here regarding satisfiable statements and tautology.

The claim here is that if you are given a compound proposition X then it is a tautology if and only if negation of X is unsatisfiable and it is very simple, and this is an if and only if statement. So, let me prove it that if X is a tautology that is always be true, then what about the negation of X the negation of X can never be satisfied, you can never find a truth assignment which will make negation of X true because if negation of X is also true and X is also tautology.

Then this is not possible simultaneously. On the other hand you assume that if negation of X is unsatisfiable then I have to prove that X is a tautology, we prove it by contrapositive. So, showing that negation of X is unsatisfiable implies X is tautology is equivalent to showing that if X is not a tautology the negation of X is satisfiable. Our goal is to show $p \rightarrow q$ and this is equivalent to showing $\neg q$ implies $\neg p$.

So if I want to show $p \rightarrow q$ it is equivalent to if it is enough if I showed $\neg q \rightarrow \neg p$ and what is negation of q? Negation of q is X is not a tautology and what is negation of p? That negation of X is satisfiable and indeed this implication that if X is not a tautology then negation of X is satisfiable is a true implication. Because if X is not a tautology it means it is not the case that X is always true.

That means there is one truth assignment for which X is false. For that specific truth assignment, what about negation of X? For that specific truth assignment negation of X will be true because for that assignment X was false. So that proves the implication of this theorem statement in the other direction as well and that is why this is a condition. So now we can utilize this theorem to get an answer for our question.

So as I said earlier you are given an algorithm I call that algorithm as A_{sat}, which takes a compound proposition and it gives you yes no answer. It gives you an answer one if X is satisfiable, it gives you the answer zero if X is not satisfiable. Using this I design another algorithm which I call as algorithm tautology A_{tau}, which will take some compound proposition and it will give me an answer yes if X is a tautology otherwise it will give me an answer no if X is not a tautology.

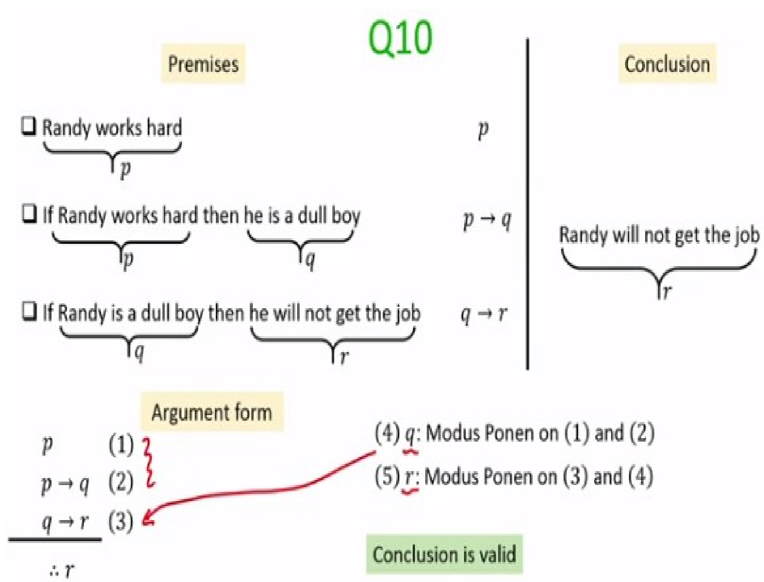
And I am allowed to use this existing algorithm A_{sat}. What I am going to do is my algorithm A tautology will do the following: it will first find a negation of my input X. So I call that expression as Y and I give the compound proposition Y as an input to my algorithm A_{sat}. The algorithm is that will give me a yes no answer. It will give me the answer 1, if Y is satisfiable it will give me the answer 0 if Y is not satisfiable.

What I have to do is, I have to use this response that I am getting from an algorithm A_{sat} to decide the outcome of the algorithm A_{tau}. And my output is the following, if my A_{sat} says that Y is satisfiable, I will say that X is not a tautology, whereas if A_{sat} says that Y is not satisfiable, I will say X is a tautology. That means I will just give the reverse answer, opposite answer which I got with respect to the expression Y from the algorithm A_{sat}.

And this is because of the theorem statement which we have just proved now. We have proved that if A_{tau} X was a tautology then Y will be unsatisfied. That is why I am just flipping or complimenting the bit or the response which I am getting for the expression ¬ X from the A_{sat} algorithm. What will be the running time of the algorithm A_{tau}? The running time will be almost the same as your algorithm A_{sat}, plus the running time that you need to convert your expression X into expression Y.

