

**Logic for CS**  
**Prof. Dr. S. Arun Kumar**  
**Department of Computer Science**  
**Indian Institute of Technology, Delhi**

**Lecture - 32**  
**Resolution and Tableaux**

So, we were doing Resolution last time and we something about completeness which we have to do. But, so today will essentially do we are also doing Tabular Methods. Because, we have already created huge foundations with propositional logic the extensions to first order logic therefore are quite simple in nature. But, let us first before we get on to the tabular method let us just go back to resolution because of because, there is a issue of completeness.

(Refer Slide Time: 01:13)

**Soundness of FOL Resolution**

**Lemma 31.1** *The resolvent  $C'_{ij}$  obtained by resolving the clauses  $C_i$  and  $C_j$  in the **resolution method** is a logical consequence of the set  $\{C_i, C_j\}$ .* □

The following theorem then follows.

**Theorem 31.2** *If  $S'$  is the set of clauses obtained by a single application of the resolution rule **Res1**, then  $S \models S'$ .* ■

**Corollary 31.3** *If the empty clause is derivable from a set  $S$  of clauses, then  $S$  is unsatisfiable.* ■

NPTTEL

So, the Resolution method are basically tries to take as many clauses as possible with complimentary pairs. And, essentially the resolvent that you get by resolving the clauses  $C_i$  and  $C_j$  again some atom  $p$  which for which they have complimentary pairs of literals uses unification.

(Refer Slide Time: 01:48)

**Resolution in FOL**

Let  $S$  be a set of clauses,  $C_i, C_j \in S$  with  $i \neq j$ ,  $FV(C_i) \cap FV(C_j) = \emptyset$  and  $p$  an atomic predicate symbol such that

- $C_i = C'_i \cup \{p(s_{i'}^{\vec{t}}) \mid 1 \leq i' \leq m_i\}$  and  $C_j = C'_j \cup \{\neg p(t_{j'}^{\vec{t}}) \mid 1 \leq j' \leq m_j\}$
- $L = \{p(s_{i'}^{\vec{t}}) \mid 1 \leq i' \leq m_i\} \cup \{p(t_{j'}^{\vec{t}}) \mid 1 \leq j' \leq m_j\}$  is a set of unifiable literals.
- $\mu = \text{UNIFY}(L)$  is an mgu of  $L$ .
- $C'_{ij} = \mu(C'_i \cup C'_j) = (\mu C'_i) \cup (\mu C'_j)$  is called the *resolvent* of  $C_i$  and  $C_j$ .

**Res1** 
$$\frac{S}{(S - \{C_i, C_j\}) \cup \{C'_{ij}\}}$$

NPTEL

So, this is the resolution thing so you take this set of clauses which have  $p$  as an atom and the set of clauses which have  $\neg p$  as an atom the terms the arguments may be different. So, what you do is you choose this set and try to unify it find a most general unifier for some set of these clauses containing complimentary pairs. And, having found the most general unifier you essentially remove these clauses and you add this new resolvent  $C_{ij}$  prime. Which, is obtained by applying that unifier and taking the union of these pairs of clauses. So, one of the things is that the resolvent  $C_{ij}$  prime that is obtained by this kind of resolution is actually a logical consequence.

(Refer Slide Time: 02:56)

Let  $A \models \{C_i, C_j\}$ . Therefore

$$A \models \bigwedge [\bigvee C_i] \quad (13)$$
$$A \models \bigwedge [\bigvee C_j] \quad (14)$$

and for any substitution  $\theta$  we have

$$A \models \theta \bigvee C_i \quad (15)$$
$$A \models \theta \bigvee C_j \quad (16)$$

If  $\theta$  is a unifier of  $L$  and  $\theta L = \{\lambda\}$  we get

$$A \models \bigvee ((\lambda) \cup \theta C'_i) \quad (17)$$
$$A \models \bigvee ((\bar{\lambda}) \cup \theta C'_j) \quad (18)$$

That is something that we fairly that is fairly easy to see you can actually start with a module A reinterpret sets of clauses in terms of the propositional connectives. And, fact that they are implicitly universally closed and you can do a case analysis and you will get this. So, that was one thing if  $S$  prime is a set of clauses obtained by a single application of the resolution rule arise one. Then, of course  $S$  prime is a logical consequence of  $S$  so that follows from this Lemma. And, if the empty clause is derivable from set  $S$  of clauses then  $S$  is unsatisfiable and this is what is known as refutation if you like.

(Refer Slide Time: 03:43)

**Ground Clauses**

**Theorem 31.4** (Completeness of Resolution Refutation for ground clauses). Let  $G$  be a set of ground clauses. If  $G$  does not possess a model, the empty clause  $(\perp)$  may be derived by **Res0**. □

Here **Res0** is the (propositional) resolution rule given by

$$\text{Res0} \frac{S}{(S - \{C_i, C_j\}) \cup \{C'_{ij}\}}$$

where  $C'_{ij} = C'_i \cup C'_j$ . Note that there is no substitution involved anywhere since all clauses are ground.

HPTTEL

So, one of the things that we proved was that and this is a this theorem for Completeness of a Resolution Refutation for ground clauses also holds for I mean you can think of it as a proof of completeness of propositional resolution. Because, essentially we looked at ground terms in the predicates and there was no unification because everything was variable free. And, we just took exact complimentary pairs and did the resolution so this is like propositional. So, Res0 stands for propositional resolution. So, there is no substitution involved in this.

(Refer Slide Time: 04:27)

**Proof of theorem 31.4.**

*Proof:* Let  $G = \{C_i \mid 1 \leq i \leq n, n > 0\}$  be a set of  $n$  clauses. If  $(\perp) \in G$  there is nothing to prove. So assume  $(\perp) \notin G$ . Consider the following measure

$$\#G = \left( \sum_{1 \leq i \leq n} |C_i| \right) - n$$

Clearly,  $\#G = 0$  iff every clause is made up of a single literal. We proceed to prove the theorem by induction on  $\#G$ .

**Basis.**  $\#G = 0$ . Then each  $C_i = \{l_i\}$  and  $G \equiv \bigwedge_{1 \leq i \leq n} l_i$  and by theorem 25.10,  $G$  is unsatisfiable iff it contains a complementary pair. Clearly by rule **Res0** the resolvent of this complementary pair is the empty clause.

**Induction Hypothesis (IH).**

HPTTEL

And, this is a theorem that so we can proof essentially by looking at this measure. So, you have the  $G$  of ground clauses and the empty clause if the empty clause is already in  $G$  that is nothing to prove. So, that needs a entire set of clauses is essentially unsatisfiable. So, if the empty clause is not in  $G$  we consider this measure hash  $G$ . So, this hash  $G$  is just the some of the sizes of the clauses minus the number of clauses. So, this  $n$  is number of clauses. So, now what happens is this measure is 0 if and only if every clause is made of a single literal. And, so now will do induction on this measure hash  $G$  and so one of the theorems we prove before a Herbrand's theorem was. A,  $G$  is unsatisfiable if and only if  $G$  is a complimentary pair and this resolution 0 will resolvent of a complimentary pair will be empty clause.

And, we can go through this induction step and I think we did this. Did we do this? I will not do this again. So, this was essentially a proof that propositional resolution refutation is complete this I mean refutation is what is complete. This word refutation is what important derivation of empty clause so in particular even though you take set of clauses. And, you do a resolution and you get a new set of clauses  $S$  prime even though, this  $S$  prime is a logical consequence of  $S$  there is no guarantee that resolution is complete for pure logical consequence. It is complete so, what this theorem tell us is that propositional resolution is complete for refutation whenever an empty clause can be is to be derived.

