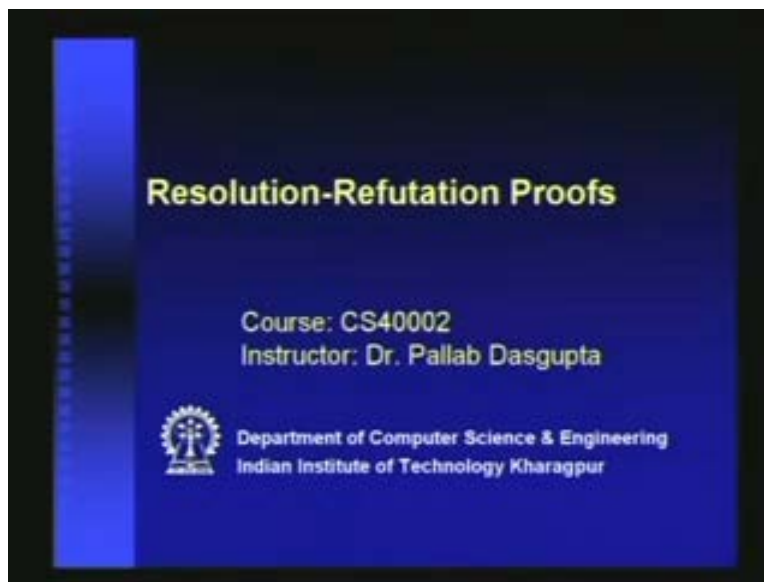


Artificial Intelligence
Prof. P. Dasgupta
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture- 11
Resolution – Refutation Proofs

Let us continue with our study of proof mechanisms in first order logic. So far, what we have seen is proof by using generalized modus ponens, which was different from propositional modus ponens, in the sense that we have to unify the left hand side of the rules, in order to derive the right hand side of the rules.

(Refer Slide Time: 00:01:04)



So, we are going to study resolution refutation proofs, which is a complete proof procedure for first order logic. But first, we must understand that what is the completeness issue for modus ponens. For that, let us look at this example.

(Refer Slide Time: 00:01:25)

Modus Ponens - completeness

- Reasoning with Modus Ponens is incomplete
- Consider the example –

$$\begin{array}{ll} \forall x P(x) \Rightarrow Q(x) & \forall x \neg P(x) \Rightarrow R(x) \\ \forall x Q(x) \Rightarrow S(x) & \forall x R(x) \Rightarrow S(x) \end{array}$$

- We should be able to conclude $S(A)$
- The problem is that $\forall x \neg P(x) \Rightarrow R(x)$ cannot be converted to Horn form, and thus cannot be used by Modus Ponens

CSE, IIT Kharagpur

We have for all x , px implies qx , right; we also have for all x , not px implies rx , right. Then, we have for all x , qx implies sx , and for all x , rx implies sx . Now, we should be able to conclude sa from this. Why? Because it is either pa or not pa , right? It is either pa or not pa . Therefore, it is qa or ra ; if it is qa , then also, we have sa . If it is ra , then also we have sa . We should be able to deduce sa from this, but we cannot do it by generalized modus ponens, because modus ponens will not observe that this and this- the left hand sides together- cover the entire domain of the predicate, because it is either px or not px . That this and this together gives us true or of this, is not observed by generalized modus ponens. And the problem is that this kind of thing that when you when you have not px implies rx , cannot be converted to horn form, because we have negation on the left hand side.

What is the problem of having negation on the left hand side? Now, recall that the basic way in which we were attempting to do the proofs was, start with what we have in the knowledge base; from that, we try to deduce newer and new things. Whatever you do not have at this moment in the knowledge base, you cannot say that it is false. You cannot say that it is false, because so far, you may not have it in the knowledge base, but through the deduction procedure, it may come into the knowledge base; you may be able to deduce that into the knowledge base. Therefore, apriori, you can never say that not of px or not of pa is true, because at this point, pa may not be there in the knowledge base, but through reduction, we may be able to find pa and put it in the knowledge base.

You could say not pa when, after your entire forward chaining is over, you have deduced whatever you could, and in spite of that, you were not able to deduce pa . Then, you can say that ok, I am not able to infer pa from what knowledge I have, right, but deduction of not pa was not possible there. Therefore, if you have not of some predicate on left hand side, then the mechanism of unification that we talked about is not going to work, because we will never have a predicate; we will never have a state of the knowledge base,

where it contains not of something. This is the problem that we have in modus ponens. Now, is it clear what I mean to say? In the knowledge base, what we are maintaining is- we are maintaining the set of facts that we have been able to deduce.

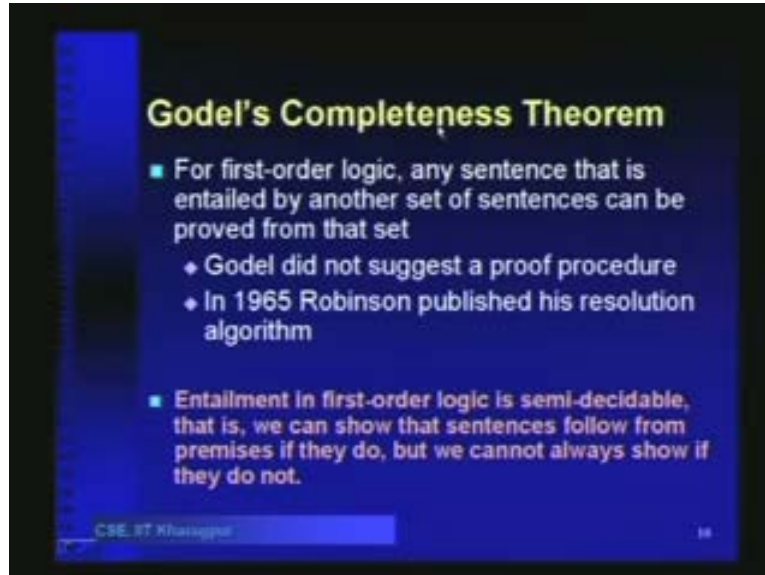
If we have been able to deduce pa , we will put it in the knowledge base, but how do we deduce not of something? We do not have a mechanism of storing the not of something in the knowledge base. What we find is, whatever truths that we have in the knowledge base, we use that to deduce new things. Now, here, we have this rule, which says not of px implies rx , so when can we apply this rule when the knowledge base gives us not of px ? But, if we go by the modus ponens kind of inferencing, then the knowledge base will never tell us not of px , because if pa is not there in the knowledge base, now, it is not able to say that not pa is true, because pa could be coming into the knowledge base as a result of the deduction.