That means if your algorithm A_{tau} is going to take 1 hour and converting expression X to expression Y takes the 10 minutes then the running time of A_{tau} will be 1 hour 10 minutes, you are almost proportional to the running time of A_{sat}.

(Refer Slide Time: 16:02)

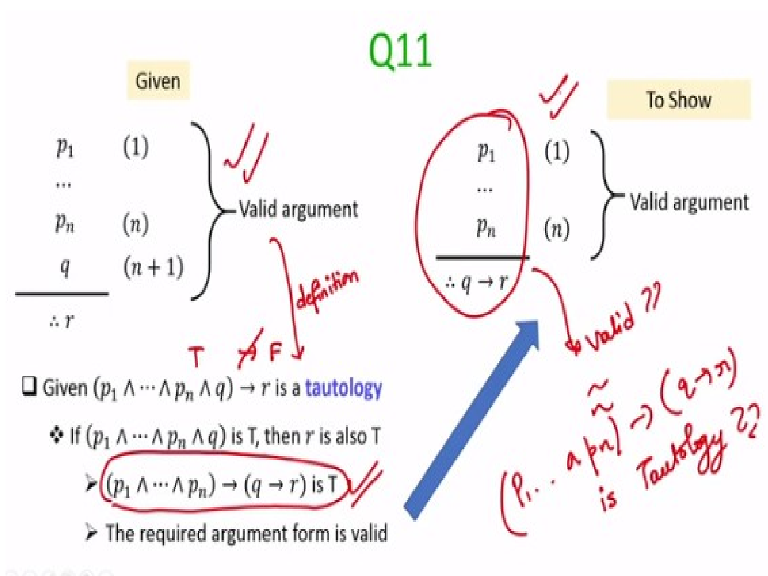


In question 10, you are given a set of premises and a conclusion and you have to verify whether this is a valid argument or not. So what we first do is we convert statements into propositions, so I introduce the variable p here and this is a simple proposition, then the second statement to represent that I introduce another variable q because Randy works hard is already represented by p and the second statement will be represented by then p → q.

For the third statement I need another variable r here to represent a Randy will not get the job. And then the third premise q → r, the conclusion that I am drawing is Randy will not get a job. The argument from here is very simple, you are given three premises and a conclusion is r. Let us see whether this argument form is valid or not, so what I do is I apply Modus Ponen on the first two statements here.

The first two premises here and come to the conclusion q. And then I apply again Modus Ponen on q and third premise and draw the conclusion r. That means this is a valid argument form, a valid conclusion because I can draw the conclusion from my premises.

(Refer Slide Time: 17:31)



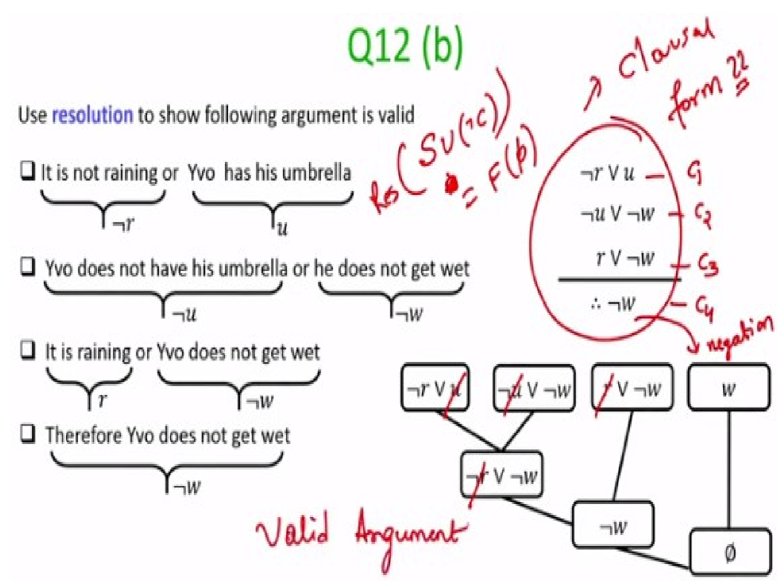
Now in question 11, you are given the following, you are given that this argument form is valid where you are given a set of n premises and $(n + 1)^{th}$ premises is q and the conclusion is r. Now you have to show if this is the case then the argument form where only p_1 to p_n are the premises and the conclusion is q → r is also valid. Again, there are several ways to do this, you can use a truth table argument and so on, we will avoid that.

