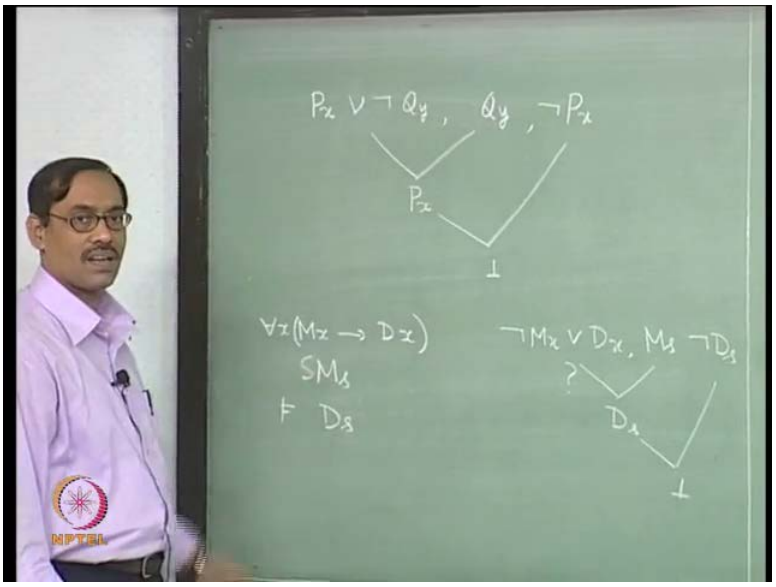
Mathematical Logic
Prof. Arindama Singh
Department of Mathematics
Indian Institute of Technology, Madras

Lecture - 36
Most General Unifiers

So, we had seen that a formula is satisfiable if and only if its SCNF is satisfiable. Our aim was to extend resolution method from propositional logic to first order logic. Now, let us see how the resolution process goes and then whatever we need we will have to introduce.

(Refer Slide Time: 00:28)



Suppose we start with these clauses, say, I have one clause having P x and say not Q x or say, not Q y; all this is one clause. That means we will be writing it as and, or which one? It is a clause, the formula is c n f; so it will be written as and, we are taking a clause, we will have, so, let us take P x and not Q y, we will take or, here, this is one clause. Suppose if I have another clause which is, say, Q y and another clause, which is simply P x. This might have come from one entailment, one consequence like, for each x each y Q y implies P x.

Similarly, you can say this is for each y Q y, and then on the right side, you have entails; you want P x, so it will be in the form: there is x not P x, some such thing. Then when you use reductio ad absurdum, bring it to the other side, that becomes for all. Then, you

have P x. Now it is easy to apply resolution on this. You can just take this clause plus this clause and you have negation of this, negation of this. So, it, exists P x, then that is all it will say. Nowhere It will move, after this.

Suppose you have not P x here. Then of course you can use this, this and reduce it to bottom. That will be a resolution refutation of the clause set. So, when you take set; that means, this is a clause, and this is another clause; and the other one, just like your PL, your c n f is that. Is that ok? It looks, we can extend. That is what it says. But let us not hurry through; see another argument. For example, you have, say, all men are mortal. That, you would try to write as, say, M x implies, some, D x for each x; which will give rise to the clause not M x or D x. Then, you have another, say, Socrates is a man. So, you would write M s, s is a constant. From which you want to conclude that Socrates is mortal; so that will be D s.

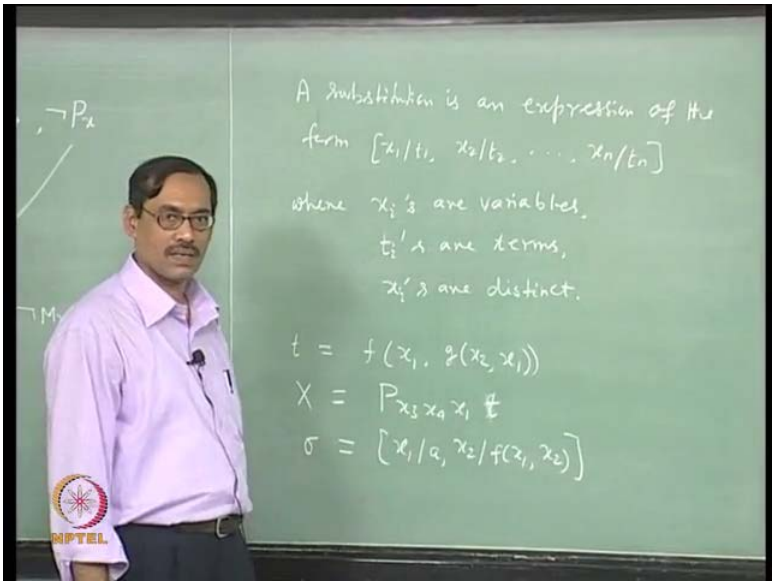
Now, in this side you have the clause set as not M x or D x, then another clause is M s. And, using Reductio ad abusrdum, not D s will go to the premises set; so you have not D s. There should be a refutation of this. That is what we feel. So, first thing is, how to conclude from not M x and M s? This should resolve. This is resolved, and we should get something like D s here. So, this step we are not convinced. But something has to be done. Then, from these two steps you would get a refutation. But what does this step mean? It really corresponds to the argument, here, for each x M x implies D x, M s, therefore D s. That is what we infer here; that should be correct. But you do not know how to tackle that in the resolution process if you just follow the PL resolution.