Being able to infer not of pa and then use this rule not of px implies rx , will not work, because the only way of doing reasoning there, was to take the left hand sides of the rules and try to unify them with what we have in the knowledge base, and then deduce the right hand side from that. And we did not have negation here. The moment we have negation on the left hand side, this procedure will not work anymore. Then, the knowledge base can become non-monotonic, in the sense that if you insert not of something at some point of time; later on, you deduce that fact, then you have to take out this not and not only do you have to take out this not, you have to take out whatever was deduced from this not, and that makes things quite complex.

But there are actually knowledge bases which use nonmonotonic reasoning, and they do maintain the consistency of the knowledge base as the direction procedure continues. But, as of now, if we just use modus ponens, we have this problem. So, this tells us that modus ponens is an incomplete reasoning procedure. It is not able to deduce whatever that can be deduced. For example, here, sa can be deduced, which we know from reasoning. From human reasoning, we know that sa can be deduced, but we cannot deduce it with modus ponens. So, it is an incomplete procedure.

There was a lot of research on the completeness of a proof procedure, and then Godel's completeness theorem partially solves this matter. It said, for first order logic, any sentence that is entailed by another set of sentences can be proved from that set. In other words, what this says is that, if there is any sentence that you can deduce from a given set of sentences, then, we will have a proof procedure that is able to do this deduction. In other words, the problem of deduction or to find out whether it is entailed, whether it can be deduced, is solvable- this is what Godel says.

(Refer Slide Time: 00:08:20)



But he did not suggest a proof procedure, and in 1965, Robinson published his resolution algorithm, which was proved to be complete proof procedure for first order logic. But there is a catch in the whole thing. This entailment in first order is semi-decidable, so what does this mean? It means that we can show that sentences follow from premises, if they do, but we cannot always show if they do not. Let us understand what this means.

Suppose there is some sentence that we want to deduce. Now, if it is the case that what we have in the knowledge base implies this particular sentence, then we will be able to deduce it. But if it is not implied, then we may not terminate, and that applies to all proof procedures of first order logic. That was **what was** the outcome of these 2 results- Godel's completeness theorem and Robinson's resolution algorithm.

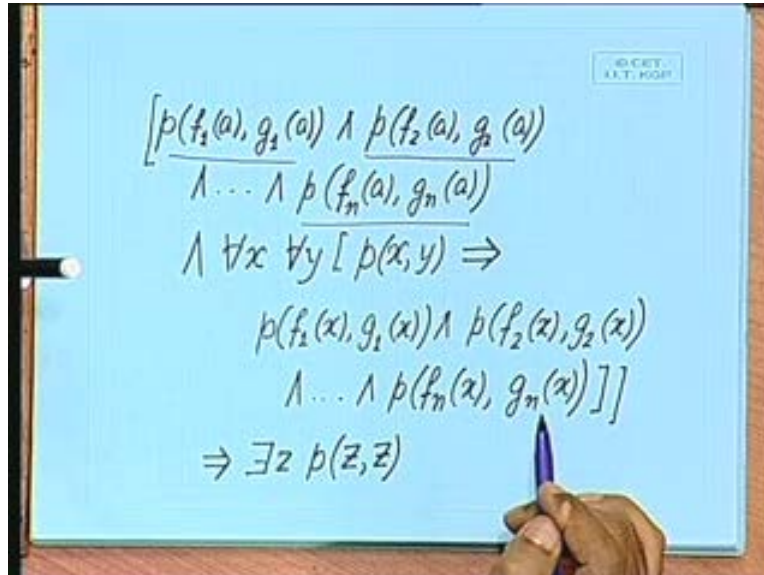
So, when we say that the resolution algorithm is complete, it means that whatever can be deduced will be deduced by the algorithm. But whatever is not implied or whatever is not entailed from the set of facts and sentences that you have in your knowledge base; whatever cannot be entailed, for those cases, the algorithm may not terminate, right, and there is no algorithm which can guarantee both. There is no algorithm which can guarantee both the yes answer and the no answer. Can be deduced: yes, cannot be deduced: no, right? Both of these answers cannot be guaranteed by any algorithm; that is the hardness of the problem itself, okay? That is why we say that entailment in first order logic is semi-decidable; it is decidable on 1 side, is not decidable on the other side.

(Refer Slide Time: 00:11:55)

The slide is titled "The validity problem of first-order logic" in yellow text on a dark blue background. Below the title, there are two bullet points in white text: "[Church] The validity problem of the first-order predicate calculus is partially solvable." and "Consider the following formula:". The formula is displayed on a light yellow rectangular background. It consists of three lines of mathematical notation: the first line is $\bigwedge_{i=1}^n p(f_i(a), g_i(a))$, the second line is $\wedge \forall x \forall y [p(x, y) \Rightarrow \bigwedge_{i=1}^n p(f_i(x), g_i(x))]$, and the third line is $\Rightarrow \exists z p(z, z)$. At the bottom left of the slide, there is a small blue box containing the text "CSE, IIT Kharagpur" and at the bottom right, the number "11".