Since we are given that this argument form is valid as per the definition of valid argument, I can say that conjunction of p_1 to p_n and q → r is a tautology that means it is never possible that your left hand side is true and RHS is false, that is not going to happen. That means if my LHS is true, RHS is also true. That means if the conjunction of p_1 to p_n and q is true then r is also true and as a result I can say that this implication is also true.

Because if the conjunction of p_1 to p_n and q is true, that means this part is definitely to this conjunction of p_1 to p_n is true. And since q is true here r is also true, then true implies true is anyhow true and true implies true is anyhow true. But if I closely see here, what does exactly this implication means? If I say that this implication is always true then another form of the implication is that you have the premises p_1 to p_n and the conclusion is $q \rightarrow r$.

And if I want to say that this argument form is valid, this is equivalent to asking whether the (conjunction of p_1 to p_n) $\rightarrow$ ($q \rightarrow r$) is tautology and that is what we have proved here. That means if this has been given, this argument form is given to be a valid argument form, then this new argument form is also valid.

(Refer Slide Time: 20:20)



Now, let me go to question 12. I will not be solving part A here, I will be focusing on part B, part A I am leaving for you. You have to show using resolution whether the following argument is valid. So, these are the premises here, these are the three premises and this is the conclusion. So as usual the first thing that we will do is we will introduce propositional variables and bring everything in the form of compound propositions.

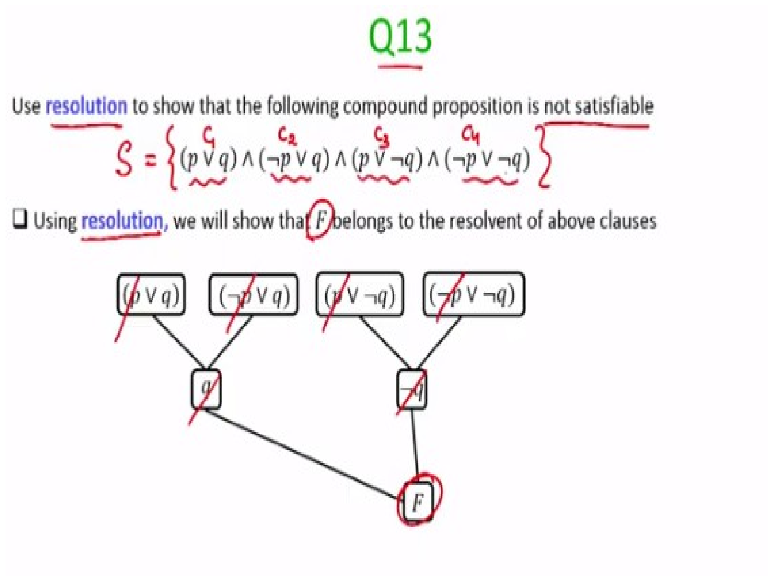
So, if I introduce the variables here like this, it is up to you what form of the variable you use. I am using it is not raining as ¬ r. You could have used r, it is not raining. In that case, it is raining and becomes negation of r and so on and you can use any name, you can use A, B, C for propositional variables, just for my convenience that I am using these names. Now if I use these propositional variables then the argument form here is the following.

And I have to show whether this argument form is valid or not. The first thing I have to check is whether this argument form is in its clausal form or not, that means everything the premises and conclusion everything is in the form of clauses or not and in this case yes, this is clause C_1, this is clause C_2, this is clause C_3 and this is clause C_4. What is the resolution refutation method? The resolution refutation method says that you take the set of clauses which are your premises and to that you add the negation of your conclusion.

Remember we have to check whether s union negation of the conclusion if the resolvent of this thing is false or not if that is the case then I say that C is a logical conclusion from my set of premises in the set s, that is what is the proof by resolution refutation, so I have added a negation of the conclusion and now I have to resolve. So, I take the first two classes here, I cancel out u here and I get a resolvent disjunction of ¬ r and the ¬ w.

Then I choose the resolvent and the next clause here and I cancel out r and r here and I get negation of w and now if I take negation of w and w I cancel them and get false. So since I am getting the resolvent to be empty that means this is a valid argument.

