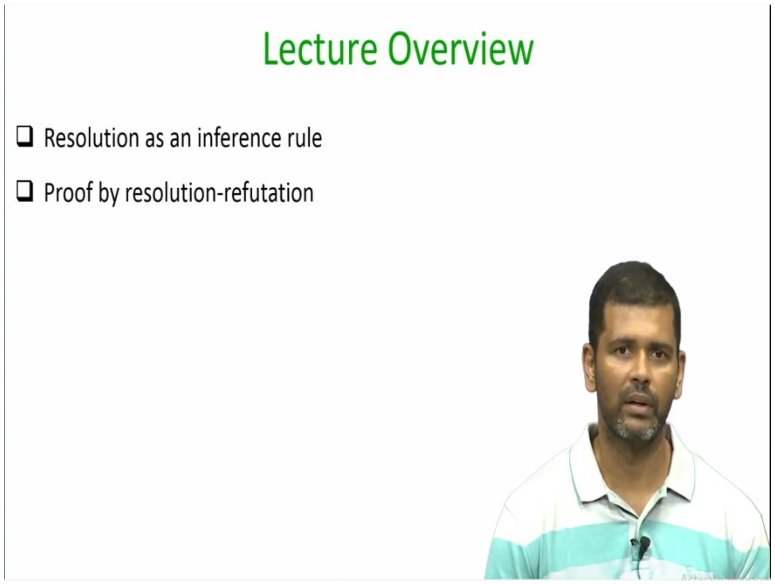
Discrete Mathematics

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

Lecture -05
Resolution

Hello everyone, welcome to this lecture on resolution just a quick recap.

(Refer Slide Time: 00:27)

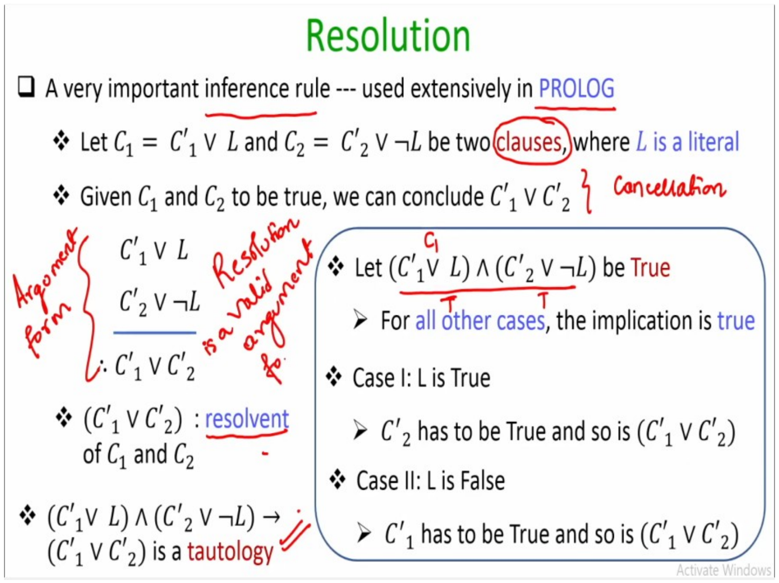


Lecture Overview

- Resolution as an inference rule
- Proof by resolution-refutation

In the last lecture we discussed about valid arguments, argument form when exactly we say an argument form to be valid and so on and we also saw various rules of inferences. The plan for this lecture is as follows in this lecture we will discuss about resolution which is an important influence rule and based on resolution we will see a proof strategy which is called as proof by resolution refutation.

