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

Lecture - 40
FC, Semidecidability of FL, and Tableau

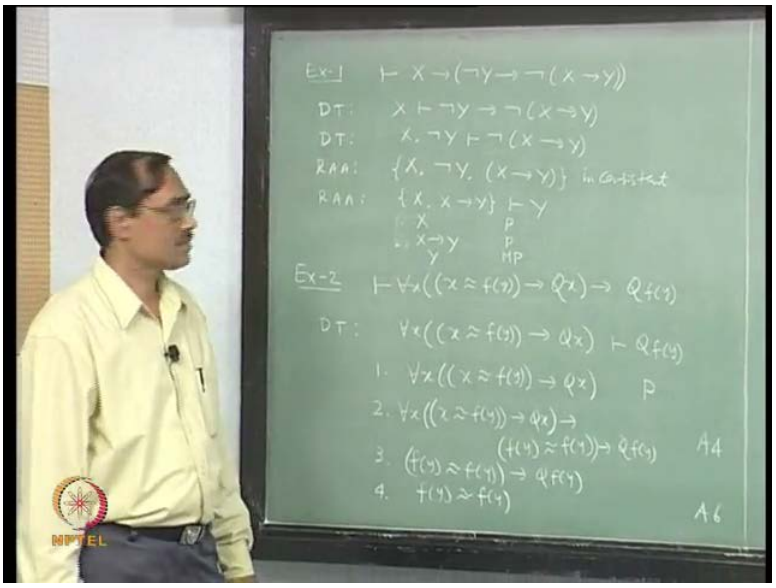
So, we were discussing the axiomatic system FC for the first-order logic. Then along with the axioms we had now two inference rules unlike your PC; PC had only one. The same we have kept as MP plus another for universal generalization. And in the axioms we have all the three axioms of PC plus there are some more. So, two more for the quantifier and two more for the equality predicate, right? That is how we have started; then we proved two theorems: monotonicity and deduction theorem. Then just like your PC you can also formulate reductio ad absurdum, the same formulation, and the same proof, right.

But, there is nothing much there; and just we tell them what it is. You have one consequence to be proved, sigma entails X. In that case what you do is, add not X as a new premise and then show that sigma union not X is inconsistent. And this process is reversible: if sigma union not X is inconsistent, then you say that sigma entails X; fine? This is your reductio ad absurdum, basically. Then, let us apply all those meta theorems in proving some of the theorems. In fact we will not prove them as theorems, because we are not going to give a proof of that directly.

That means, we will show that it is provable, fine. Which means, you take some formula. And then may be using some reductio ad absurdum or deduction theorem, you transfer it to some other consequence. Then give a proof of the consequence, because of the meta theorems the original formula is provable as a theorem; that will be the approach. So, let us see Example 1, which is really propositional. X implies not Y implies not of X implies Y. There is no quantifier involved directly though they are now formulas from FC, not propositions necessarily. Begin. For any formulas you have, the reductio ad absurdum and the deduction theorem, you can apply, and get it done.

One easy way to look at it is, see you have the law of contraposition; if you can prove X implies X implies Y implies Y, then it is done, if you use law of contraposition. Even without using it you can do, because you have RAA. Let us try that way; say, first is by deduction theorem.

(Refer Slide Time: 02:59)



It is enough to prove X entails not Y implies not of X implies Y, fine. Then you say again, another application of deduction theorem says X not Y entails not of X implies Y, right. Then with reductio ad absurdum you show X not Y and X implies Y that is inconsistent. Then you say, again by reductio ad absurdum X, X implies Y entails Y, right. Choose this one as your formula; which has come from the consequence. We look from this to this; you will see easily not Y is added; and that is inconsistent. And there is nothing to prove; it is your MP, right; for which you can give a proof, say, X implies Y. Therefore, Y or just give: this a premise, this a premise, this is modus ponens; that is all; is it clear? So, this contraposition has been proved inside it; these are not new to you; ha, the same technique of PC, that is what we have done; is it clear? We proceed to next example.