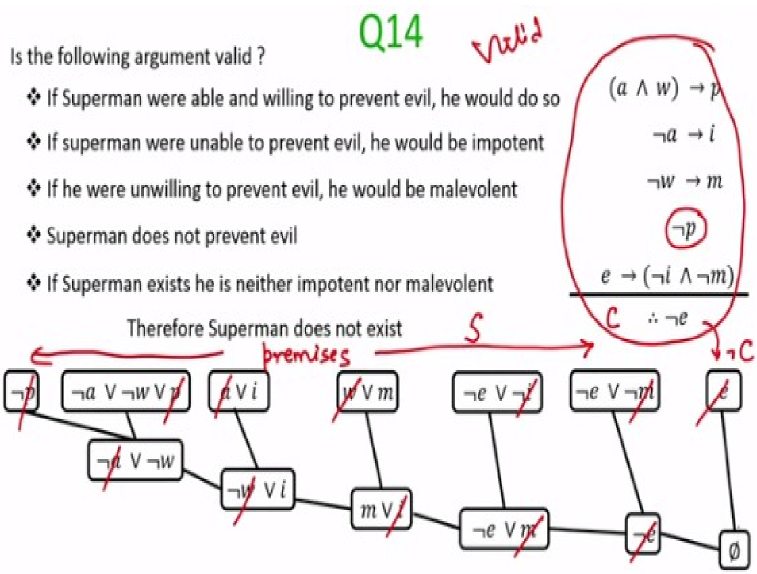
(Refer Slide Time: 23:28)



So now let us next go to question number 13 and in question number 13, we have to use resolution to show that the following compound proposition is not satisfiable and as per the properties of resolution basically to show that the conjunction of these clauses is unsatisfiable. I have to show that the constant F belongs to the resolvent of the above clauses. So let us build a resolvent or resolution tree for this set of clauses here.

So I can pick the first two clauses and resolve p and then I can pick the last two clauses and I can again resolve p. So the resolvents are now added to the tree and now I can pick these two resolvents for resolving and I obtain the conclusion, the constant, F which shows that the conjunction of these four clauses is not satisfied. So that is how you can actually prove whether a kind compound proposition is satisfiable or not you have to bring everything in the clause form and then build a resolvent, resolution tree and then arrive at the constant F.

(Refer Slide Time: 25:15)



Let us go to the last question for the first tutorial. So again, you are given a set of English statements and you have to verify whether it is a valid argument or not. So what I do here is as I am repeatedly doing it I will be introducing propositional variables and converting each statement into some compound proposition, you are free to use any variable name here, I am just using these variables for my convenience.

After converting everything, this is the argument form here and I am going to use proof by resolution refutation because I find it very comfortable because we just have to keep on canceling clauses here. Cancelling literals in two clauses and keep on doing the simplification till you either get an empty conclusion or you cannot resolve further. You do not have to worry about logical identities, De Morgan law etc etc. That is why resolution refutation is a very very powerful proof mechanism.

So these are your set of premises converted into their equivalent clauses, so in this case I have to convert some of the premises into their corresponding clause form or cnf form. In fact I have to convert everything because none of the premises are available in their cnf form except this ¬ p. Negation p is available in its cnf form; everything else has to be converted, after converting I have added all the premises, so that is my set of clauses or premises s. And what is the conclusion?

The conclusion that I am trying to draw is negation of e which is already in its cnf form but I have to add the negation of C in the resolution tree to do the resolution refutation proof mechanism. And now I have to resolve, so I start with the first two things and cancel out p and then I cancel out a, then cancel out w, then cancel out i, next I cancel out m and then after canceling out e I have left with nothing, empty.

So I got a resolvent to be empty and that shows that this is a valid argument. Well you could have shown that this argument form is valid by using simplification, rules of inferences, Modus Ponen etc etc not stressing that you have to only use resolution refutation. Just that I find the resolution refutation to be a simpler proof mechanism. And again the tree that I have constructed here need not be the only tree when you are building the resolution, when you are doing the resolution refutation proof.

You might pick the pair of clauses to resolve in any arbitrary order. It is just that some trees lead you to the empty conclusion very soon; some trees might lead you to the empty conclusion after a long time. So depending upon which two clauses you cleverly use at each stage that will determine how fast you reach to the empty conclusion that is all. So with that, I conclude the first tutorial. Thank you.