So, for pure logical consequence by a direct proof through resolution it is not clear that you can prove everything that you want to prove as logical consequence. What is clear is you want to prove something as a logical consequence of set of clauses you negate it and, add it to the set of classes and then there is a guaranteed derivation of an empty clauses. So, this theorem only shows that resolution is not a useful may not be a very useful thing if you want to do direct proofs it is useful and complete only if you are looking at refutation again. Let, us an important observation which for example I mean there are some trivial examples one can give sort of illustrate this. So, let us look at this propositional resolution take this example.

(Refer Slide Time: 08:41)

**Resolution Examples: Biconditional**

**Example 7.1** Sometimes there may be more than one complementary pair of literals in the same pair of clauses. Consider the biconditional operator ( $\leftrightarrow$ ) on atomic propositions  $p$  and  $q$ . We have

$$p \leftrightarrow q \Leftrightarrow (p \wedge q) \vee (\neg p \wedge \neg q) \Leftrightarrow (p \vee \neg q) \wedge (\neg p \vee q) \equiv \{p, \neg q\}, \{\neg p, q\}$$

Applying resolution on the pair of literals  $p$  and  $\neg p$  we obtain the clause set which after clean-up yields the empty set of clauses.

$$\{q, \neg q\} \equiv \{\} \equiv \top$$

There is really nothing more to this resolvent than that  $\top$  is a logical consequence of any proposition (including  $\perp$ ).

NPTEL

So, take this Biconditional. So,  $p$  biconditional  $q$  so here I am considering only propositions. So, there are so this  $p$  biconditional  $q$  is equivalent to this and comes down to this set of clauses  $p$  comma  $\neg q$   $q$  comma  $\neg p$ . Suppose, now I have a set of containing two clauses and I have one pair of complimentary pairs actually they have two pairs of complimentary pairs. But, I do a resolution on one of them then, I get the other pair as a single clause. And of course, this clause is identically equivalent to the true so all that we have been able to get by a direct proof is just that the true is the logical consequence of this one. I mean that it is not clear that you can get anything else out of this.

(Refer Slide Time: 09:50)


**Resolution Examples: Exclusive-Or**

**Example 7.2** The exclusive-or operation  $\oplus$  is simply the negation of the **biconditional operator**  $\leftrightarrow$ . Hence we have

$$p \oplus q \Leftrightarrow (p \oplus q) \wedge (\neg p \vee \neg q) \equiv \{p, q\}, \{\neg p, \neg q\}$$

which on resolution on the pair  $(p, \neg p)$  and subsequent clean-up again yields the empty set of clauses.

$$\{\}, \{\neg q\} \equiv \{\} \equiv \top$$



So, similarly if you take the negation of this operator which is essentially the exclusive or operator then, also you get true as a logical consequence.

(Refer Slide Time: 10:03)

**Resolution Refutation: 1**

**Example 7.3** Consider the simple logical consequence


$$p \wedge q \models p$$

which we prove by resolution refutation. The set of clauses representing the hypothesis and the negation of the conclusion is  $\{p\}, \{q\}, \{\neg p\}$ . Resolving on the pair  $(p, \neg p)$  yields

$$\{\}, \{q\}$$

Notice that  $\{\} \equiv \perp \equiv \{\}, \{q\}$ .

Since for any clause  $C \neq \emptyset$ ,  $C \supseteq \{\}$ , the clean-up always reduces every set of clauses  $\Delta$  to  $\{\{\}\}$  whenever  $\{\} \in \Delta$ .



However, if you have a goal directed method you want to prove specifically that p and q p is a logically consequence of p and q. Then, what you do is you actually negate this consequence p add it to the set of clauses and then try to derive an empty clause and for that your propositional

resolution is complete. So, which means all proofs by resolution are essentially guaranteed only if they are indirect proofs only if you can prove the derive the empty clause. So, in fact the tabular methods are also very much like that we will come to that. But, let me go back to first order relation. So, essentially the refutation is the only thing for which we can expect completeness. But, then as we know from our original definition refutation is sufficient for proving anything that can be proven by direct proof can also be proven by refutation by contradiction. So, in that sense this is so will only look at refutations.

(Refer Slide Time: 11:30)

**The Lifting Lemma**

**Lemma 31.5 (Lifting Lemma).** (see figure) Let  $C_1$  and  $C_2$  be clauses and let  $\theta_1, \theta_2, \sigma$  be substitutions such that disjointness-of-free-variables

- $FV(C_1) \cap FV(C_2) = \emptyset,$
- $FV(\theta_1 C_1) \cap FV(\theta_2 C_2) = \emptyset$  and
- $C'_{12}$  is the resolvent of  $\theta_1 C_1$  and  $\theta_2 C_2$  via a substitution  $\sigma$ , by a single application of resolution.

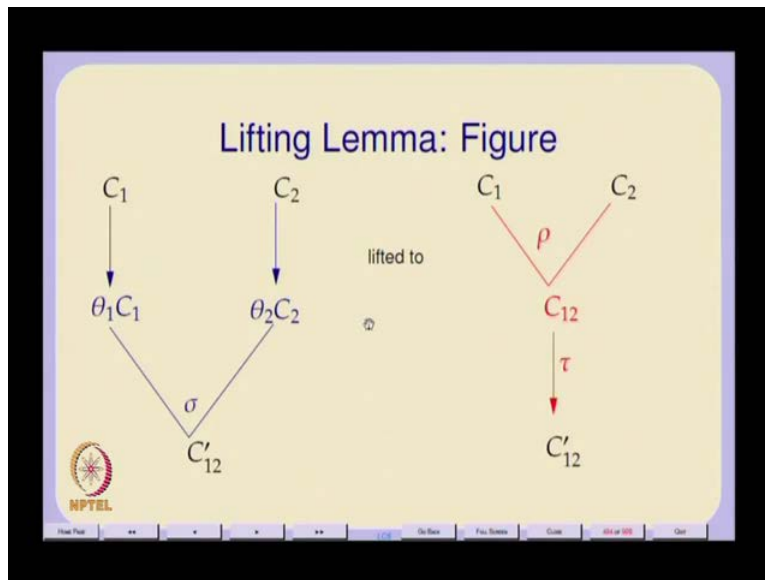
Then there exists a resolvent  $C_{12}$  of  $C_1$  and  $C_2$  by a single application of resolution via a substitution  $\rho$  and a substitution  $\tau$  such that  $C'_{12} \equiv \tau C_{12}$ .

NPTEL

But, now given that propositional resolution refutation is complete what happens to first order refutation that is what we are looking at. Here, is some interesting things so the main difference between propositional case and the first order case is the presence of variables. The fact that the propositions or that your predicates or parameterized on variables. And, those and the fact that you have a sigma algebra of terms which, will be which will act as parameters of the predicates and different terms give you different predicates. So, let us look at this so before I proof the completeness I want this thing known as a Lifting Lemma. So, what does is this is also complicated. You can read it this later.