(Refer Slide Time: 00:52)



Resolution

- A very important inference rule --- used extensively in PROLOG
  - Let $C_1 = C'_1 \vee L$ and $C_2 = C'_2 \vee \neg L$ be two clauses, where L is a literal
  - Given C_1 and C_2 to be true, we can conclude $C'_1 \vee C'_2$ } Cancellation

Argument form

$$\begin{array}{l} C'_1 \vee L \\ C'_2 \vee \neg L \\ \hline \therefore C'_1 \vee C'_2 \end{array}$$

Resolution is a valid argument form

- $(C'_1 \vee C'_2)$: resolvent of C_1 and C_2
- $(C'_1 \vee L) \wedge (C'_2 \vee \neg L) \rightarrow (C'_1 \vee C'_2)$ is a tautology

- Let $\underbrace{(C'_1 \vee L)}_{T}^{C_1} \wedge \underbrace{(C'_2 \vee \neg L)}_{T}$ be True
  - For all other cases, the implication is true
- Case I: L is True
  - C'_2 has to be True and so is $(C'_1 \vee C'_2)$
- Case II: L is False
  - C'_1 has to be True and so is $(C'_1 \vee C'_2)$

So to begin with, let us try to understand what exactly is the resolution rule. It is a very important inference rule and it is used extensively in this programming language called PROLOG. So recall I said that PROLOG is an important programming language, which is used in AI applications. So what exactly is this resolution rule? So it says the following, imagine you are given two clauses. So C_1 is the clause and C_2 is another clause.

And the important property here is that I have a literal L which is present in positive form in C_1 and negative form in C_2. So you can imagine that C_1 is a huge clause consisting of one or more literals, one of the literals is L. So just to recall a literal is propositional variable or the constants True or False. So what I am saying here is that we have two clauses C_1 and C_2. In C_1, we have some literal L and the same literal is available in a negation form in C_2.

The remaining portion of C_1, I am denoting it as C_1' and the remaining portion of C_2, I am denoting it as C_2'. So you have 2 such clauses and what this resolution rule says is the following. It says that, if it is given that the clause C_1 and C_2 are true, then based on the truth of these 2 clauses we can conclude, conclusion $C_1' \vee C_2'$. So in some sense you can imagine that resolution rule is something equivalent to cancellation rule.

That means you can cancel out the literal L if it is available in positive form in C_1 and negative form in C_2 and whatever is left in C_1 and C_2 you take the disjunction of that will be the conclusion of C_1. and C_2. So in some way you are actually simplifying your clause C_1 and C_2. So in argument form the resolution rule can be stated as follows. So this is the argument form of resolution inference rule.

It says that if you are given the clauses C_1 and C_2 where, C_1 is $C_1' \vee L$ and C_2 is $C_2' \vee \neg L$. Then based on these two premises, you can conclude the conclusion $C_1' \vee C_2'$. I stress that to apply the resolution rule you need C_1 and C_2 to be clauses. That means C_1 and C_2 have to be compound propositions which are available in the form of clause. It should not be available in a different form.

So the conclusion that we can draw from the resolution rule namely the disjunction of C_1' and C_2' is also called as the resolvent of the clauses C_1 and C_2. That means after resolving the clause C_1 and C_2 we are getting the resolvent $C_1' \vee C_2'$. And remember as per our definition of argument forms since we are saying that our resolution is a valid argument form.

And a definition of valid argument form is that conjunction of premises implies conclusion should be a tautology. That was our definition of a valid argument. Then since we are saying that resolution as a valid inference rule we will prove that, assume for the moment resolution is a valid inference rule, it means that we can say that the conjunction of clauses C_1 and C_2 where C_1 and C_2 have the common literal L available in positive as well as in negative form in C_1 and C_2 respectively implies the disjunction of C_1' and C_2' is a tautology. It will always be a true statement we will prove that very soon. So, that is the resolution. So now we want to prove that indeed the resolution is the, indeed resolution in principle that we are stating here is a valid argument form. So what we have to prove is we want to prove this statement that indeed this implication is a tautology. So for that we assume that a left hand side of this implication namely the conjunction of C_1 and C_2 is true.

Why we are assuming it to be true because remember we want to show that this implication is a tautology and this implication is true for all other cases. Remember the truth table of implication of false $\rightarrow$ false is true, false $\rightarrow$ true is true and in the same way true $\rightarrow$ true is true. So for these three cases by default this implication is always a true statement we have to consider the fourth case when your left hand side of this implication is true and we have to show in that case the right hand side of the implication is also true.