We will not go into the details of the validity problem of first order logic, but we will just get a glimpse, or try to see why this becomes so difficult. With this example, I will show you why this becomes difficult. But I also encourage you to study Church's proof of the **of the** fact that the validity problem of the first order predicate calculus is partially solvable. This proof is available in several books including Russell Norvig and ___ Manna's book. So, you should, at some point of time, have a look at how this proof looks- like, it is not a very big proof, but it requires a little bit of breaking your head into it. Now, let us see this formula. What we have here is a conjunction of predicates p, each having a different argument. We have p f1a, g1a, and p f2a, g2a, so, let me expand it out and write. This has p f1a, g1a and p f2a, g2a, and like this up to p fna, gna.

(Refer Slide Time: 00:16:29)



And we have for all x , for all y , p of x y implies p of f_1x , g_1x , and p of f_2x , g_2x . And then, this whole thing **this whole thing** from here to here implies, there exists z - now, let us see what this says. It says that this is the starting point: we are given a value a , this a is a constant- **this a is a cons** actually, I should have written it in capital letters: this a is a constant. So, we are given that p of f_1a , g_1a is true; p of f_2a , g_2a is true, and so on, and p of f_na , g_na is true, right? And we are given this rule, which says that for all x , for all y , if p of xy is true, then, we have p of f_1x , g_1x , is true and p of f_2x , g_2x is true, and so on; p of f_nx , g_nx is true, right? And then, we say that if this whole thing is given to us, can we deduce that there exists some z , where the arguments of p will become equal? Now, let us see- how would forward chaining work?

It will start with the knowledge base having p ; having this part, it will have p of f_1a , g_1a ; it will have p of f_2a , g_2a , and it will have p of f_na , g_na , right, this is what the knowledge base will have. And then, you will repeatedly use this rule, because you will take each of this p of xy , and then you will deduce this whole set of statements from that. So, when you take say p of f_1a , g_1a , then, x will be instantiated with f_1a , y will be instantiated with g_1a , and then, you will have p of f_1 of f_1a , g_1 of f_1a , right, and so on, you generate this whole stuff. Again, you will use that repeatedly. But **whether these functions will eventually by repeated application of these functions** whether these 2 functions will take you to a common value z , is the question that we have to answer.

Now, these functions can be arbitrarily complex functions, right, and to determine whether they will actually converge to a common value, is a difficult problem. And this is where the unsatisfiability will create, because see, **these will** these can keep on growing, without you being able to infer whether they will arrive at a common value. So, actually, the proof which tries to establish that this is partially solvable, starts with a formula like this, and then develops it to lead you to the proof that- you will not be able to, if you are able to deduce this, if you are able to find the z , then you are done.

And if you progressively systematically expand these, then if there is a z, you will find- because you can recursively start enumerating this- the values for which this p holds, right, you can recursively start enumerating that. So, if there exists a z, then eventually, you will find it, but if there does not exist a z, you will keep on applying this repeatedly, and you will not come for it, you will not terminate it. So, I am not going into the detail proof in the class, but please study that. Here is the generalized resolution rule, so let us study this closely.

(Refer Slide Time: 00:19:05)

Resolution

- Generalized Resolution Rule:
For atoms p_i, q_i, r_i, s_i , where $\text{Unify}(p_i, q_i) = \theta$, we have:

$$p_1 \wedge \dots \wedge p_j \wedge \dots \wedge p_{n1} \Rightarrow r_1 \vee \dots \vee r_{n2}$$

$$s_1 \wedge \dots \wedge s_{n3} \Rightarrow q_1 \vee \dots \vee q_k \wedge \dots \vee q_{n4}$$

SUBST(θ ,

$$p_1 \wedge \dots \wedge p_{j-1} \wedge p_{j+1} \wedge \dots \wedge p_{n1} \wedge s_1 \wedge \dots \wedge s_{n3}$$

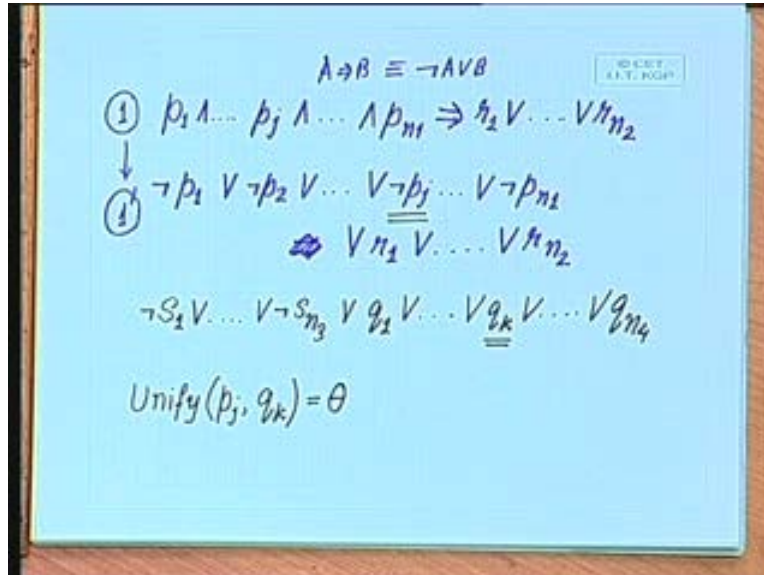
$$\Rightarrow r_1 \vee \dots \vee r_{n2} \vee \dots \vee q_{k-1} \vee q_{k+1} \vee \dots \vee q_{n4})$$

CSE, IIT Khargpur 15

We are given, say, 2 rules, which has p_1 through p_{n1} it implies r_1 through r_{n2} , and we have s_1 through s_{n3} , which gives us q_1 through q_{n4} , and for some j and some k , we have $\text{unify}(p_j, q_k) = \theta$. It means that 1 of these this p_j out here and this q_k out here- they unify, and they unify under the substitution θ . Now, do you see what we have here? We have this q_k here and this p_j here, and they unify with substitution θ . If they do, then we can deduce p_1 through p_{j-1} , so, excluding this p_j , and p_{j+1} through p_{n1} , right, and this s_1 through s_{n3} - this whole thing together, will imply r_1 through r_{n2} , and on this side,oops, I have missed out the q_1 here.