(Refer Slide Time: 12:27)



The more important thing is what does this Lifting Lemma says is essentially this. Supposing, think I have two clauses let us say  $C_1$  and  $C_2$ . And, there is some and let us assume that these two clauses have no variables in common. That, is so if I have two clauses one of the things that we can do is standardizing variables apart so we can rename all the variables make them all unique. So, there will be a set  $X_1$  of variables occurring in  $C_1$  and a disjoint set  $X_2$  which are the set of variables occurring in  $C_2$ . And, I perform one substitution and I perform two substitutions. So, essentially on  $C_1$  I perform substitution  $\theta_1$  on  $C_2$  I perform a substitution  $\theta_2$ . And, clearly the domains of these two substitutions are going to be disjoint because the variables are disjoint. So, I am substituting disjoint sets of variables with disjoint sets of terms. Since, the  $C_1$  are distinct from the variables of  $C_2$  the terms in the substitution are also going to be disjoint from the terms in this substitution. So, the disjointness carries all the way up to here.

So, that is what is to first is two conditions specify the free variables of  $C_1$  are disjoint from the free variables of  $C_2$ . The free variables of  $\theta_1 C_1$  are also disjoint from the free variables of  $\theta_2 C_2$ . Supposing, think I guarantee these two conditions.

If, I can guarantee these two conditions and then I can find resolvent  $C'_{12}$  through some substitutions through some unifiers  $\sigma$  this  $\sigma$  unifies  $\theta_1 C_1$  and  $\theta_2 C_2$ .

Basically, what we are saying is you find some atom  $p$  you find complimentary pairs  $p$  with some terms of  $x_1$  involving variables of  $x_1$  occurring in  $\theta_1 C_1$  naught  $p$  with let us say with variables of  $x_2$  occurring in  $\theta_2 C_2$ . And, you are able to do a resolution a single step of resolution and you will obtain let us say at a clause  $C'_{12}$ . So, basically what we are saying is all the other the rest of  $C_1$  which was not involved in the resolution in rest of  $C_2$  you will do the substitutions given in  $\sigma$ . Now, since these two are disjoint in their sets of variables what this lemma just says is that. It is possible for me to find a unifier row. Which, will resolve these two clauses without doing the substitutions. It will give me some  $C_{12}$  and it is possible for me to get the  $C'_{12}$  through another substitution term.

So, what we are saying is having done these substitutions and then realizing that there is a there are complimentary pairs and so the resolution step possible is equivalent to finding a resolvent here. And, in order to get this I find an appropriate substitution term. So, what you are doing is you are lifting this resolution step to the one with variables. Note that  $C_1$  and  $C_2$  are variables in that I mean where as in particular supposing, here I have not made any assumptions about ground terms. But, in particular we have to relate it to the completeness of resolution for ground terms which is what we had proven before. We are going to use that in order to prove completeness. So, in particular  $\theta_1$  and  $\theta_2$  are ground substitutions. Then, this  $\sigma$  is essentially an identity substitution I mean because you are essentially doing propositional resolution.

So, this is equivalent therefore this can be lifted to doing a resolution on terms of variables and then doing a ground substitution term to get (Refer Time: 17:27). So, what this if  $\theta_1$  and  $\theta_2$  are ground such that  $\theta_1 C_1$  and  $\theta_2 C_2$  are variable free then you can do only propositional resolution. But, whenever that is possible what you are also saying is it is possible to do a predicate logic resolution a first order resolution. And, get the same effect by doing an appropriate ground substitution term. So, this is what the theorem says. This is what this lemma says. And, the proof of this lemma I think I am going to leave it to you to study on your own it is an important lemma. You have to use the fact that these have disjoint sets of variables these also have disjoint sets of variables.

(Refer Slide Time: 18:35)

and

$$C'_{12} = \tau(\rho(C'_1 \cup C'_2)) = (\sigma \circ (\theta_1 \cup \theta_2))(C'_1 \cup C'_2)$$

A superposed version of the figure is shown below.

The diagram illustrates the lifting lemma. It shows two clauses,  $C_1$  and  $C_2$ , at the top. Red arrows labeled  $\rho_{12}$  point from  $C_1$  and  $C_2$  to a central clause  $C_{12}$ . Blue arrows labeled  $\theta_1$  and  $\theta_2$  point from  $C_1$  and  $C_2$  to  $\theta_1 C_1$  and  $\theta_2 C_2$  respectively. A blue arrow labeled  $\sigma_{12}$  points from  $\theta_1 C_1$  and  $\theta_2 C_2$  to a bottom clause  $C'_{12}$ . A red arrow labeled  $\tau_{12}$  points from  $C_{12}$  to  $C'_{12}$ . The NPTEL logo is visible in the bottom left corner of the slide.

So, it is a fairly general purpose lemma but we are going to use it only when the case when theta 1 and theta 2 are ground substitutions that is enough for us to prove completeness of resolution refutation. So, I can superposed those two figures these two figures can be superposed to yield something like this yield a figure like this. So, a figure like this will represents a lifting lemma basically.

(Refer Slide Time: 19:04)

## Completeness of Resolution Refutation: 1

1. The **lifting lemma** helps us to use the **completeness of resolution refutation for ground clauses** and "lift" it to clauses with variables.
2. By **standardizing variables apart** we may guarantee that the conditions of **disjointness of free variables between different clauses** may be enforced.
3. Any set of clauses  $S = \{C_i \mid 1 \leq i \leq m\}$  represents the **conjunction of the universal closure of each clause**.

The NPTEL logo is visible in the bottom left corner of the slide.

So, now the essentially the lifting lemma helps us to use the completeness of resolution refutation for ground clauses. And, lift it to clauses with variables in that and by standardizing variables apart we can guarantee those two conditions of lifting lemma are satisfied. And, of course any set of clauses represents a consumptions the universal closure of each clause. One of the things we did was.

(Refer Slide Time: 19:32)

**Facts about Clauses**

**Lemma 29.4** Let  $\{C_i \mid 1 \leq i \leq m\}$  be a set of clauses. Then

$$\forall \left[ \bigwedge_{1 \leq i \leq m} C_i \right] \Leftrightarrow \bigwedge_{1 \leq i \leq m} \forall C_i$$

*Proof:* Follows from the semantics of  $\forall$  and  $\wedge$  or alternatively from corollary 25.2. ■

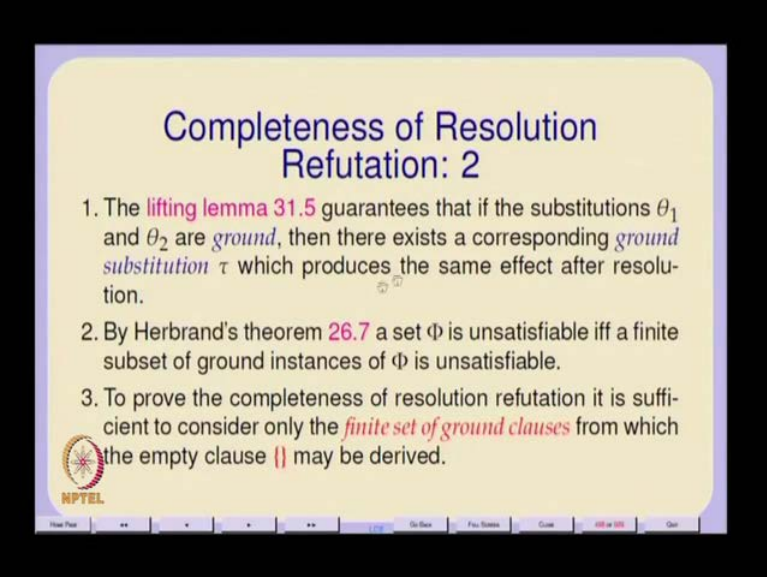
Notice that even if there are free variables common between two clauses, this lemma holds, mainly because of the fact that there are no existential quantifiers. For example

$$\forall x [p(x) \wedge q(x)] \Leftrightarrow \forall x [p(x)] \wedge \forall y [q(y)] \Leftrightarrow \forall x, y [p(x) \wedge q(y)]$$

HPTTEL

We showed that you know actually a set of clauses that represents a universal closure of an and of all the clauses. But, that is equivalent to taking the universal closure of each clause separately so that will have a smaller number of variables. And, taking a (Refer Time: 19:48) in that so we will use basically in this.