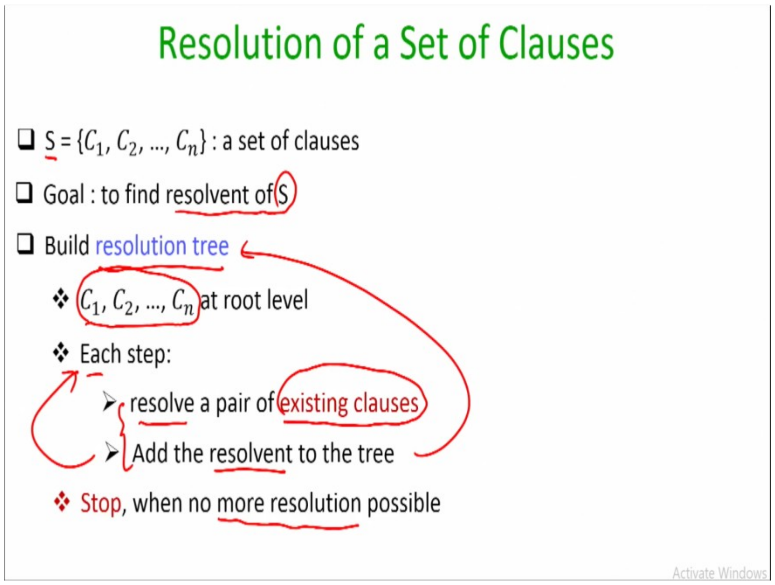
That will prove that this implication is indeed a tautology. So that is why I am assuming here that the left hand side of your implication is true. So now I can split my proof into two cases depending upon whether my literal L which is available in positive form and negative form in C_1 and C_2 respectively is true or not. So if L is true since I am assuming that this whole conjunction is a true statement and since L is true, that means this portion here, this portion of your left hand side. Since I am assuming it to be true this has to be true right the disjunction of C_2‘ and negation of L has to be true because then only the overall conjunction can be true. But since I am assuming L to be true negation of L will be false. And if negation of L is false then in order that is the overall C_2 should be true. I require that C_2‘ should be true.

And if C_2‘ is true then you take the disjunction of C_2‘ with anything, say with C_1‘ the overall disjunction will always be true. So that proves that this implication is a tautology for case one that means if you assume your left hand side is true of this implication and if L is true, then I draw the conclusion that even RHS is also true. Now take the second case when L is false, so these are the two possible cases with respect to the literal L, can be either true or it can be either false.

So if you literal L is false then I focus on C_1 this is your whole C_1. And since I am assuming that this overall conjunction is true, this overall conjunction will be only if the individual clauses here are true. But if I focus on the clause C_1 I am assuming case two where L is false. Then in order that C_1 is true, C_1‘ has to be definitely true. Because if C_1‘ is also false and if L is also false your C_1 can never be true.

But I am assuming that my left hand side is true. So now if C_1‘ is true, I take the disjunction of C_1‘ with anything the overall disjunction will be true. So that proves that even for case 2 my RHS is true and that proves that, this implication that I have stated here is indeed a tautology. And since it is a tautology as per my definition of valid argument form I can say that resolution is indeed a valid inference rule or it has a valid argument form and hence the corresponding inference rule is indeed valid. Resolution is a valid argument form.

(Refer Slide Time: 09:35)



Resolution of a Set of Clauses

- ❑ S = $\{C_1, C_2, \ldots, C_n\}$: a set of clauses
- ❑ Goal : to find resolvent of S
- ❑ Build resolution tree
  - ❖ $C_1, C_2, \ldots, C_n$ at root level
  - ❖ Each step:
    - ➢ resolve a pair of existing clauses
    - ➢ Add the resolvent to the tree
  - ❖ Stop, when no more resolution possible

So we have seen already how to find or how to resolve a pair of clauses, now next we want to see how exactly we resolve a set of clauses where we may have more than two clauses. So imagine you are given a set of n clauses and I would be interested to resolve the clauses in this set, which is often called as a resolvent of the set of clauses. So the idea remains the same that means we will keep on finding two clauses from this collection and keep on resolving them and we stop till we cannot proceed further.