Somehow, for each x, you have to go for any particular s. All that we see is, x has become substituted by s. So, substitutions we have to take care. Now, wherever we want this to be done by a machine, not always looking at it and finding the appropriate substitution; So, a machine should be able to find out what should be the suitable substitution there. Now our responsibility is to formalize this part; how mechanically it can be done.

For that purpose, let us start studying substitution a bit; they might be helpful. One more thing is, you may not always be able to solve using one variable under term substitution. There can be many variables and many substitutions of terms by those variables, or replacing those variables simultaneously. So, it will be something like a set of

expressions of the form x by t. But we can confuse with the set, already sets are there in clauses, we will use some other notation instead of curly brackets. Let us write square brackets.

(Refer Slide Time: 06:12)



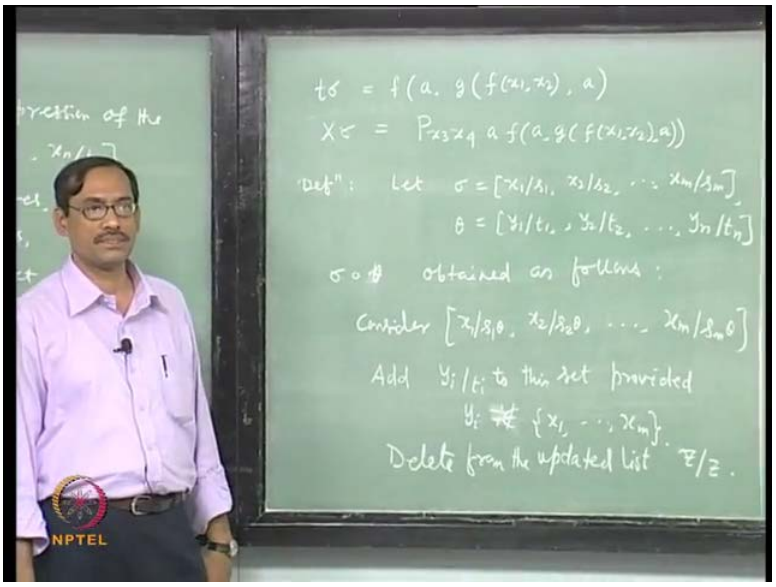
So, we say that a substitution is an expression, it is really a set written in a different way, of the form say x 1 by t 1, x 2 by t 2, x n by t n. Simultaneously, all these x 1 to x n will be substituted by their corresponding terms t 1 to t n; that is what I mean. Now, where these x i's should be variables and t i's are terms. and something else we want; since we want it to be simultaneously substituted because you have x by a here and also x by b there, in which to substitute we do not know. So, we also assume that all these x i's are distinct. So, x i's are distinct; this was at least one unit, that is how you will be tackling the substitutions.

Suppose you have come one step in the resolution process using the substitutions; now you see that other substitution is required. You should be able to compose the substitutions in the earlier substitutions. So, first we have to study what is composition of substitutions. Slowly, we will be dragged to show many things because we require resolution to work, but let us see how this substitution works.

Suppose, I have a term t which is equal to f of x 1 comma g f x 2 x 1. I take a formula which is say P x 3, x 4, x 1 and then this t. Suppose, I take the substitution sigma equal to

say, x 1 by a, x 2 by f of x 1. Let f be of two variables x 1 and say x 2 again; so we will just start with this. Now let us apply this sigma on t.

(Refer Slide Time: 09:00)



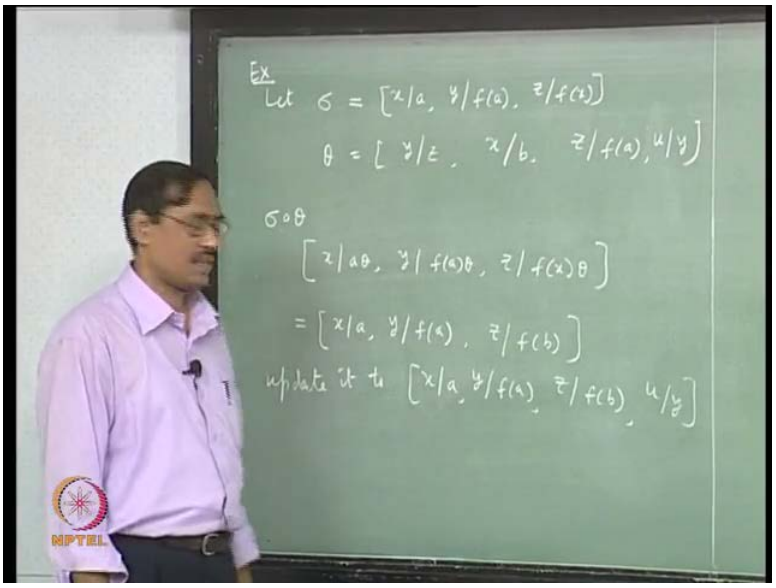
We will write that as t sigma. Sometimes if its readability is at problem, we will write another bracket here for t or brackets for sigma and others. Sometimes you will use that. Let us see what is t sigma. What will it be? At a time we have to substitute all these things; that should happen simultaneously, not one after another. So, x 1 will be substituted by a, that means it will be equal to f of a. Next is, g of x 2 will be substituted by f of x 1 x 2; so that is f of x 1 x 2, that is, x 2 is gone. Next, x 2, x 1 is substituted by a, that is it. We are not supposed to write again substitution to x 1, x 2; that will be again t sigma, another sigma will be coming up; that is how you will be proceeding.