(Refer Slide Time: 19:56)



The slide is titled "Completeness of Resolution Refutation: 2" and contains three numbered points. The first point discusses the lifting lemma 31.5, the second discusses Herbrand's theorem 26.7, and the third discusses the completeness of resolution refutation. The slide also features the NPTEL logo in the bottom left corner and a navigation bar at the bottom.

### Completeness of Resolution Refutation: 2

1. The **lifting lemma 31.5** guarantees that if the substitutions  $\theta_1$  and  $\theta_2$  are *ground*, then there exists a corresponding *ground substitution*  $\tau$  which produces the same effect after resolution.
2. By Herbrand's theorem **26.7** a set  $\Phi$  is unsatisfiable iff a finite subset of ground instances of  $\Phi$  is unsatisfiable.
3. To prove the completeness of resolution refutation it is sufficient to consider only the *finite set of ground clauses* from which the empty clause  $\square$  may be derived.

NPTEL

So, the lifting lemma guarantees so that  $\theta_1$  and  $\theta_2$  are ground then there is there are there exists corresponding ground substitution  $\tau$  which will produce the same effect as the lifting lemma. We have Herbrand's theorem which shows that a set  $\Phi$  is unsatisfiable if and only if there is a finite sub sets of ground instances of  $\Phi$  which is unsatisfiable. So, what we do is to prove completeness of resolution refutation we just consider only a finite set of ground clauses, from which the empty clause may be derived. So, assume that let us go out of the theorem is just that. Suppose,  $\Phi$  is the set of clauses which is unsatisfiable there is no model which means then what we are saying is the empty clause can be derived in first order resolution. And, how we are going to do it? We are going to use Herbrand's theorem.


(Refer Slide Time: 21:16)

**Proof of theorem 31.6**

*Proof:* Without loss of generality we may assume that the variables in every clause are disjoint from the variables occurring in any other clause.  $\odot$

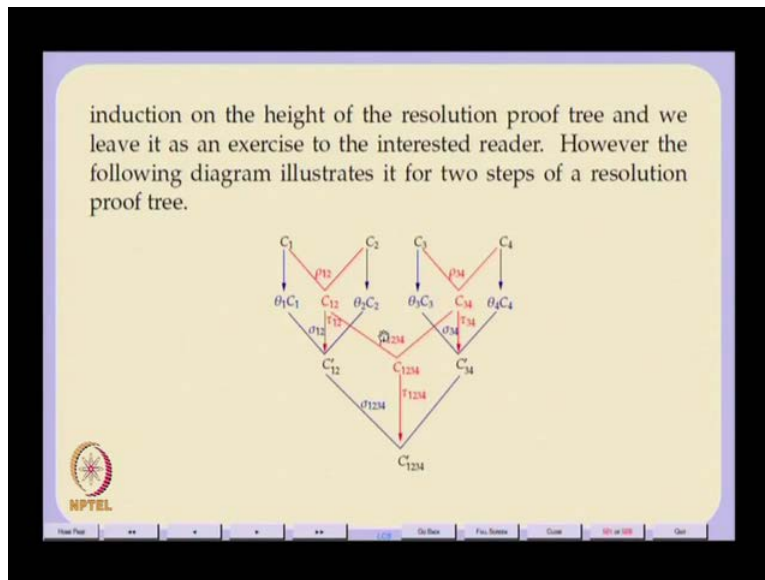
By Herbrand's theorem 26.7 there exists a finite set of ground clauses  $G = \{gC_i \mid 1 \leq i \leq m\} \subseteq_f \mathfrak{g}(\Phi)$  such that  $G$  is unsatisfiable.  $gC_i = \theta_i C_i$  for each  $gC_i \in G$ . Further for  $i \neq j$ ,  $dom(\theta_i) \cap dom(\theta_j) = \emptyset$  and since all the clauses in  $G$  are ground the disjointness conditions of the lifting lemma are trivially satisfied.

Each application of rule Res0 in  $G$  may be lifted to finding a resolvent of the appropriate clauses. This fact may be proved by

 NPTEL

And, essentially what we are saying is that first of all we are going to standardize the variables apart. And, by Herbrand's theorem basically what we are saying is that you can finite may be a finite or infinite set I do not care. Whatever, it is that ground instances of all those clauses is going to be some infinite sets. But, if  $\Phi$  is unsatisfiable then by Herbrand's theorem there is a finite subsets of ground clauses which are also unsatisfiable. Now, if that finite set of ground clauses is unsatisfiable we know from the completeness of resolution of ground clauses then the empty clause can be derived from them ground clauses. Now, all that we need to do is take that finite set of ground clauses I am calling them  $gC_i$ . Each of them have a parent in the original set  $\Phi$  through the ground substitution  $\theta_i$ . So, take only those clauses  $C_i$  and now apply the lifting lemma. So, what you are saying is only those that subsets of clauses so there is a finite subset  $G$ . Which, is refutation which derives an empty clause completely ground you take that parent clauses all the  $C_i$ 's. And, the lifting lemma guarantees that I can find a most general unifier and a resolvent. And, I am and the lifting lemma guarantees that all that provided that satisfies disjointness conditions which I can fulfill by standardizing variables apart.

(Refer Slide Time: 23:11)

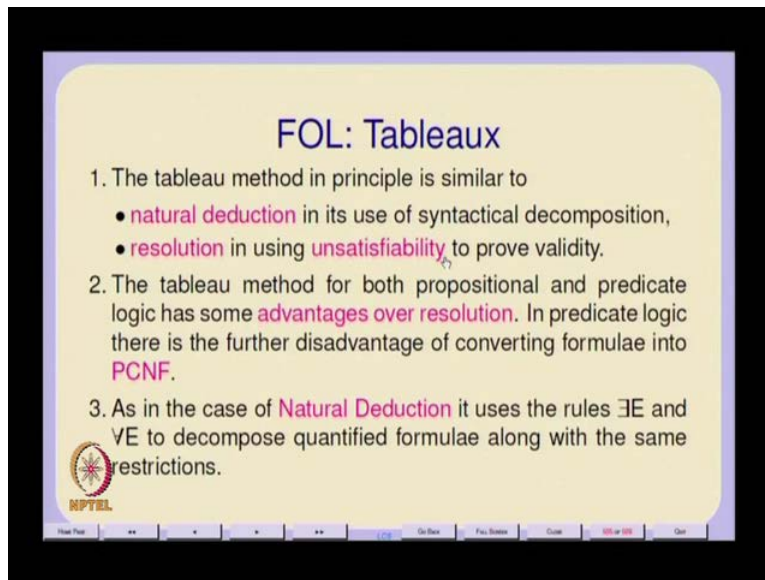


And, once we do that essentially we can by induction on the number of steps of ground resolution. For, each step of ground resolution you can apply the lifting lemma and I am not going to go through the details of that. But, here is the simple illustration where, there are two steps of resolution so the blue ones are all the ground ones. So, let us assume that  $C_{1234}$  is the empty clause. And, it is derived from these 4 clauses  $C_1$ ,  $C_2$ ,  $C_3$ ,  $C_4$ . You take let  $\theta_1$ ,  $\theta_2$ ,  $\theta_3$ ,  $\theta_4$  be ground instances of all these clauses. Then, what you are saying is that there exists some resolvent  $\sigma_{12}$  between  $C_1$  and  $C_2$  here there exists another resolvent  $\sigma_{34}$  between these two. That will give you  $C_{12}$  and  $C_{34}$  then, there exists a  $\sigma_{1234}$ . Which, gives you this  $C_{1234}$  let us say is a empty clause.