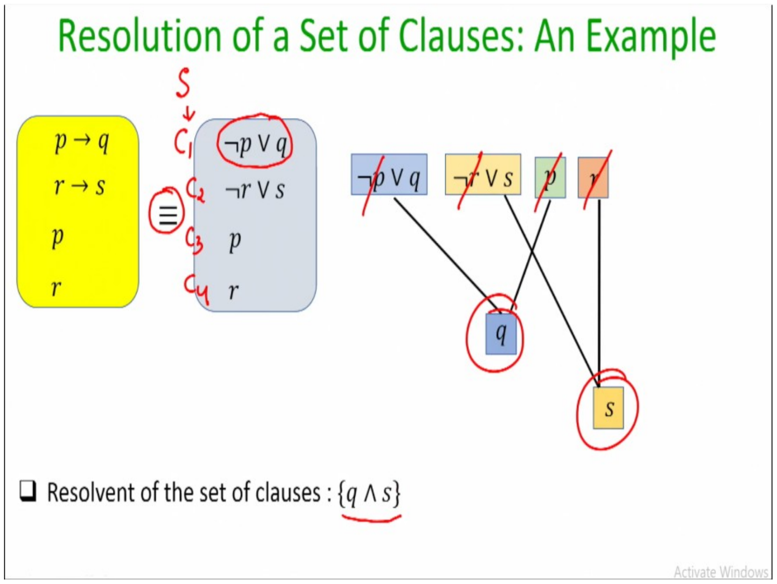
So basically we build what we call as a resolution tree and in the resolution tree we can keep the n clauses that are given in my set S at the root level, that means by default we can imagine that the clauses C_1, C_2 C_n each of them belongs to the resolvent of my set of clauses S because I can always conclude C_1, I can always conclude C_2 and I can always conclude C_n from my set of clauses in S.

Now, next what I have to do is the following I have to resolve a pair of existing clauses which are already there in my tree and whatever is the resolvent I obtain by resolving the pair of clauses, which I have resolved that will be treated as a new clause which will be again added to my tree.

And then I go to this step and again pick two clauses which I can resolve, I find a resolvent and again I add them to the resolution tree and I stop this process when I cannot find any more clauses which can be resolved at that step I stop. That is how I can find a resolvent for a set of clauses. I stress here that there is no restriction at each step regarding the choice of the clauses which you can pick for resolving in what order you have to resolve the clauses and so on.

As long as you are picking two clauses which can be resolved and adding the resolvent to the tree you are fine.

(Refer Slide Time: 11:57)



Resolution of a Set of Clauses: An Example

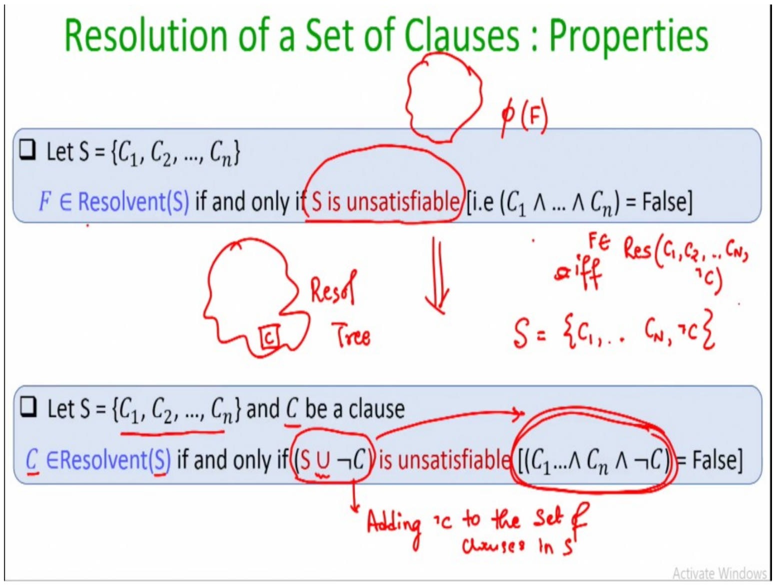
❑ Resolvent of the set of clauses : $\{q \wedge s\}$

So let me give you an example to show how exactly we compute the resolution of a set of clauses, so imagine you are given here compound propositions p → q, r → s, p and r. So the first step will be that I will be converting this compound propositions into their corresponding clause form because as of now p → q is not in its clause form. But by applying the rules of logical equivalence, I can convert p → q into ¬ p disjunction q and so on.