So, you have x sigma. Now, x is P x t x for x 1; so they are not substituted at all. There kept as that x 4, but this x 1 becomes now a, and then t; is that clear? That is how you'll be proceeding. Now, for the compositions. We need compositions. Let us define how composition is taking care of the replacements. We again give a definition.

Let, say, sigma is equal to given substitution in the form x 1 over s 1. There are m variables to be replaced by m terms; and say, theta is equal, similarly, for y 1 by t 1, y n by t n; they can be different in number. So, we define sigma composition with theta as another substitution; this will be a substitution, that we have to check also. So, obtained in the following way. Well, how do you suggest? Suppose I have this theta; I am taking it

from the right side. That means, here, there will be t or some x on which it will be applied. That means, first sigma is applied, then on that theta is applied. Suppose in sigma x substituted to a, x changes to y. Let us say now, in theta y changes to a, then ultimately x will change to a. So, that means there will be a component in sigma which is of the form x by y, and there is a component of the form y by a in theta. In place of x you are going to replace y, theta y; and y has changed to a again; so theta is really applied to y, the denominator. So, you must consider the set first. Consider x 1 by s 1 theta, x 2 by s 2 theta, x m by s n theta; proceed. There is some change. So, you first use this theta on all the denominators. Now what happens is, there can be some others y's which are not covered in x 1 to x n. This also we added. Because on a formula, suppose this a formula; like your x 3, here, I apply theta, where x 3 is there. I do not have x 1, x 2 for simplicity, but x 3 is there; say x 3 by a. Now, x 3 by a should also be added here, otherwise it will not give a composition. So, you have to add all those things. Now, add y i by t i to this set provided y i is not equal to any of this x. That means, it does not belong to x 1 to x m. If that y is already covered here, then this one should have taken care. So, you do not have to add those things. I will add only those which are already not covered in x 1 to x m. This may not be a substitution. After this addition of all these y i's you have updated it, so it is in the form x 1 to x m, s m by theta; and then some other y's which are not x's. But that need not be substitution because some of these things can give same variable say x 2 and s 2 is, or x 3, x 3 has become x 2 in theta, then x 2 by to x 2 is not allowed in a substitution. They should be different. Otherwise, it is vacuous. So that vacuousness you can really deal with, while defining the substitution itself. You may say no x i is equal to t I, that t I, right? You may say x i's are different and x i is not equal to t i. Otherwise, unnecessarily you will be writing it out, which is not required. So now, we will delete all those things, delete from the updated list. Any substitution of the form say z by z, if they are same, just delete. So this deletion will be done only from this side, really, not from y i's by t i's, because they are already in a substitution. So, they will be deleted from this set. Then we can update this z by z; instead of writing z by z, you can write delete x i by s i theta provided x i equal to s i theta. So, let us update it. Delete from the updated list x i by s i theta if x i is equal s i theta, right? That should give us the composition of sigma and theta. Let us see an example; how it proceeds.

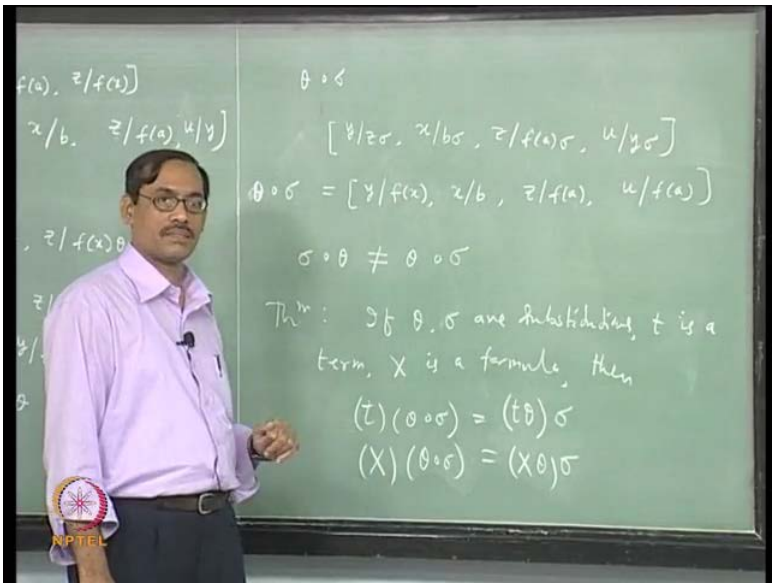
(Refer Slide Time: 16:58)



Suppose I take sigma equal to, let us start with x by a, y by f of a, z by f of x, and theta is equal to y by z, x by b, z by f of a. You can add one more say, u by y. Let us see what is sigma composed with theta. For computing this, first we have to take sigma, where denominators will be replaced by our u operated with theta. So, first we consider the list x by a theta, y by f of a theta, z by f of x theta; this is equal to, a is a constant, does not matter. So, x by a remains; f of a is also a closed term, so that remains; now x is a variable, so x is taken to b, so this becomes z by f of b. Next, what we do, add anything extra here; which is not x y z, so x is there, y is already there, z is there, only u is not there; so that is all; only u will be added. So, update it to x by a, y by f of a, z by f of b, and u by y. Now, you have to delete all those things which are repetitions; x a y f a z will be, if there is no repetition here, however z by z, then you would have to remove it; otherwise leave it as it is. So, this is your sigma composed with theta.