It is q_1 through q_{k-1} , or q_{k+1} to q_{n4} - there is a q_1 here, which I missed out. Now, try to see- how do we get this? How do we get this, okay? Let me rewrite let me rewrite this stuff. The first 1 which tells us p_1 through p_j ; p_{n1} implies r_1 or r_{n2} . We can remove the implication, and write this as not of p_1 , or not of p_2 , or not of p_j , or not of p_{n1} implies- sorry, we have eliminated the implication.

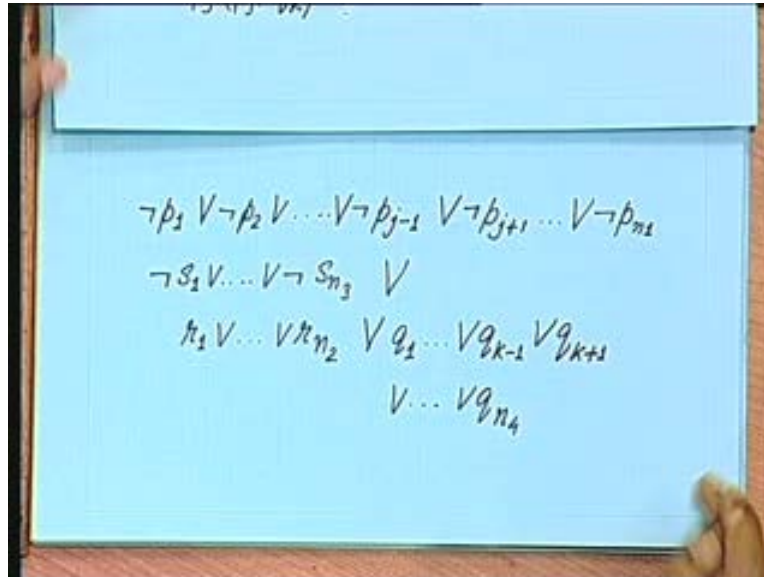
(Refer Slide Time: 00:24:18)



So, not of this thing, or r1 or r of n2, right? This is the first one. This 1 to this 1 dash, same thing, but we have written it **without** by eliminating the implication. Recall that we eliminate implication by writing a implies b is not a or b, which is equivalent to not a or b- that is how we got this, right? Then, likewise, if you look at the second one, we have s1, so we will have not of s1 or ..., or not of sn3, right? Or q1- this is the second state. And we are given that our pj here- so, we have unify the pj here, this pj- it unifies with qk; this qk, with the substitution theta, right? Then, if these 2 become the same- if this pj, mind you, it is pj which unifies, not not pj, right? Then, when these 2 under the substitution- these 2 will become the same under the substitution theta; this will become the same.

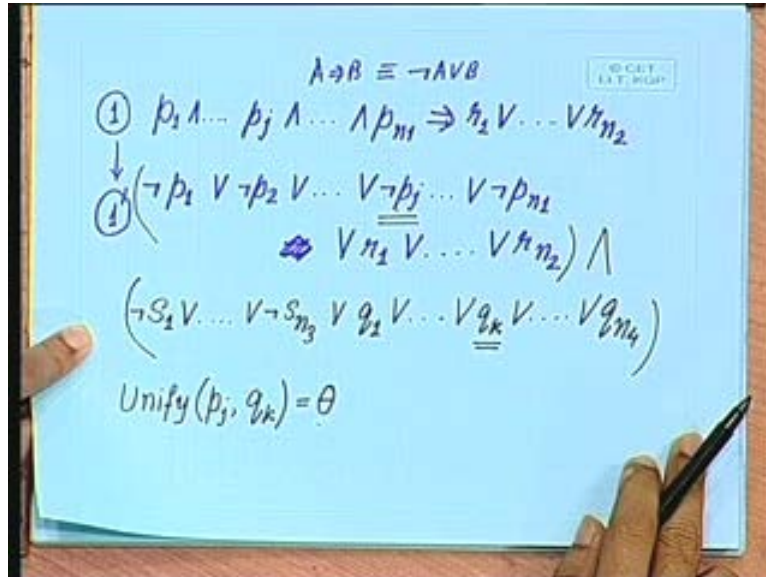
If they become the same, then either this is true or this is true, because we have not here. If we call this, say j, then this is either j is true or not j is true, so it is either of these 2 which is true. If either of these 2 is true, then we can say that the or of this whole thing is true, right? So, we can deduce the or of this whole thing, and when you take the or of this whole thing, and again turn it back into implicational form. If you take the or of this whole thing, what are we going to have?

(Refer Slide Time: 00:26:44)



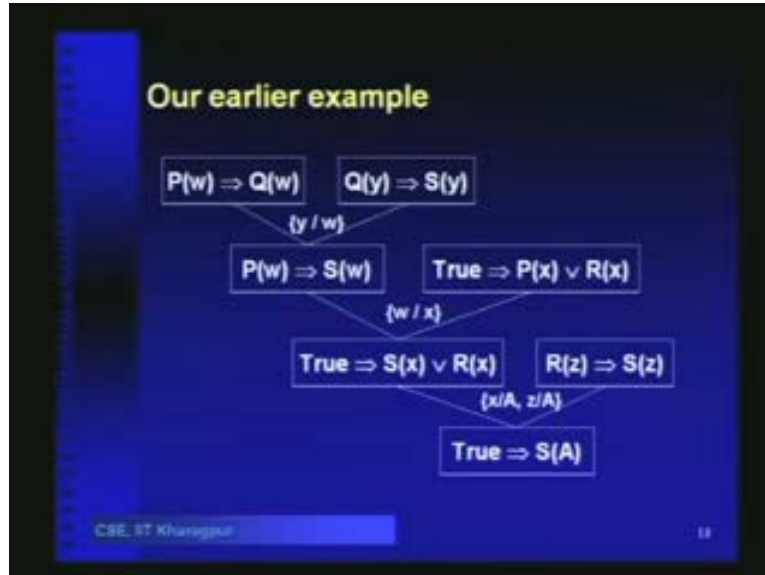
We are going to have not of p_1 or not of p_2 , or not of p_j minus 1, or not of p_j plus 1, right, ...- not of p_{n_1} . And then, if you take these nots, then we will have not of s_1 through not of s_{n_3} , right? Then, with this or of- we have the r_1 through r_{n_2} and we have, with that, q_1 through q_{k-1} or q_{k+1} , or ... q_{n_4} . How did we arrive at this? We observed that this- if we take this and this, then, either this is true or this is true. So, actually, what do we have? We have this whole thing, and this of whole thing, and if you have 1 literal, which is the complement of the other, then it becomes the or of the whole thing, right? So, we are able to get this stuff. Now, we if we translate this back into implicational form, then we will have p_1 through p_j minus 1 conjunction of del, right? And then, p_j plus 1 through p_{n_1} , and s_1 through s_{n_3} , will imply this whole thing will go to the left hand side of the implication.