So now I obtain clauses C_1, C_2, C_3, C_4 and this will be my set S. So now here is how I can build my resolution tree at the root I can pick, I can keep all the clauses that are there currently in my set S. And now I keep on picking clauses, pair of clauses, which I can resolve. So for instance, I can resolve these two clauses by cancelling p and ¬ p and the resolvent will be q which will be now added to the tree.

Next, I can resolve r from these two clauses and a resolvent will be S and now you can see that I can no longer find a pair of clauses which can be resolved further and I stop here and hence I will say that the resolvent of the set of clauses consist of the conjunction of clause q and the conjunction of the clause s. So that is how we actually built the resolution of a set of clauses.

(Refer Slide Time: 13:35)



Resolution of a Set of Clauses : Properties

- Let S = $\{C_1, C_2, \ldots, C_n\}$
  $F \in$ Resolvent(S) if and only if S is unsatisfiable [i.e $(C_1 \wedge \ldots \wedge C_n)$ = False]
- Let S = $\{C_1, C_2, \ldots, C_n\}$ and C be a clause
  C ∈Resolvent(S) if and only if $(S \cup \neg C)$ is unsatisfiable $[(C_1 \ldots \wedge C_n \wedge \neg C)$ = False]

ϕ(F)

Res^n Tree

$\Downarrow$

$F \in$ Res$(C_1, C_2, .., C_N, \neg C)$

iff

$S = \{C_1, .. C_N, \neg C\}$

Adding $\neg C$ to the Set of clauses in S

Activate Windows

Now, I will be discussing two key properties regarding the resolution of a set of clauses based on which we will see a very nice proof strategy which we call as proof by resolution refutation. So the first property here is the following, imagine you are given a set of n clauses. Now, the claim here is that the empty clause or the constant False, or the false statement you can imagine you can interpret it in different manner.

So this is the constant False which is also denoted by the notation ϕ in some text that I will be interchangeably using the constant F as well as ϕ for denoting the false statement or the false constant. So the claim here is that a constant false will belong to the resolvent of set of these clauses if and only if this, the set of clauses in S is unsatisfiable. What does that mean? That means that if the conjunction of the n clauses is always false that is what it means when I say that the set of clauses in S is unsatisfied.

If the set of clauses is unsatisfiable that means no truth assignment of the clauses C_1 to C_n can satisfy to make it true. That means it is always false. So the statement here is that if the set of clauses in S is unsatisfiable, then when you build the resolution tree for resolving the set of clauses in S, you will see that, the constant F appears in the tree. So due to interest of time I am not going to give you the proof for this but you can easily verify that this is indeed true.

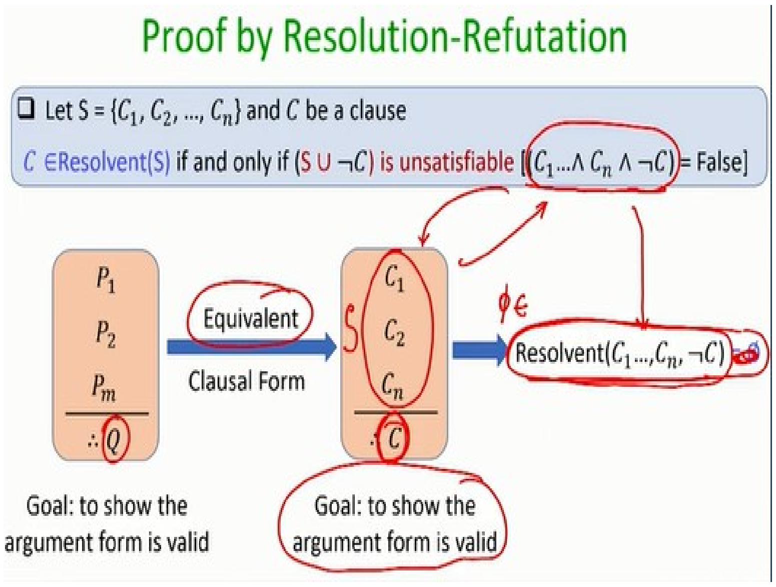
And in fact later on we will demonstrate the statement with an example. Now, based on this statement I can prove another statement which states the following. So here again you are given with the clauses C_1 to C_n and suppose C is another clause. Now the statement says that, the clause C belongs to the resolvent of the set of clauses in S if and only if the set of clauses in S, along with the clause $\neg$ C is unsatisfiable.