Now, the lifting lemma in each case guarantees these red lines. So, it says that it is possible to find the resolvent  $C_{12}$  without doing any substitution. Such that if, I want  $C_{12}$  there is a ground substitution  $\tau_{12}$  give me that. And, similarly in this case and this so look at this  $C_1$  to  $C_3$ ,  $C_4$   $C_{12}$  to  $C_{34}$ ,  $C_{12}$  to  $C_{34}$  they are also satisfy condition of the listing lifting lemma I can standardize a variable apart. And, so I can apply the lifting lemma again and that guarantees that I can find unifier a most general unifier  $\tau_{1234}$ . And, a ground substitution  $\tau_{1234}$  which should be my empty clause.


So, from propositional resolution we can actually derive first order logic resolution but only for refutations. So, the completeness is only for refutation though it is clear that all the clauses that you get are actually logical consequences of original sets of clauses. So, that means resolution is first order logic.

(Refer Slide Time: 25:48)



### FOL: Tableaux

1. The tableau method in principle is similar to
  - **natural deduction** in its use of syntactical decomposition,
  - **resolution** in using **unsatisfiability** to prove validity.
2. The tableau method for both propositional and predicate logic has some **advantages over resolution**. In predicate logic there is the further disadvantage of converting formulae into **PCNF**.
3. As in the case of **Natural Deduction** it uses the rules  $\exists E$  and  $\forall E$  to decompose quantified formulae along with the same restrictions.

 MPTCL

Slide Time: 25:48

Now, the some of the other things that we have to do is first off all we have to look at tabular methods and we have to look at the completeness of Hilbert's style proof system. I think those are two important things that we want to do. We did propositional tabular what about first order logic tabular. In tabular are also similar to resolution in the sense that again you are trying to prove the unsatisfiability. So, what happens a tabular is a tree we have these various tabular rules.



(Refer Slide Time: 26:28)

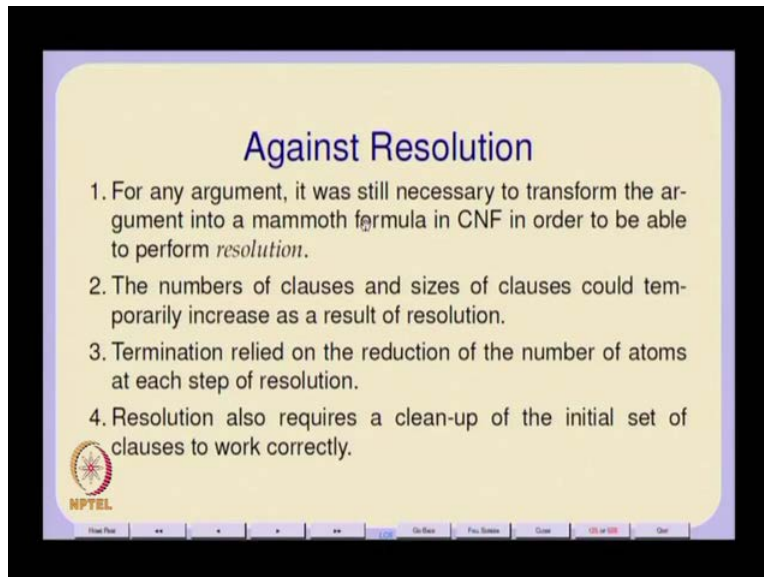
Tableaux Rules	
	$\neg\neg. \frac{\neg\neg\phi}{\phi}$
$\wedge. \frac{\phi \wedge \psi}{\phi \quad \psi}$	$\neg\wedge. \frac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi}$
$\vee. \frac{\phi \vee \psi}{\phi \mid \psi}$	$\neg\vee. \frac{\neg(\phi \vee \psi)}{\neg\phi \quad \neg\psi}$
$\rightarrow. \frac{\phi \rightarrow \psi}{\neg\phi \mid \psi}$	$\neg\rightarrow. \frac{\neg(\phi \rightarrow \psi)}{\phi \quad \neg\psi}$
$\leftrightarrow. \frac{\phi \leftrightarrow \psi}{\phi \wedge \psi \mid \neg\phi \wedge \neg\psi}$	$\neg\leftrightarrow. \frac{\neg(\phi \leftrightarrow \psi)}{\phi \wedge \neg\psi \mid \neg\phi \wedge \psi}$

Which, essentially gives you a branching with elongation and branching rules. So, this is an example of a branching rule and this is another example of a branching rule then these two are also branching rules. This is an example of an elongation rule and so is this. And, what we did was. We started with the set of clauses get on elongating as much as possible. And then, if necessary to be so we broke down the clauses based on their syntax on this structure of this syntax tree of the are not clauses of the formula. So, the important thing here is that in general so the tableau method in principle because it works on the root of the root operator of the formula is bit like natural deduction. The natural deduction had introduction and elimination rules the tableau just allows you to break up break the formula according to the decompose the formula in terms of the root. So, it applies on the root operator and so in that sense it is similar to natural deduction.

But, on the other hand it is similar to resolution because basically unsatisfiability is what you prove. So, you are looking for a closed tableau. A closed tableau is one is a tree is a tableau tree in which all the paths are closed. And, a path is closed if you if along the path you can get a complimentary pair. So, if all paths have complimentary pairs then all paths are closed and the tableau cannot be extended any further basically. So, as long as there is no complimentary pair the conjunction of all the formulas in a path is satisfiable is assumed satisfiable. So, it is similar to resolution in the sense that you are going to look for finite tableaus which are closed and the

tableau is closed if all the paths are closed. If, even a single path is open then the tableau what you have got is an assignment satisfying assignment.

(Refer Slide Time: 29:02)



**Against Resolution**

1. For any argument, it was still necessary to transform the argument into a mammoth formula in CNF in order to be able to perform *resolution*.
2. The numbers of clauses and sizes of clauses could temporarily increase as a result of resolution.
3. Termination relied on the reduction of the number of atoms at each step of resolution.
4. Resolution also requires a clean-up of the initial set of clauses to work correctly.

NPTEL

So, one of the advantage is that the tableau had and continues to have for first order logic is really that resolution required a huge amount of initial paperwork to be done. You had to take these formulas convert them into pre next normal form. Then, take the formula take the body of the pre next normal form convert it into conjunctive normal form. And, that conjunctive normal form might often involve distribution of or over and. Which, can expand the body and in that sense there are certain natural disadvantages of resolution which are not hidden when we look at resolution only in terms of sets of clauses. So, if you are given only sets of clauses then that is fine resolution is fine. But, otherwise there is you basically create a through pre processing you create a mammoth formula this is something that the tableau does not require. Because, the tableau does not require any of these transformations it just naturally breaks up formulas according to their operators.