What about theta composed with sigma? Let us try to see what it is. So then I have to consider first theta; denominators will be replaced by or operated over sigma. That gives theta as y by z sigma x by b sigma z by f of a sigma u by y sigma. So this is equal to y by z sigma, which is f of x, x by b, because b is a closed term, f is a closed term. So, f of a, as it is, u by y sigma, so y sigma is f of a. Then there is nothing to be added; in sigma we have x y z already. We have x y z here; so nothing to be added. Now anything to be deleted? No, because no numerator is same as the denominator here. If they are same, then you have to delete it. So that is your theta operated with sigma.

(Refer Slide Time: 19:32)



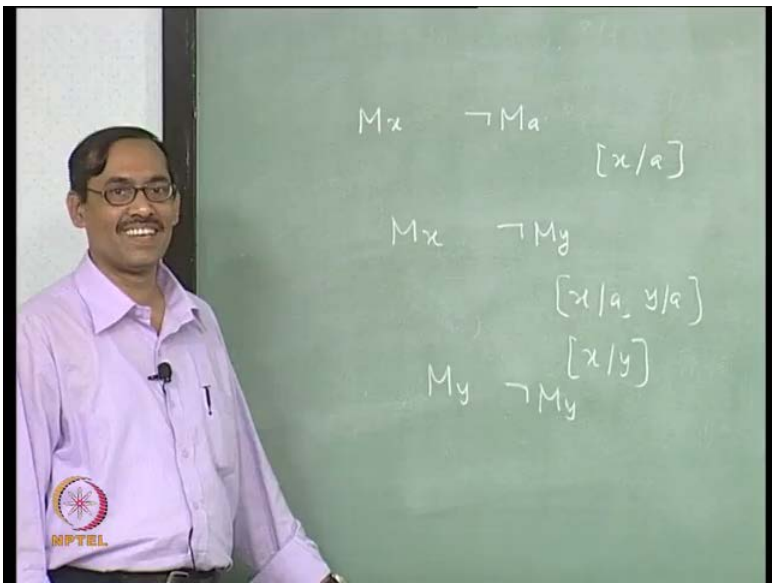
Or, composed with sigma. Are they same? They are different, because x to a, x to b, we found the difference. You can go further, but that is fine. So, we see that sigma composed with theta is not equal to theta composed with sigma. It is just like a permutation. Two permutations; they need not commute; that is what exactly being done here also.

So, what you can see is that if theta, sigma are substitutions, t, X, let us write one only; so t is a term, X is a formula; t operated with theta composed with sigma should be equal to t theta, then, operated sigma. That is your purpose of defining the composition. Similarly, x theta oh sigma should be equal to, x theta operated over sigma, this is how you are convincing, how does it go, whether you need a proof?

Proof is very uninteresting, because it is by induction. Anyway we have nothing else. So, that induction on formulas will be easier because you can just have the induction on the number of free variables in X. We can also limit our formulas to quantifier free formulas, no problem? Because that is the way we will be using. But, none the less, it is true for any formula. So, you can have induction on the number of free variables over X there. Then for t is a bit difficult; whether you can have free variables; but that will not give any structure of t. There, you may need “how t is looking like”, what is the pattern of t inside, how many brackets are there, and so on. That is, you may say on the level of the term t.

So, we will leave the proof here. But you see our definition itself is mechanical, giving a mechanical way how to proceed for the composition of substitutions. We can compose that. Our aim is to use the composition or the substitutions to do resolution process, where you may have M x. Then, another not D x, that is of not M a. M x and not M s should resolve. Naturally, if you choose the substitution x by a, and both of them, M x and not M s, use both of them, same substitution x by a. So, on a there is no change, you would get M a, not M a and there is bottom, that is what we are going to do. So, applying the substitutions for this resolution to work.