So this union here means that I am adding the Clause $\neg$ C to the set of clauses in S. That means I am basically taking the conjunction of the existing clauses in the set S and the clause $\neg$ C. So the statement says here that the clause C will belong to the resolvent of S, if I build a resolution tree, for the set of clauses in S, I will see a node with label C and the statement says that this is possible if and only if the conjunction of the clauses in S along with $\neg$ C is false.

And by the previous statement I know that a set of clauses is unsatisfiable if and only if the constant False belongs to the resolvent of the set of clauses. So when can it be possible, so now my set S is modified to C_1 to C_n and along with that $\neg$ C. So when can I say that the set S consisting of the n clauses along with negation of C is unsatisfiable well, that will be unsatisfiable if and only if the resolvent of C_1, C_2 and C_n and negation of C, gives me constant False.

That means, I can say that a constant False belongs to the resolution of resolvent of the set of n clauses along with negation of C. So that is the second statement.

(Refer Slide Time: 18:23)



Proof by Resolution-Refutation

- Let S = $\{C_1, C_2, \ldots, C_n\}$ and C be a clause

$C \in$Resolvent(S) if and only if ($S \cup \neg C$) is unsatisfiable [$(C_1 \ldots \wedge C_n \wedge \neg C)$= False]

P_1
P_2
P_m
$\therefore Q$
Equivalent
Clausal Form
S
C_1
C_2
C_n
$\therefore C$
$\phi \in$
Resolvent($C_1 \ldots, C_n, \neg C$)
Goal: to show the argument form is valid
Goal: to show the argument form is valid

So based on these two properties, I can next discuss what we call as proof by resolution refutation, so in the proof by the resolution refutation the goal is the following you are given an argument form and you have to verify whether this argument form is valid or invalid. Namely, I have to check whether the conjunction of premises implies conclusion is true or not, or equivalently whether Q can be concluded logically from the conjunction of my premises.

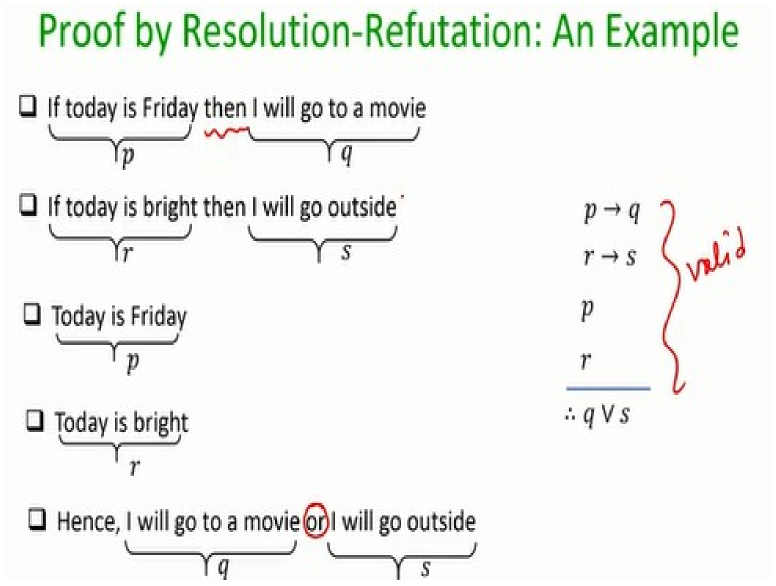
Now the first thing that I do in the proof by resolution refutation is I can convert my premises as well as conclusion into its equivalent clausal form because the premises P_1, P_2, P_m may not be available in clausal form, so I have to convert them into the clausal form in the same way my conclusion also need not be available in the clausal form. So I convert it into it is clausal form and now my goal is to verify whether the equivalent argument form where everything is in the form of clauses is valid or not.