So, it has these advantages over resolution and it is since it works on the syntax of formulas on the root operator it is almost like natural deduction. Natural deduction has introduction and elimination rules for each operator. Where, as tableau keeps on breaking up operators it just keeps on breaking up the formula based on the root operator. So, it has essentially only so it uses

essentially only the elimination rules of natural reduction. But, it uses a branching which a branching tree and so otherwise it is a lot like natural deduction and it is a lot like resolution. It is a it is like natural reduction because it is totally syntactic. And, it decomposes formulae's into some in terms of their sub formulae. It is like resolution in the sense that you looking for a closed tableau you are looking for ways to close all the paths.

(Refer Slide Time: 31:16)

**FOL: Tableaux Rules**

Besides the usual **tableaux** rules for the propositional connectives we have the following.

$\forall. \frac{\forall x[\phi]}{\{t/x\}\phi}$	$\neg\forall. \frac{\neg\forall x[\phi]}{\neg\{a/x\}\phi}$
$\exists. \frac{\exists x[\phi]}{\{a/x\}\phi}$	$\neg\exists. \frac{\neg\exists x[\phi]}{\neg\{t/x\}\phi}$

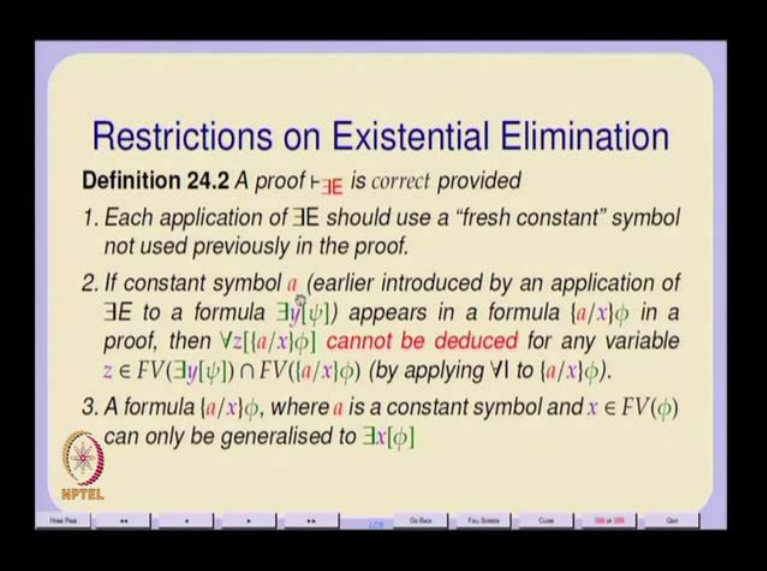
1. The restrictions on the use of a constant symbol  $a$  in both rules  $\neg\forall.$  and  $\exists.$  are the same as those for  $\exists E$  in both  $\mathcal{H}_1$  and  $\mathcal{G}_1$ .
2. Since  $\forall E$  holds for all terms  $t$ , the rules  $\forall.$  and  $\neg\exists.$  may have to be applied several times before unsatisfiability can be proven.

MPTEL

So, that tableau rules therefore it is they are not surprising at all. We want to look for ways of breaking up the each quantified formula and you want to negations of quantified formula that is how the tableau works. So, we have two rules the equation so there are two rules so and these two rules there are these four rules. And, these four rules are essentially like your universal quantifier elimination and existential quantifier elimination. Because, by Demorgan's law the negations are also duals they dualise. So, you take this so you have for all x phi and basically what you are saying is that for all terms t, tx phi is something that should follow from for all x phi. In the case of existential quantifier essentially the same restrictions hold as for existential quantifier elimination in the Hilbert's style systems. So, there was a notion of a scope adding a creating a new and fresh constant so this A here is very much in that spirit. So, this A has to be a fresh constant. Which, does not occur anywhere in the path in that path in which you are doing

this existential quantifier elimination. So, it should not have occurred before so this has to be fresh.

(Refer Slide Time: 33:20)



**Restrictions on Existential Elimination**

**Definition 24.2** A proof  $\vdash \exists E$  is correct provided

1. Each application of  $\exists E$  should use a "fresh constant" symbol not used previously in the proof.
2. If constant symbol  $a$  (earlier introduced by an application of  $\exists E$  to a formula  $\exists y[\psi]$ ) appears in a formula  $\{a/x\}\phi$  in a proof, then  $\forall z[\{a/x\}\phi]$  **cannot be deduced** for any variable  $z \in FV(\exists y[\psi]) \cap FV(\{a/x\}\phi)$  (by applying  $\forall I$  to  $\{a/x\}\phi$ ).
3. A formula  $\{a/x\}\phi$ , where  $a$  is a constant symbol and  $x \in FV(\phi)$  can only be generalised to  $\exists x[\phi]$

NPTEL

So, I mean this is you remember these are these restrictions on existential elimination which we had for the Hilbert's style system. So, in each this constant symbol  $A$  has to be absolutely fresh in net scope in the case of the tableau. the, scope that we are talking about is the path which is open at the moment which you are extending with something more. So, no parent along that path no ancestor along that path should have had an occurrence of  $A$ . If, you are applying the root the negation of the universal quantifier is very similar to the existential quantifier. So, the notion of a fresh  $A$  happens here to I mean it is after all a dual and the notion and this of course is models the same as the universal quantifier. In particular when you looking at it as the universal quantifier this is for all terms  $t$ . For every term  $t$  this actually holds what in the case when we did the Hilbert's style proof system.

(Refer Slide Time: 34:33)

**Proof Rules: Hilbert-Style**

**Definition 22.1**  $\mathcal{H}_1(\Sigma)$ , the Hilbert-style proof system for Predicate logic consists of

- The set  $\mathcal{L}_1(\Sigma)$  generated from  $A$  and  $\{\neg, \rightarrow, \forall\}$
- The three logical axiom schemas **K**, **S** and **N**,
- The two axiom schemas

$\forall E$ .  $\frac{}{\forall x[X] \rightarrow \{t/x\}X}$ ,  $\{t/x\}$  admissible in  $X$

$\forall D$ .  $\frac{}{\forall x[X \rightarrow Y] \rightarrow (X \leftrightarrow \forall x[Y])}$ ,  $x \notin FV(X)$

- The *modus ponens (MP)* rule and

$\forall I$ .  $\frac{\{y/x\}X}{\forall x[X]}$ ,  $y \notin FV(X)$

It did not matter to us really notice that this is essentially for all terms  $t$  that are admissible in  $x$ . Where, as if you are looking at refutation which is what your tableau does for closure then if the tableau path closes for one particular choice of their term  $t$  it might close it may not remain closed for all possible terms  $t$ . Because, there is a combination of there might be combination of existentially quantified formulas and universally quantified formulas. Because, of which you might have to reuse this root we are not doing direct rules we are doing a refutations actually. So, let us look at an example so, this is so what I am trying to say is that this for all and not there exists may have to be applied several times before unsatisfiability can be proven.

(Refer Slide Time: 35:55)

**FOL Tableaux: Example**

**Example 32.1** The set  $\Phi = \{\neg p(c), p(f(f(c))), \forall x[p(x) \vee \neg p(f(x))]\}$  (where  $c$  is a constant symbol and  $f$  is a unary function symbol) is unsatisfiable using the tableau rules.