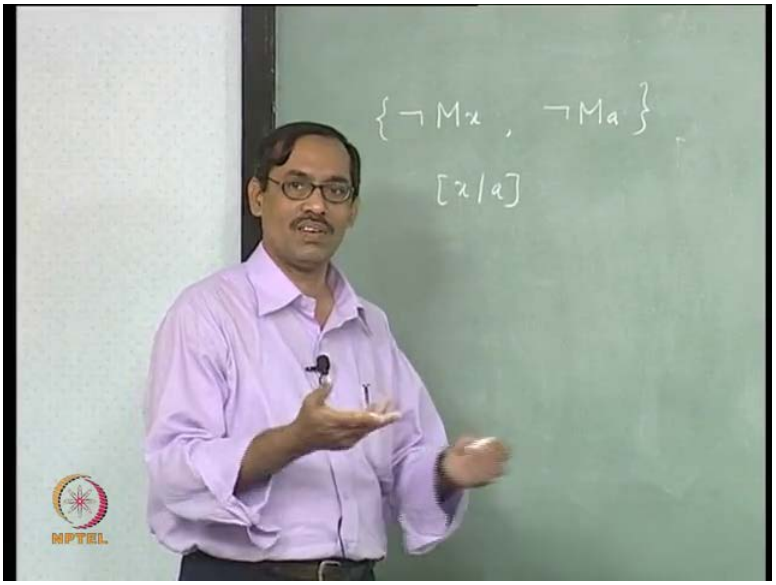
(Refer Slide Time: 25:00)



Now, you see if you start with say M x and not M a, it is easy to see what substitution it is. Because you have M x and you have not M a. Now, you see that our substitutions will be x by a; so as to the substitution so that both of them will be able to resolve. Once this substitution is applied, this becomes M a, this is, remains as not M a, and there itself. Now, the thing is, they may not be that simple; as one variable and one constant, there can be many terms, many variables. Also, a general substitution might be required to be applied. Now, how to get such a substitution? You can guess here easily; like, you want to write one algorithm, how mechanically you will be able to guess or get that substitution, which should be applicable? But there is also a problem: in which way we substitute. You get some way; that may not be very general. For example, I have M of x, I have not M y. I can have one substitution, which is x by a, y by a. With that itself, I can go for the resolution. This becomes M a, this becomes not M a. But then there is one

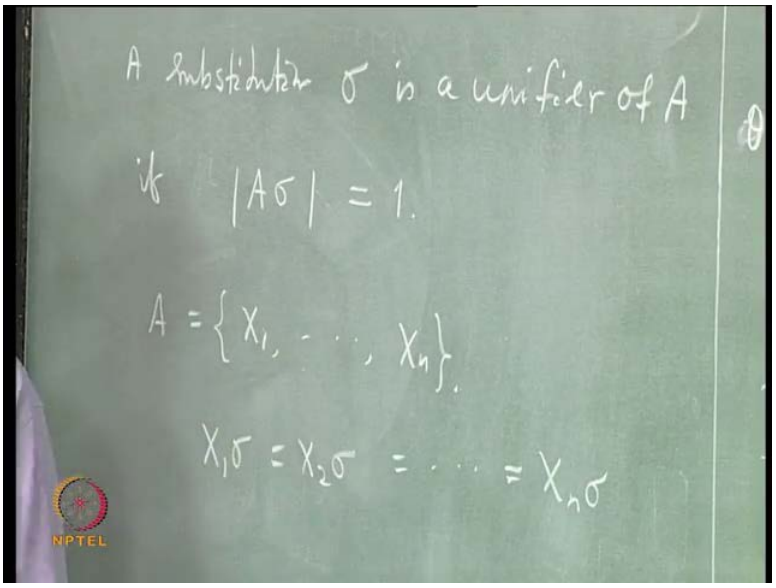
more. I can say x by y, with which I will get m y here, not m y here. They can also resolve. But I would prefer this, because if by this I can have another free variable. I can use the same thing later; this, whatever conclusion I get from this, can be used later. But these two? If I take x by a, y by a, I can never use it. It is gone. The generality will be lost if I choose such a substitution. So, you want a general unifier to do it. They are called the unifiers, or substitutions which will keep this generality preserved, until we need it. This is what we want to do. For this purpose, we will define again some other concept.

(Refer Slide Time: 27:15)



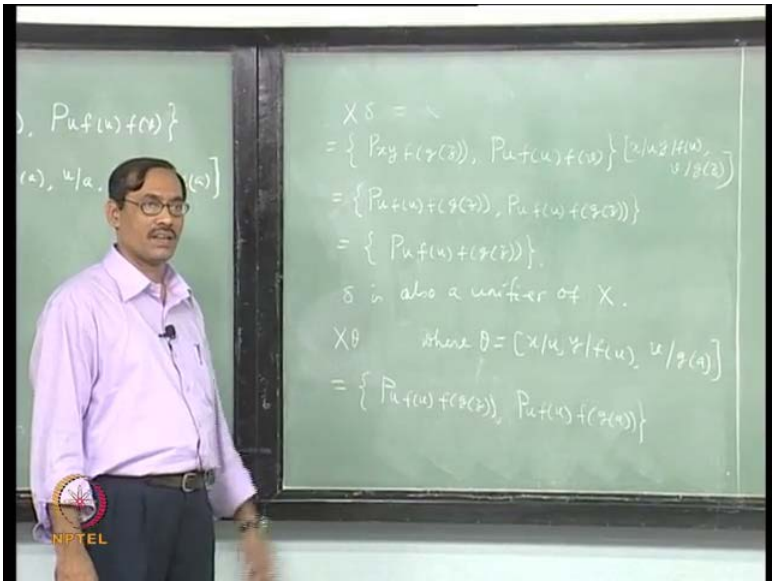
Say, from these two, I would consider not M x, not M a, this set, then I will say that x by a unifies both the literals. Once I apply x by a, both of them become same; that is the concept of unifier. A substitution is a unifier of a set of literals, or even you can say set of clauses, let us take only literals now, if that becomes a single term after the substitution. Now, for the set of formulas you can go, there is no problem. Of course, you need only literals or clauses, still you can apply for general formulas. Let us say for clauses. Let A be a set of clauses. So, your assumption is all these clauses are in DNF, because we are using only SCNF here; so already they are disjunctive clauses, disjunction of literals. Now, you say that a substitution sigma is a unifier of A if A sigma is a singleton; which means, if you have A equal to say X 1 up to X n, I should have X 1 sigma is equal to X 2 sigma is equal to X n sigma. If this happens, then that substitution is called a unifier of the set A. Sometimes, we will say unifier of the clauses X 1 to X n.