Namely, I have to check whether C belongs to the resolvent of this set of clauses. So this is my set S, I want to check whether I can conclude C from this set of clauses in S which is equivalent to saying whether I want to check whether the clause C belongs to the resolvent of the set of clauses and for that as per this property, I have to check whether the conjunction of my premises along with the negation of the conclusion is unsatisfiable or not.

And that will be unsatisfiable only if the resolvent of my premises along with the negation of the conclusion is empty or not. Actually it is not exactly equivalent to checking whether the resolvent is empty or not is actually check equivalent to checking whether ϕ belongs to the resolvent of this or not, because the resolvent of this set of clauses may consist of many clauses. Among those clauses I have to check whether one of the clauses is empty close or not.

If that is the case then it will show that this collection of clauses is unsatisfiable which will show that this argument form is a valid argument form.

(Refer Slide Time: 21:00)



Proof by Resolution-Refutation: An Example

- If today is Friday then I will go to a movie: $\underbrace{\text{If today is Friday}}_{p}$ then $\underbrace{\text{I will go to a movie}}_{q}$
- $\underbrace{\text{If today is bright}}_{r}$ then $\underbrace{\text{I will go outside}}_{s}$
- $\underbrace{\text{Today is Friday}}_{p}$
- $\underbrace{\text{Today is bright}}_{r}$
- Hence, $\underbrace{\text{I will go to a movie}}_{q}$ or $\underbrace{\text{I will go outside}}_{s}$

$$\left.\begin{array}{l} p \rightarrow q \\ r \rightarrow s \\ p \\ r \end{array}\right\} \text{valid}$$
$$\therefore q \vee s$$

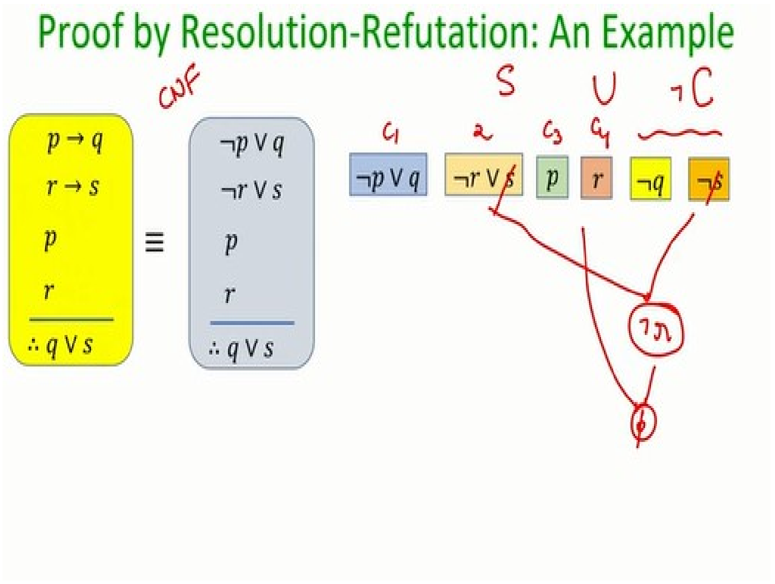
So, let me demonstrate it with an example that will make things clear, so I am given a bunch of premises you are given premise number 1, 2, this is your 1, 2, 3 and 4 premises and this is your conclusion and I want to verify whether this is a valid argument form using proof by resolution refutation method. The first thing I will do is, I will convert each of these English statements into an abstract argument form where everything will be in terms of propositional variables.

So for doing that, I will introduce propositional variables to represent various statements. So let p denote a statement today is Friday and q denote the statement I will go to a movie, if that is the case then the first premise can be represented by p → q because this is a statement of the form if-then. Then let r denote the statement today is bright and s denote the statement I will go outside, then the second premise can be represented by r → s.