$\neg p(c)$	$p(f(f(c)))$	$\forall x[p(x) \vee \neg p(f(x))]$
$p(c)$	$\neg p(f(c))$	$p(f(c) \vee \neg p(f(f(c))))$
$p(f(c))$	$\neg p(f(f(c)))$	$p(f(c)) \vee \neg p(f(f(c)))$
$\neg p(f(c))$	$p(f(c))$	$\neg p(f(f(c)))$

Notice the **two applications** of rule  $\forall E$ . ■ indicates a closed path in the tableau.

And, here is an example which illustrates that take this set, naught of C is a constant let us assume that C is constant in the signature that is why I have not written it in red I have written it in violet I mean let us assume C is part of sigma. So, I have naught p of c and f is unary function symbol I have p of f of f c. And, then I have for all x p x or naught p f x and now this set is unsatisfiable. And you, can prove it by some other means through unification through resolution whatever but let us look at a tableau proof and what it involves. So, we just use elongation initially so we first list out all the so this is tree by the way I know it does not look much of tree but it is a tree. So, here this horizontal line is where the branching takes place so this p of c or let us take this for all x p of x or naught p f of x I choose instead of I choose to instantiate this universally quantified formula with c for x.

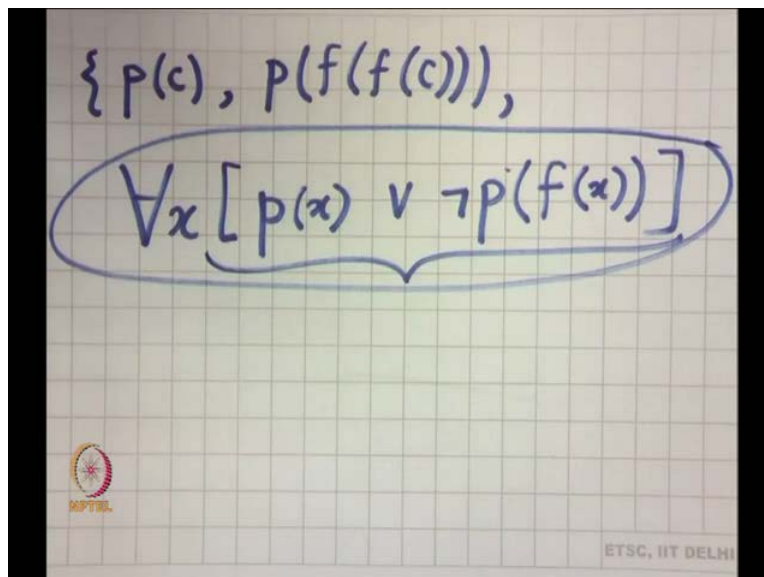
And, if I choose to do that then I get p of c or naught p of f c. So, this or is requires a branching rule so you assume p of c and you assume naught p of f c. But, p of c of course contradicts this naught p of c on the path and so this path is closed that is what this red square is supposing to indicate that the path is closed. So, now you take this naught p of f c this path is open. Now, I do another instantiation of this for all and this time I instantiate it instant replace x with f of c. So, then I get p of f of c or naught p of f of f of c. So, p off c and naught of p of f c here, a contradictory this or implies a branching again. And, this p of f c contradicts this not p of f c. So,

this path closes and this naught p of f f of c contradicts this p f f of c and so this path is also closed.

So, there is a difference here that in a propositional tableau method we actually scored out any formula that been used all those all those formula were well essentially disposable if, you like. Once you decompose the formula you did not require the original formula but here we see the first important reference. That, is a universal formula cannot be discarded after instantiation it has to be reused all other formulae can be discarded once they have served their purpose. Once they have been decomposed. So, for example once this branching has taken place this p of c or naught p of f c is useless and, can be scored out. But, and similarly this p of c or naught p of f f c can be discarded once I have done this branching.

Because, they are essentially proposition but all you know all quantified formulae will have to be reused. Actually, what happens is that it is only for all x and naught of there exists x which are both only universal quantified formulae may have to be reused before you can prove that the entire tableau is closed. If, this supposing instead of naught p of c I had just p of c. Then what would happen is this, universal quantifier would essentially give you all I mean there would be an infinite path in the tableau would not close.

(Refer Slide Time: 41:14)



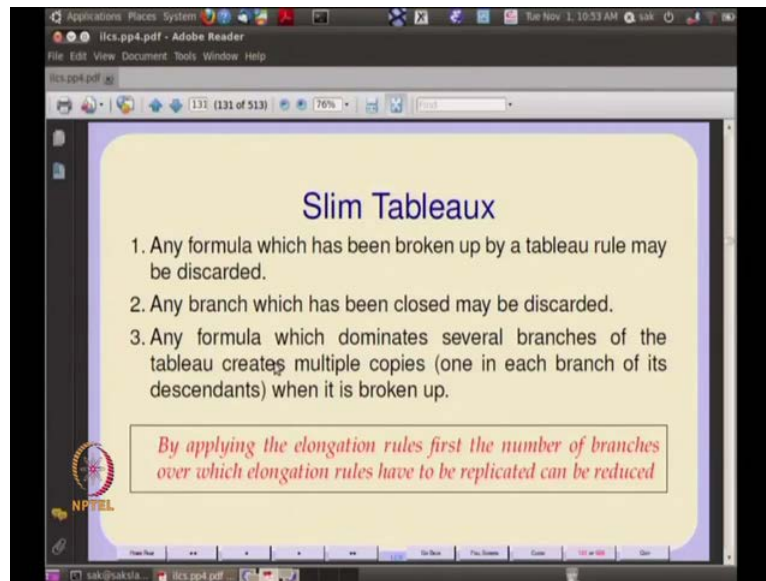
So, if we were to take if I were to take this set  $p$  of  $c$  well  $p$  of  $f$  of  $f$  of  $c$  and for all  $x$   $p$  of  $x$  or naught  $p$  or is this  $f$  or then some parts of the tableau will not close. But, if you are looking at a complete tableau then, in that complete tableau you will essentially have to have all instances all ground instances of this formula. That, is like taking the entire Herbrand universe and that is like elongating with an infinite number of formulae because afterwards for all  $x$  it is like an infinite tree. And, your if there is a constant if there is at least one constant then your Herbrand universe the set of all ground terms that are possible is an infinite set. So, this essentially encodes an infinite possible path in the tableau.

So, now here so that is so the only tableau's under quantification under universal quantification or equivalently negation of existential quantification. Though, only finite tableau's are those which are closed all the other tableau's. So, for any set of any set  $\phi$  so, in the propositional case because of the fact that we did pure decomposition. And, your formulas were essentially like toilet paper use once and throwaway. So, you had only finite tableau's and those finite tableau's could have for finite set of formula you had only a finite tableau and it was always finitely branching of course. And, so you had only a finite number of paths some of those paths may have been closed the other paths would have been opened.

But, in the case of universal formulae you have the possibility of actually having an infinite path. And, this universal form universal formula in an open in a tableau which is in which all paths are not closed essentially encodes an infinite tree path for, all possible instances of that you can have for  $x$ . This, I do not have any more slides but do some example. There is one quick example I can do one of the things that we did in the if you remember I do not know how well you remember but there is a tendency to understanding to also the volatile. But, there is something called something called slim tableau.