(Refer Slide Time: 28:26)



Let us see one unifier. Suppose if you take X equal to, we will start with that formula, P x y f of g f z and then the next is, P u f of u f of v. This is my set of clauses. I take sigma equal to, I suggested, x by a, y by f of a, then u by a, z by a and v by g f a. It is a unifier or not? You want to verify.

(Refer Slide Time: 31:30)



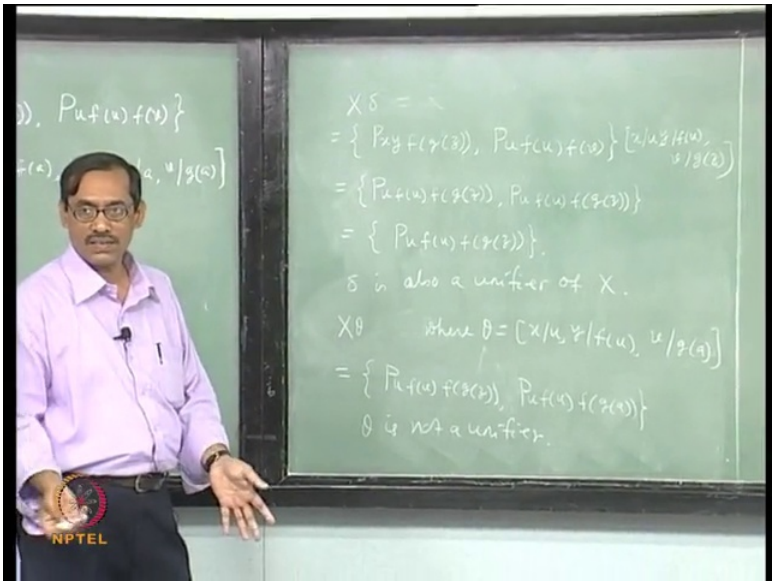
Now, we see P x y f of g of z sigma is applied over that; so this is equal to P, now in sigma x is a y is f of a f is remaining as g is remaining as it is z. That is what it is and

then P u f of u f of v sigma, so this is P, u is a then f a f v is g of a; they are same. So, we see that both of them are same therefore, X sigma is a singleton. So, sigma is a unifier.

What about X delta? This is X, and let us write, equal to P x y f of g f z, P u f of u f of v. Now one a set directly we are applying the substitution. It means, once you have a set sigma, sigma and some theta, will be equal to set of all X theta, so that X belongs to sigma; this is the notation we are using. So, this is equal to, and we have delta; delta, let us take x over here y f of u v g of z, so this is P, now x becomes u, y becomes f of u f of g f there is no z. So, it remains now v is x, u is z, x is u, y is, of u should remain, this is z and then P u as it is, f of u as it is, f of v is f of g of z; so this is singleton. So, delta is also a unifier.

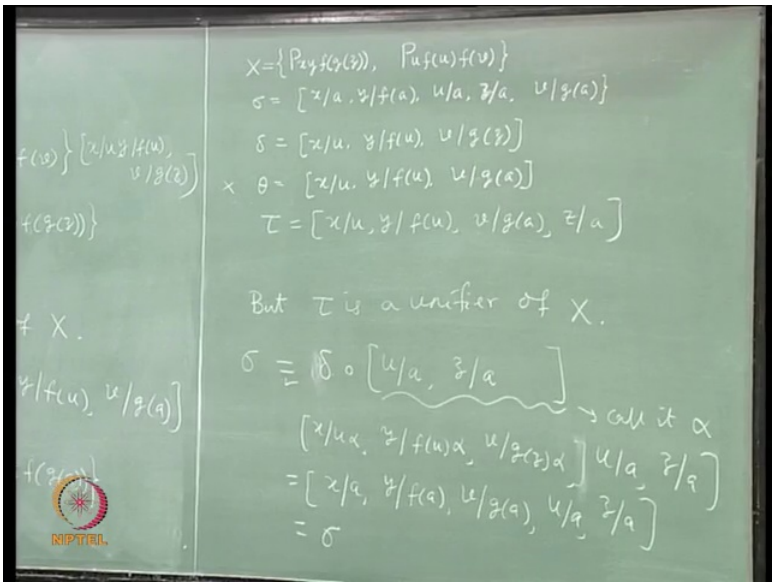
What about theta? Let us try that also. X theta is equal to, now here theta is equal to what? Where, let us write that. Theta is x over u, y f of u, v g of a. The difference between delta and theta is, x, u is same, y by f u same, g f a instead of g of z. So, let us see, x theta is P x is u and y is f of u, and then v is, v is not occurring, it is not occurring there; z again, same, it is a here. So here, what happens, v is replaced by g of a; it is not same, second one is different now. Second one is, P u is, there is no u, so remains f of u, remains, f of v becomes f of g of a, there are not same, so theta is not a unifier. But suppose you have z by a now in theta, then it becomes a unifier.