(Refer Slide Time: 00:27:09)



This or will become the implication, and we will have this conjunction- the disjunction of these things. That is what we have out here, **in the um out here** that we have p_1 through p_j minus 1, and p_j plus 1 through p_{n_1} , and s_1 through s_{n_3} . Clear? This is the generalized resolution rule, and I will ask you to verify that this rule will encompass all the rules that we have seen so far. I mean, all the rules, in the sense that it will also encompass modus ponens that we have seen. Now, let us see our earlier examples and see whether it is solved by this. Do you remember the example that we had? Or, let me write it down once again? Yes, we had this example. We had, for all x , p_x implies q_x and for all x , not p_x implies r_x . Then, we have, for all x , q_x implies s_x and **for all or x** for all x , r_x implies s_x . We will use this now, here.

(Refer Slide Time: 00:28:25)



So, we start with pw implies qw , and qy implies sy , right? Now, here we have the substitution of y being replaced by w , to give us pw implies sw . This is simple modus ponens; there is no resolution rule being applied here. Then, when we come here, this is where we find slightly different thing. This is a sentence- this true implies px or rx , is a sentence, which you cannot express in horn logic, because you have or on this side, right, but we can always express that for resolution. So, you have true implies px or rx , which says that either px or rx must be true every time. If we substitute w by x , then we have px implies sx , and therefore, we have true implies sx or rx . Then, we have rz implies sz , and again, if we use a in place of x , and z in place of z , then this becomes ra implies sa , right, and this becomes true implies sa or ra .

So, that gives us true implies sa or sa , which is sa . The difference that we had with modus ponens was that, we were not being able to use rules of this form in modus ponens and we are now being able to use them. Is this clear? Any questions? Right. Now, how we do this thing in an automated way? We will allow the user to write the formulas in first order logic, using the normal constants of first order logic in any way. And then, we will try to reduce the formula into a canonical form- into a form from which resolution will be able to work. Now, what kinds of canonical forms do you have for normal satisfiability, cnf- conjunctive normal form, injunctive normal form, right? So, these are standard normal forms into which you can deduce a Boolean function. There are other canonical representations, like binary decision diagrams.

For this, we will also convert to something which is called a clause form, and clause form is like cnf, where you have a conjunction of clauses, and each clause is a disjunction of literals. That is what we mean by conjunctive normal form in propositional logic. Here, we have these quantifiers, so, we have to find out some mechanism of getting rid of the quantifiers, before we can apply those kinds of strategies. So, the first step; this is the

definition of the clause form, where we will have only universal quantifiers, followed by a conjunction of clauses; each of these clauses can be a disjunction of some predicates. All first order logic formulas can be converted to clause form, and we will demonstrate this conversion on this formula, okay? We will demonstrate the conversion on this formula. Now, follow the steps closely.

Step 1: we will take the existential closure and eliminate redundant quantifiers. So, if there is any quantifier which is redundant, that will be eliminated, and we will take the existential closure of those variables which are not quantified. For example, here, we have this- there exist z, but there is no z here; there is no z in the formula, we have this- there exist z.

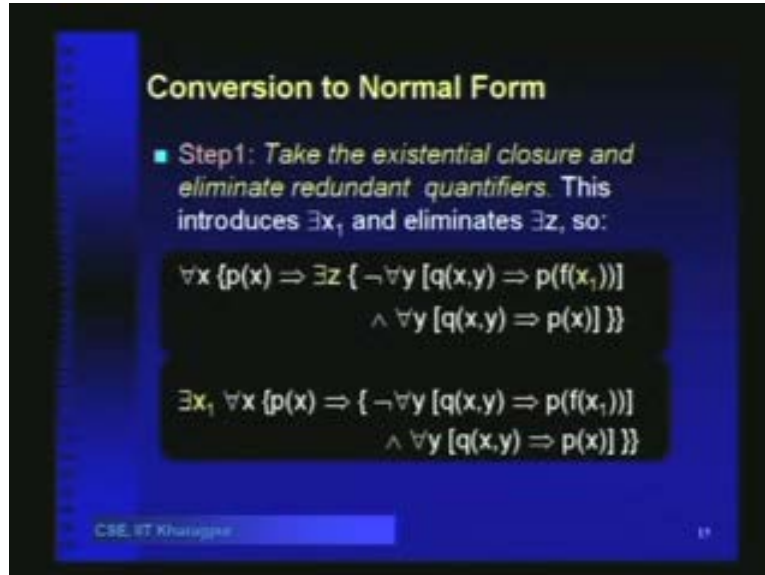
(Refer Slide Time: 00:33:15)

Conversion to Normal Form

- A formula is said to be in clause form if it is of the form:
$$\forall x_1 \forall x_2 \dots \forall x_n [C_1 \wedge C_2 \wedge \dots \wedge C_k]$$
- All first-order logic formulas can be converted to clause form
- We shall demonstrate the conversion on the formula:
$$\forall x \{ p(x) \Rightarrow \exists z \{ \neg \forall y [q(x,y) \Rightarrow p(f(x))] \wedge \forall y [q(x,y) \Rightarrow p(x)] \} \}$$