(Refer Slide Time: 45:04)



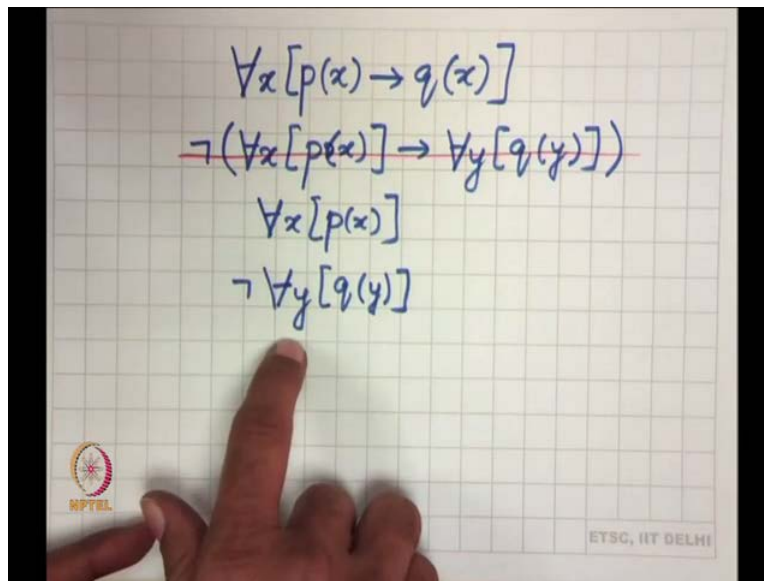
So, there is there is the notion of I mean a heuristic you normally apply is that you postpone the application of branching rules as far as possible. And, use elongation give priorities to elongation rules. So, you first do all the elongation rules possible and when the tableaux got only branching rules possible that is when you do branching so, you get Slim Tableaux because of that. In this particular case this is essentially a huge elongation if you like so it encodes a huge elongation that is not a serious issue. But, this huge elongation comes in the way of finding closure I mean after all in what order are going to this in a put in all the do all the instantiation this is the question. If, you notice here there was some guidance which there was guidance which this constant gave. So, if you were to take a proof like this so let us take a refutation proof because I want only a finite tableau.

(Refer Slide Time: 46:22)

$$\forall x [p(x) \rightarrow q(x)]$$
$$\neg (\forall x [p(x)] \rightarrow \forall y [q(y)])$$
  
$$\forall x [p(x) \rightarrow q(x)] \models$$

$$\forall x [p(x)] \rightarrow \forall y [q(y)]$$

So, let us take this simple thing for all  $x$   $p$  of  $x$  no function symbols two atomic predicate symbols unary one logical consequence of this is for all  $x$   $p$  of  $x$ . And, I am going to do an alpha renaming just to keep all the variables separate I mean it does not really matter actually it does not matter at all but, anyway let us do this. So, if you had to prove this so clearly tableau proof for this would start with taking the set. This, set the set is going to be before all  $x$   $p$   $x$  arrow  $q$   $x$  and then the negation of this. So, that means not of for all  $x$   $p$   $x$  arrow for all  $y$   $q$   $y$  so the negation rule essentially this is negation of arrow I mean that. So, there are those propositional tableau rules.

(Refer Slide Time: 48:26)



So, you take negation of arrow so it elongates it so it elongates it to for all x p of x and naught of for all y q of y so this thing is essentially propositional. So, essentially this can be discarded because we have now applied this elongation rule. Now, you have a question of how to proceed with this tableau. And, you the choice of instantiating this universal quantifier this universal quantifier or this essentially existential quantifier. These two universal quantifiers provide more guidance at all. So, if you want to slip and finite tableau you have to first instantiate this so if you instantiate this then what you get is essentially a fresh constant a, you going to use a fresh constant a.

(Refer Slide Time: 49:56)

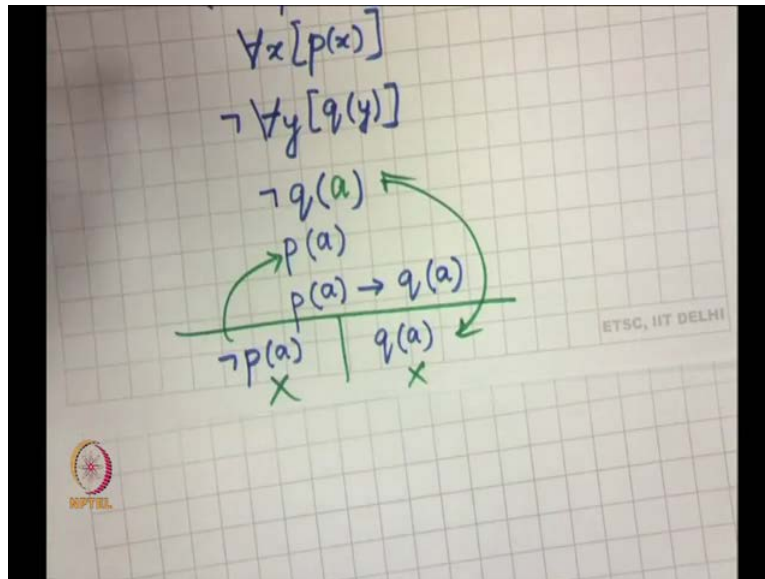
$$\begin{aligned} & \forall x [p(x) \rightarrow q(x)] \\ & \neg (\forall x [p(x)] \rightarrow \forall y [q(y)]) \\ & \forall x [p(x)] \\ & \neg \forall y [q(y)] \\ & \neg q(a) \\ & p(a) \\ & p(a) \rightarrow q(a) \end{aligned}$$

NPTEL

ETSC, IIT DELHI

And, I am going to write it simply as naught q of this is this also I will use a green for the a. So, this is a new fresh constant and this is what I would get. But, a of course you can think of this as being disposable essentially it is the equivalent. So, the ones which are not disposable that the universally quantified once and, therefore the negation of existential quantifiers. So, once you do this a gives you an guidance as so how you going to initialize these. And, so essentially what you are going to do is you going to get p of a here as one instantiation. And, then you going to get p of a q of a arrow p of a q of a. And, then and then you going to go through this now it is becomes propositional this will branch into essentially.

(Refer Slide Time: 51:34)



So, this will branch essentially into not  $p$   $a$  here and  $q$   $a$  here. So, now what happens is so this because of this and because of this both the branches close. And, so you have you have closed tableau so this one so in addition to the fact that you postpone branching as much as possible. This, it is not just slim tableau it also ensures finiteness if it is indeed unsatisfiable they the instantiation of existential quantifier. And, therefore also of naught of the negation of the universal quantifier should be done first to provide an appropriate guidance. That, is to what should be instantiation of the universal quantifier. So, it is I could have of course for example I could have had here formula like  $f$  of  $y$  or  $g$  of  $y$  for instance. Supposing, I had  $f$  of  $y$  or  $g$  of  $y$  somewhere something like that then this would give me naught  $q$  of  $f$  of  $a$ .

But, that the instantiation I would have to do on this there would be  $f$  of  $a$  for  $x$  rather than for  $a$  for  $x$ . So, it gives that guidance that guidance has to be program which is notice that, we do not have unification. So, if you use the existential quantifier to give you the appropriate guidance you can just take that terms from the existential quantifier instantiation and use them as substitution. So, universal quantifier and then do a propositional tableau and so you get not only a slim tableau. You, also get a not a guarantee but possibility of a closing the tableau early so that is far as tableau is concerned.