(Refer Slide Time: 36:04)



So, let us have another, say, tau is equal to x by u, y by f of u, v by g of a, z by a. Now let us verify whether this is also a unifier or not. Once you have z by a, so this z becomes a; that will unify. So, this is, this says theta is not a unifier. But tau is a unifier of X. So unifiers are not unique. That is clear now.

(Refer Slide Time: 36:30)

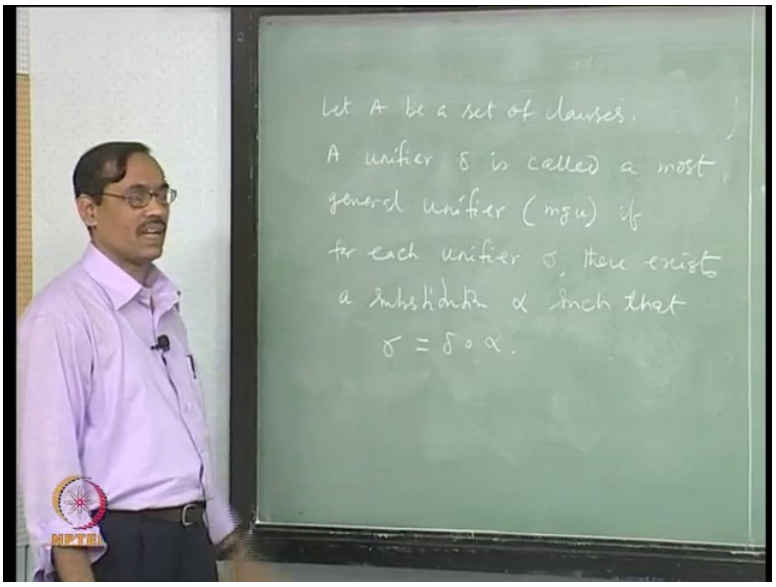


We want a general one; which one of these you choose? Theta is not a unifier. Which one of sigma, delta, tau you choose for a general one? Delta, you would like? Yeah? Why? It gives the variables, but you can give another mechanical way. You can say that all the other two; these are; the two unifiers can be seen as delta composed with another substitution, which is possible. For example, let us say sigma. Is it that sigma equal to delta composed with another substitution?

Let us see. Sigma, x to a. So, u should be going to a, u has not gone anywhere, it is there. u is going to a; I can say u goes to a, here also it will be deleted anyway, in the composition. Then what else? z to a, yeah; is that all? That should. Let us find out whether it is that or not. Now, with delta first, I have to take delta and update it. So, from delta I have to get x y u and this, call it something, call it alpha; now x by u, alpha y by f of u, alpha v by g of z alpha; always consider this set first; what else I should do? I should add to it all these in alpha which are not x y or v. Now, one of them is x y or v; so both of them will be added; so we will add to it u by a, z by a.

Now, this is equal to x by u, alpha is here, so u alpha is a, y by f of u alpha, so that becomes f of a v by g of z alpha, z also becomes a. So, g of a and u by a, z by a; is it same as sigma? Yes, it is same as sigma: x by a, y by f of a, u by a, z by a, u by g of a, this is equal to sigma. So we would prefer delta as our chosen unifier.

(Refer Slide Time: 40:03)



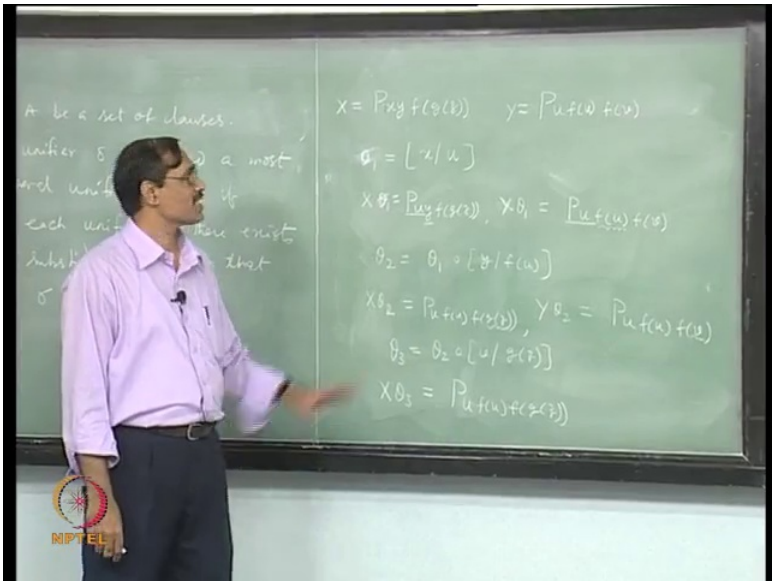
We will give a name to it. Let A be a set of clauses. A substitution, or we can say a unifier directly, unifier delta is called a most general unifier. Let us say mgu, if for each unifier sigma there exists a substitution alpha such that sigma equal to delta composed with alpha. So, delta is a your chosen one. You would, we will be happy with delta because any other time we see some unifiers you can use another substitution to get that back. That is why we will choose this delta. But then most general unifiers are also not unique, so you see that; why? They are not unique, for example, come to this X; this was your X; delta was the most general unifier, as we have seen here; we have not proved yet.