CSE, IIT Kharagpur 14

(Refer Slide Time: 00:34:16)



Conversion to Normal Form

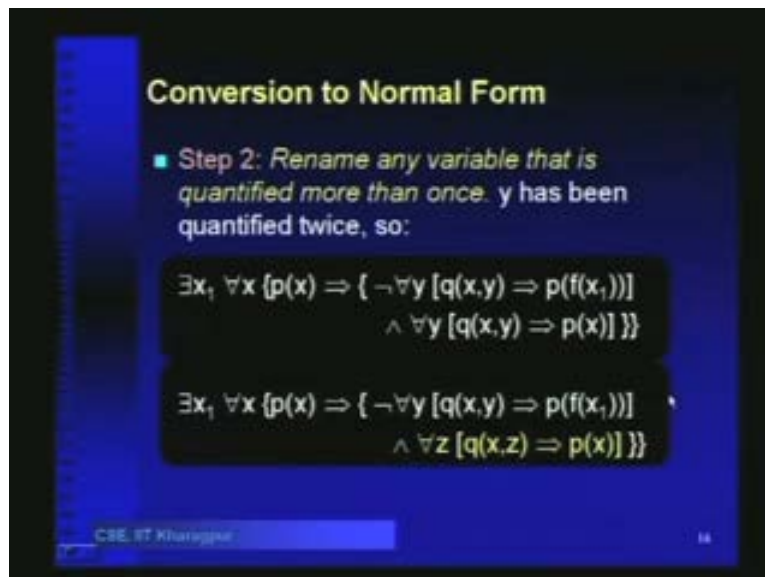
- Step 1: Take the existential closure and eliminate redundant quantifiers. This introduces $\exists x_1$ and eliminates $\exists z$, so:

$$\forall x \{ p(x) \Rightarrow \exists z \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,))] \} \wedge \forall y [q(x,y) \Rightarrow p(x)] \}$$
$$\exists x_1 \forall x \{ p(x) \Rightarrow \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,))] \} \wedge \forall y [q(x,y) \Rightarrow p(x)] \}$$

CSE, IIT Kharagpur 13

So, there exist z , can be safely dropped, right? This is the elimination of a redundant quantifier. Also, here, we have this x_1 , and there is no quantifier on x_1 . In the absence of the quantifier, we will put an existential quantifier here. We will see whether there exists some x_1 , for which this whole thing holds. That is step 1, clear?

(Refer Slide Time: 00:35:30)



Conversion to Normal Form

- Step 2: Rename any variable that is quantified more than once. y has been quantified twice, so:

$$\exists x_1 \forall x \{ p(x) \Rightarrow \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,))] \} \wedge \forall y [q(x,y) \Rightarrow p(x)] \}$$
$$\exists x_1 \forall x \{ p(x) \Rightarrow \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,))] \} \wedge \forall z [q(x,z) \Rightarrow p(x)] \}$$

CSE, IIT Kharagpur 14

Step 2: rename any variable that is quantified more than once. This y has been quantified more than once- see, we have this- for all y here, we have this for all y here: Now, this y and this y need not have the same value, because they are separately quantified. If this 'for all y ' was not there, then these y 's would be the same y , because

they would come under the scope of the same quantifier. Because they are under different quantifiers, therefore, these 2 ys are different. So, we will rename 1 of them. We will rename this y to z, and here, we will have for all z, qxz implies px, clear? So, any quantifier- any variable which is quantified more than once- for that, we will just give it a new name. If you change the name of the variable, it does not change anything, as long as it is quantified, okay?

(Refer Slide Time: 00:36:37)

Conversion to Normal Form

- Step 3: *Eliminate implication.*

$$\exists x_1 \forall x \{ p(x) \Rightarrow [\neg \forall y [q(x,y) \Rightarrow p(f(x_1))] \wedge \forall z [q(x,z) \Rightarrow p(x)]] \}$$

$$\exists x_1 \forall x \{ \neg p(x) \vee [\neg \forall y [\neg q(x,y) \vee p(f(x_1))] \wedge \forall z [\neg q(x,z) \vee p(x)]] \}$$

CSE, IIT Kharagpur 17

Step 3: we will eliminate implication. This is straight forward. We know how to do that. This implication is eliminated by not of px, or this whole thing, right? Not a or b.

(Refer Slide Time: 00:36:56)

Conversion to Normal Form

- Step 4: *Move \neg all the way inwards.*

$$\exists x_1 \forall x \{ \neg p(x) \vee [\neg \forall y [\neg q(x,y) \vee p(f(x_1))] \wedge \forall z [\neg q(x,z) \vee p(x)]] \}$$

$$\exists x_1 \forall x \{ \neg p(x) \vee [\exists y [q(x,y) \wedge \neg p(f(x_1))] \wedge \forall z [\neg q(x,z) \vee p(x)]] \}$$

CSE, IIT Kharagpur 18

Step 4: move the negation all the way inwards- we will move this negation all the way inwards. Here, this negation was not directly on the predicates- this negation was on this whole formula, on the sub-formula. So, we will move it inwards. By moving it inwards, we have this- there exists y for all y becomes, there exists y, and we have $q(x,y)$ and not of $p(f(x))$, right? So, **this not** this not has been moved inwards, right?

(Refer Slide Time: 00:37:47)

Conversion to Normal Form

- Step 5: Push the quantifiers to the right.