I have already introduced the variable p for denoting the statement that today is Friday, so the third premise is p. I have already introduced a variable r to denote the statement today is bright, so the fourth premise will be r. And what is the conclusion I am trying to do well I am trying to do a conclusion that the disjunction of q and s. So I have to verify whether this argument form is valid or not.

Well you can use truth table method or you can keep on applying Modus Ponen and simplification rule and so on to verify whether these argument form is valid or not. But goal of this exercise is we will show that how this argument form is valid or not, how we can check that using the resolution refutation method.

(Refer Slide Time: 22:56)



Proof by Resolution-Refutation: An Example

The first thing that we have to do is we have to convert the premises as well as the conclusion everything in the form of equivalent clauses. So p → q is not in the form of clause, so we have to convert p → q into its clause form and I can rewrite p → q in the form ¬ p or q by the way if you are wondering how we are converting statements into its equivalent clause form that is nothing but converting statements into its conjunctive normal form.

If you have to conjunctive normal form equivalent of the original statement that is nothing but the clause form of your original statement. So, if I convert p → q and to conjunctive normal form I get ¬ p or q, r → s is equivalent to ¬ r or s, p is anyhow in it is conjunctive normal form r is anyhow in its conjunctive normal form and my conclusion is already in it is CNF. So that is the equivalent clause form.

Now I have to verify whether this is a valid argument form or not and the proof by resolution refutation says the following you take the premises. That is your set of clause s and you add the negation of the conclusion to that and what will be the negation of the conclusion. The negation of the conclusion will be negation of $q \vee s$. I can apply the De-Morgan's law and take $\neg$ q, take the negation inside and these are the two clauses corresponding to my conclusion which I am adding to my resolution tree.

And now I have to, so the four clauses was your set s and these two clauses correspond to your negation of conclusion and now I have to find a resolution of this set of clauses s union of $\neg$ c and check whether I get the conclusion false or not. Again, I can pick any pair of clause and keep on resolving, so what I do is I choose clause number c_1 and c_3 to resolve because p is available in positive and negative form, I can cancel out.

And resolvent of c_1 and c_3 will be q. Then, let q be the clause which I pick and negation of q is the second clause which I am choosing here and I can cancel them and I obtain False because if you take q and $\neg$ q and cancel out your left with nothing and that is a false conclusion. Since, I have obtained a false conclusion that means this argument form in it is clausal form is indeed valid.

Remember you might be wondering here that I am not using c_2 and r and negation of s in my resolution process. That is not necessary when I am constructing, when I am doing the resolution refutation proof, my goal is to arrive at the false conclusion as soon as possible for that it is not necessary to touch upon each and individual clauses in my tree, it might be possible that just by using two clauses at the first place I can arrive at a conclusion false.

That will complete the process I need not have to touch upon the other clauses that is not necessary. So, in this particular case just by resolving c_1 and $\neg$ q, c_1 and c_3 I can come to the conclusion q and then by resolving q and $\neg$ q, I can come to the conclusion. Well, you can do the proof other way around as well in the sense that you can do the proof differently as well that means instead of say resolving c_1 and c_3 first, you can do the following as well, you can resolve this c_2 and negation of s you can cancel out and you can come to the conclusion negation of r.

And then you can choose this r and negation of r also to derive at the false conclusion that is another resolution refutation proof. So there can be multiple resolution refutation proof or proving the validity of the same argument you just have to come to the conclusion ϕ if at all it is possible well if you cannot come to the conclusion ϕ even after repeatedly applying or resolving pair of clauses that shows your argument form is not valid.

So in this case the argument form is valid, the clausal form of the argument form is valid that shows that the original argument form is also valid. So that brings me to the end of this lecture the references used for today's lecture are the chapters in the Rosen's book and just to summarize in this lecture we have introduced the resolution refutation proof strategy, which is based on the resolution inference rule.

The resolution inference rule can be considered as an equivalence form of proof by cancellation where you pick two clauses where you have a literal available in positive form in one of the clauses and negative form in other clause. You can cancel both literals and whatever is left in both the clauses the conjunction of that you can consider as the resolvent or the conclusion of the two clauses which you have simplified.

This is a very powerful proof mechanism, which is very extensively used in programming language PROLOG, thank you.