Now what happens, can you choose another most general unifier? You have x by u, y by f of u. Instead of this, suppose I choose u by x, y by f of x. So, ultimately in X, u is figuring out. Now, u will not figure out, x will figure out; that is also a most general unifier, up to renaming of the variables. But there can be others as well; also not only renaming of variables, that is also possible. Here, what we have seen is, most general

unifiers are also not unique. But then we need one most general unifier; that is enough for us; we will continue with that.

So, even if not unique, it is for us, now you have to give a procedure how to get one most general unifier; fine? And then be content with that, whatever we get from the algorithm. Once the algorithm is used, we will say the most general unifier; because whatever the algorithm computes, that is our mgu. For others, we forget. Others can be renaming or substitution can be applied on it to get that. So, let us see the procedure, to get one most general unifier.

(Refer Slide Time: 43:15)



I have a clause P x y f of g of z, I have another clause P u f of u f of v. Now, how do I compute a substitution which will make them same? That is what the most general unifier is. I do not want to take or replace variables by constants unnecessarily. If possible, I keep the variables as it is. So, what I do, I just check both of them; start scanning from the left; you are mechanizing it. So, you should not use any semantics or anything. Now, once I scan, I see that P and P are same, matching, I want to match them, that is all, to make them same. Next, I have x here, I have u there, so one of them should be replaced by the other. Whatever the algorithm will do, it will have one as the first, another as the second. So, first one if it is a variable, second one is a term, that is what it will search for, then substitute x by u, variable will be substituted by the term. If the first one is a variable, then, it does not want to verify whether the second one is a variable or

not, but if first one is a term which is not a variable, it will try to see whether that is a variable; then that will be taken.

So, it will get ultimately one unique, that is what; now I have to find or add to it x by u; this is added. But once this is added, there can be other places where x is there; it is not here. Fortunately, there are other places where x is, see, then there will be a matching problem. Here again. But once you apply this x by u, they, all those x's have been removed. There are really u. When you apply the substitution you will be getting u not x, so better we should not go hurriedly to find out the next mismatch. You should apply the substitution on the formulas, then go for the mismatch.

So, now I have, say, call it theta 1, which is this. I compute X and Y, let us say; so I compute X theta 1 and I compute Y theta 1. This becomes P u y f of g of z and this becomes P u f of u f of v, that is the effect of theta 1. There is no x there. So it is reproduced as it is. Now again I match. Already I know up to this, it is matching. I do not need to match, to make it efficient; but you can just write: match again. Now, again a mismatch is occurring: y and here is term f of u. So I take theta 2 equal to theta 1 composed with y by f of u. So, ultimately what I have to do is, this y over f of u will be applied on earlier computed formulas; so I get X theta 1 apply theta two to this, and again Y theta 2. That means, y by f of u will be applied on X theta 1. Similarly on Y theta 1. So apply. You get: P u y becomes f of u, then f of g f z, and Y theta 2 is P u f of u f of v. Again, go for matching. P matches, u matches, f of u matches, f of matching brackets matching, so g of z equal to v. That means, I take theta 3 equal to theta 2 composed with v by g of z; that is my most general unifier. By this time my two literals have been unified to one literal which is P u f of u f of g of z; that is the formula.

This is basically the algorithm of computing a most general unifier. You can write the algorithm; but there is something you have to say in the beginning, when they will not match. For example, we have started conveniently with P. Suppose this starts with P, this starts with Q. You cannot get a unifier because predicates cannot be made same; only variables and terms can be, right? So, this type of things you have to write in the beginning. First you check for the not, you have the whole set, for that you are computing the most general unifier.

Here, the set comprising two literals X comma Y, there can be clauses also, fine? Now for the whole set you have to start. Take any clause or any literal rather, rather than the clauses. Let us come to literals, you have to match the literals. Now, in that literals, is a set of literal, let us say; first thing is, if one is having not symbol the other is not having not symbol, then unifiers do not exist. One is starting with not, another one is not having not. So, mismatch will be there; nothing can be done. So, it is not of this and it is P u as it is, then they cannot be matching. Next thing you say, if they do not start with same predicate, then they do not match; they will never have any unifier. Now come to predicates; that is, all of them should be starting with the same predicate or starting with negation, all of them, and then predicate. Then, only you come for the next step, for matching; then find the first mismatch. If the mismatch is for something else, two different function symbols, then, there cannot be a unifier. One is f of u, another is g of z, nothing can be done. So, the mismatch should be: one is a variable, other is a term; both can be variables, because variables are terms also; at least one of them should be a variable. Then take the first variable. The other term for the substitution, then compose. Go to the next step, right? Computing in loops, that is how most general unifier will be computed.