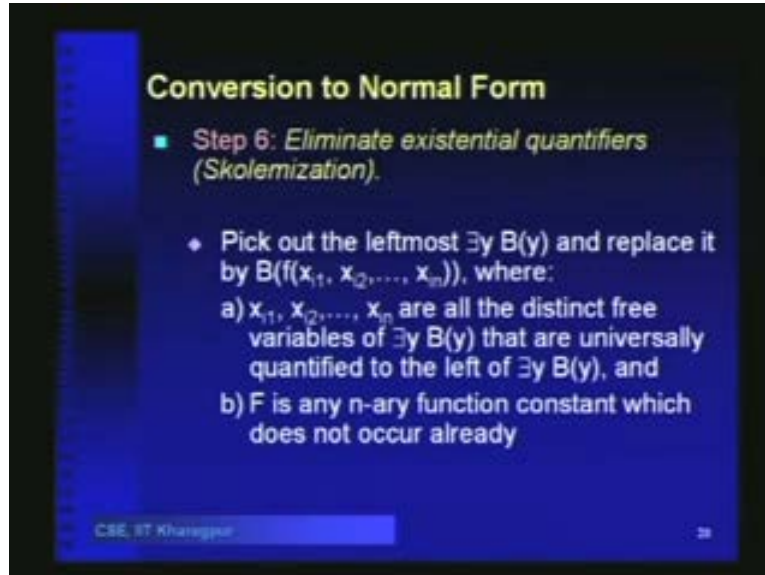
$$\exists x, \forall x \{ \neg p(x) \vee [\exists y [q(x,y) \wedge \neg p(f(x))] \wedge \forall z [\neg q(x,z) \vee p(x)]] \}$$

$$\exists x, \forall x \{ \neg p(x) \vee [[\exists y q(x,y) \wedge \neg p(f(x))] \wedge [\forall z \neg q(x,z) \vee p(x)]] \}$$

CSE, VT Kharagpur 18

Step 5: push the quantifiers to the right. Here, this, there exists y was quantifying this whole formula, but then, this part- this not of $p(f(x))$, does not have y. So, we can associate this there exist y directly with this predicate $q(x,y)$. That is what we mean: we move it inside, and align it with the predicates which they are quantifying. So, it is, here, there exists y $q(x,y)$. Likewise, this for all z, is not quantifying this $p(x)$, because it does not have z, so we have moved it inwards, right? Then, this is an important step: eliminate the existential quantifiers using Skolemization.

(Refer Slide Time: 00:38:40)



Conversion to Normal Form

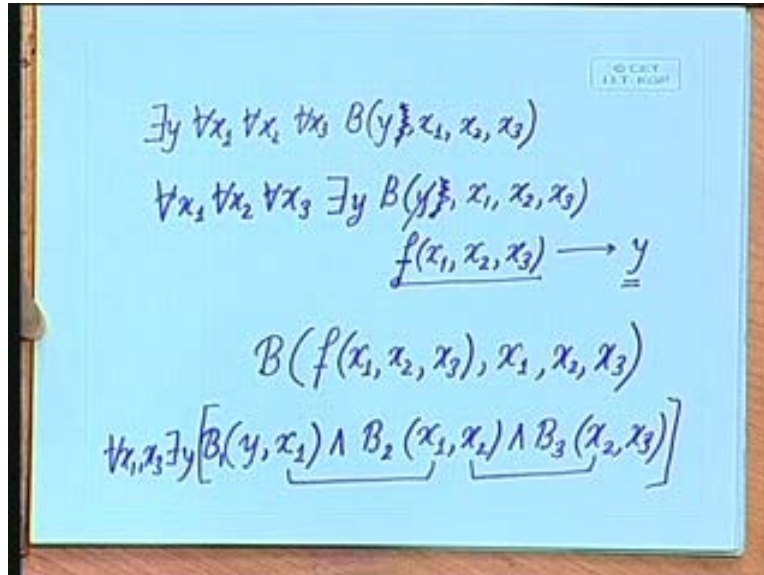
- *Step 6: Eliminate existential quantifiers (Skolemization).*
 - ◆ Pick out the leftmost $\exists y B(y)$ and replace it by $B(f(x_1, x_2, \dots, x_n))$, where:
 - a) x_1, x_2, \dots, x_n are all the distinct free variables of $\exists y B(y)$ that are universally quantified to the left of $\exists y B(y)$, and
 - b) f is any n -ary function constant which does not occur already

CSE, IIT Kharsappur 28

We had briefly seen what we mean by Skolemization: we replace a existentially quantified variable by some ground thing, by some ground term, or some something else, in order to get rid of the existential quantifier. But, we can do it with only something which is not there in the knowledge base at the moment. So, what we do is, we pick out the left most- there exists y by and replace it with f of some x_1 through x_n . Where these are all the free variables of there exists y by, that are universally quantified to the left of there exists y by, right? Now, let me explain this through an example. That suppose, we have there exists y by, right, and you have some for all x_1 , for all x_2 , for all x_3 , here.

Now see, what does this say- that for all x_1 , for all x_2 , for all x_3 , there exists y , there exists such a y ; so, given a value of x_1 , x_2 and x_3 , there will be a y which satisfies y . This is not the same as writing- there exists y for all x_1 , for all x_2 , for all x_3 by. What is the difference between these 2?

(Refer Slide Time: 00:46:34)



In the first one, there has to be a single value of y , for which no matter what is the value of x_1 , x_2 and x_3 , by will be satisfied, right? But here, it says that for all values of x_1 , x_2 and x_3 - for each of those combinations of x_1 , x_2 , x_3 - there has to be some y . So, if x_1 is a , x_2 is b and x_3 is c , then we can have 1 y . If x_1 is p , x_2 is q and x_3 is r , then we can have another y . Those 2 y s need not be the same y , right? Therefore, when we replace this y with a skolem, **we will write** we will replace this by a function of x_1 , x_2 , x_3 . Given the value of x_1 , x_2 , x_3 , y is that value which satisfies by. Now, the value of y which satisfies by, for this x_1 , x_2 and x_3 , is the function of these 3.

So, we can have a mapping from triplets of x_1 , x_2 , x_3 , to the value of y which satisfies by. For a different value of x_1 , x_2 and x_3 , you will get a different y , possibly, a different y , right? Is this clear? What we mean by Skolemization? (Student speaking). Here? No, because it is a function. For any given value of x_1 , x_2 , x_3 , there will be a y . There can be more than 1 y , like we had. You remember that there exists x likes x ice cream, and we replace x by man, there can be more than 1 x who likes ice cream, but man is 1 which satisfies, right? There could be others also. Similarly, we are interested in finding out whether there exists any such y . If we are able to find out a function which gives you 1 y for every combination of x_1 , x_2 , x_3 , then this formula is valid.

So, it is the question of this; the question is whether such a function exists, that is- what is going to determine whether this whole formula is valid or not? If it is **if it it is** valid, provided that there is a function like this, which gives, for every value of x_1 , x_2 and x_3 , or value of y , which satisfies by, but, I agree that there can be otherwise which will satisfy the y . What we have here is, we eliminate the existential quantifiers. Slides please. We have these existential quantifiers. So, this is the formula- we write it down- that pick out the left most, there exists y by, and replace it by this b of x_1 , f of x_1 , through x_n , where this x_i 1 through x_i n , are the free variables of by, that are universally quantified to the left of there exists y by. Now, 1 thing that **no 1** none of you asked me here was- text

please, text please. Yes, we had- none of you asked us, that look, this y- if this y, if this predicate does not have x1, x2, x3, then how does it matter? I mean, why does it have to be a function of this x1, x2, x3?

In this case, we do not need this. We need this when you have- here also, okay? Then, Skolemization of this whole thing is going to give us b of- , right? This x1, x2, x3 could be in the same predicate, or it could be some- it could also be a collection of predicates. For example, this b could be a formula of the form, say by x1, right, and then, some say b1 and b2 x1 x2 and b3 x2 x3. See, even in this case, though they do not appear in the same predicate, but there is a binding which is there like this. Therefore, this y- if you have for all x1 through x3, here, and you have there exists y, then to skolemize this y, you will have to put function of x1, x2, x3, right, because though you do not have x1, x2, x3, all in this formula, in this predicate, there is a binding which has this. So, this whole thing is our b, right? Fine. So, this x1 through- xi 1 through xin are all the distinct free variables of- there exists y by, that are universally quantified to the left of there exists y by.

(Refer Slide Time: 00:46:55)

Conversion to Normal Form

- Skolemization:

$$\exists x, \forall x \{ \neg p(x) \vee [[\exists y q(x,y) \wedge \neg p(f(x))] \wedge [\forall z \neg q(x,z) \vee p(x)]] \}$$

$$\forall x \{ \neg p(x) \vee [[q(x,g(x)) \wedge \neg p(f(a))] \wedge [\forall z \neg q(x,z) \vee p(x)]] \}$$

CSE, IIT Kharagpur 21

In this case, for example, in our example- we have this; there exists y q xy, and we replace this y by gx. Why? Because this x is universally quantified here, so the value of y that we have here has to be a function of that x, because there exists y is inside this for all x. So, given the value of x, the value of y will be a function of that x, so, we have this gx. This is the skolem function- we call this the skolem function. For the x1, there no such requirement, because there is no other variable in this predicate, which is universally quantified from the outside, so, we simply replace it by a constant a, which does not appear anywhere in the knowledge base. Also, this function g that you have here, has to be a new constant, new function constant- it should not be 1 of the existing ones in the knowledge base, right?

(Refer Slide Time: 00:48:07)

Conversion to Normal Form

- Step 7: Move all universal quantifiers to the left

$$\forall x \{ \neg p(x) \vee [[q(x, g(x)) \wedge \neg p(f(a))] \wedge [\forall z \neg q(x, z) \vee p(x)]] \}$$
$$\forall x \forall z \{ \neg p(x) \vee [[q(x, g(x)) \wedge \neg p(f(a))] \wedge [\neg q(x, z) \vee p(x)]] \}$$

CSE, VT Kharagpur

Finally, we move all universal quantifiers to the left. Now, we do not have any existential quantifiers, so we can move the universal quantifiers to the left. Previously, we could not do this, because the existential quantifiers were there, and you can not move a universal quantifier across an existential quantifier. But here, now, we do not have any problem, so we have moved it to the left. So this- for all z, has come to the left. So, on the right side, we now have just a set of predicates which are quantified from the outside. And now, what we are going to do is, we will distribute and over or, to bring it to clause form. See, up to this part is what is mainly the jugglery with first order logic- that we have to do. Once you have all the quantifiers to the left side and they are all universal quantifiers, what you have on the right side is like a set of predicates, which are quantified from the outside.

(Refer Slide Time: 00:48:45)

The slide is titled "Conversion to Normal Form" and is set against a dark blue background. It contains two main steps:

- Step 8: Distribute \wedge over \vee .** Below this step, a mathematical expression is shown in a white box:
$$\forall x \forall z \{ [-p(x) \vee q(x, g(x))] \wedge [-p(x) \vee -p(f(a))] \wedge [-p(x) \vee -q(x, z) \vee p(x)] \}$$
- Step 9: (Optional) Simplify** Below this step, a simplified mathematical expression is shown in a white box:
$$\forall x \{ [-p(x) \vee q(x, g(x))] \wedge -p(f(a)) \}$$

At the bottom left of the slide, the text "CSE, IIT Kharagpur" is visible. At the bottom right, the number "23" is displayed.

So, we can use our Boolean algebra on these, to convert this stuff into our cnf form. That is what we do here. We have distributed the and over or, and that gives us a set of 3 clauses. 1 is not px or qx gx, the second class is not px or not of pfa, third class is this, right? And then, we simplify this- this is an optional step to obtain this: not of px or qx gx as 1 clause, and not of pfa as other clause. How do we get this from this? Try to solve it. Take this down as an exercise, and just try to see how we deduce this thing from this. This is- it is not difficult, **it is just give you an idea of** it will give you an idea of how to use unification, in order to get rid of the third clause, okay?

So, in the next lecture, **we will** we will see how we can use resolution refutation with some examples, to deduce or to prove certain sentences in first order logic. Yes, the slide please. Thank you. **Just to** We come to the end of this lecture here.