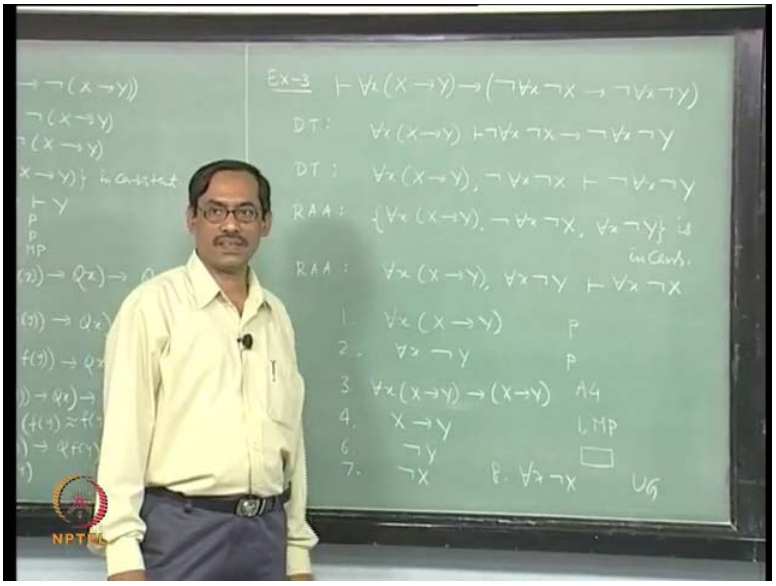
Here, we have for each x x equal to f of y implies Q x, then you show that Q f of y. Again, deduction theorem might be handy. Let us say, by deduction theorem, you just show for each x x equal to f of y implies Q x entails Q f of y, right. Now this can be shown. It is easy to show here. Just what you have to do is, take a suitable instantiation of this universally quantified variable. You start with our premise for each x x equal to f of y implies Q of x; that is a premise. Next one axiom for instantiation. So, x equal to f of y implies Q x implies f of y equal to f of y implies Q of f of y right. Is that ok? For each x x implies x x by t, is that right? This gives you, which axiom; f of; next, use MP to get f of y equal to f of y implies Q f of y. Next, use another axiom: equality axiom; and then

modus ponens. We will leave it there; this is which axiom? Again, just to remember them.

Three were from PC, two were from universal, next one, A6, and then an application of MP finishes it; clear? Let us try next example.

For each x X implies Y implies not for each x not X which is really your there is x X. And you are given in that form; that is also fine. You can use the definition and reduce it to this one. Finally, this is to be proved, so this will say for each x X implies Y implies there is x X implies there is x Y, right?

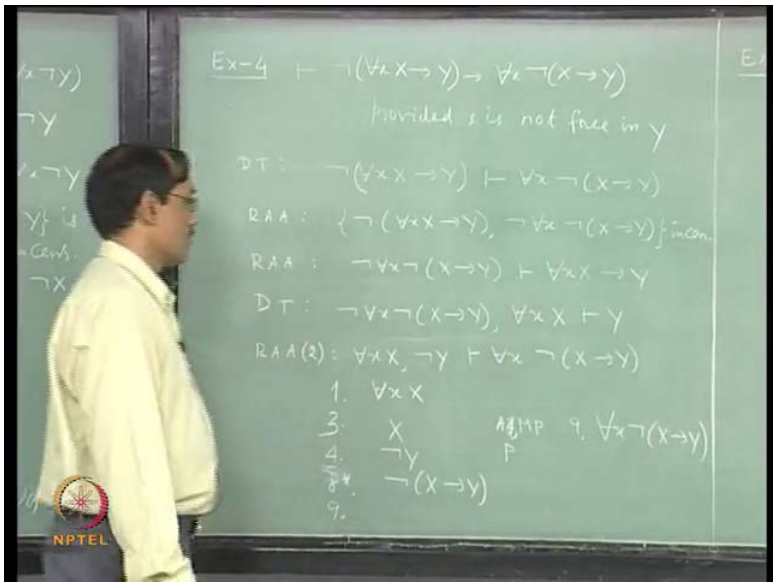
(Refer Slide Time: 07:48)



Again, we use deduction theorem. This says, for each x X implies Y entails not for each x not X implies not for each X not Y. Here itself you can use contraposition, right; by contraposition, it is enough. But, contraposition, we cannot use here; it has to be used inside the proof again. Only metatheorems, we can use outside the proofs. Then, let us use again deduction theorem. This says for each x X implies Y and not for each x not X entails not for each x not Y. Next use RAA, because this not will be difficult to use as a premise; fine. So, you take for each x X implies Y not for each x not X along with for each x not Y is inconsistent. Once more you use RAA to take that negation to the other side. You say for each x X implies Y for each x not Y entails for each x not X. Now you see contraposition has been used through RAA. You should be able to prove this. It does not look to be more simplified than this. So, let us try a proof.

For each x X implies Y, premise; for each x not Y, premise. Next, well we can just say X implies Y; and similarly. In fact we did not go in one stage; axioms has to be used, and then MP has to be used; it has to be brought; right. So, what do we do is, for each x X implies Y implies X implies Y, that is axiom four. Then MP, which will give you X implies Y; two steps will be required; right. Similarly, another two steps will give you not Y; there you will be using PC. So, this comes from, again same Y; you can fill it up. Next, seven will be, not X; right. Then, universally generalized; for each x not X; is it ok? So, eighth line will be for each x not X; UG. All that you have to see is, x is not free in the premises; which is not; in no premise it is free. So we will use that; for whatever it is, x is free, x is not free in all the premises you used till now. So, for each x not X; clear? Now, next one.

(Refer Slide Time: 11:44)



By deduction, we have to not for each x X implies Y entails for each x not of X implies Y, right. Now, some mental block. How to use this not? Well, we use RAA; transfer it to the other side; see what happens. This gives, by RAA, not for each x X implies Y and then not of this; this is inconsistent, right. Then again RAA, which gives not for each x not X implies Y entails for each x X implies Y; fine. Then use deduction theorem, which says, not for each x not of X implies Y for each x X entails Y; clear? Again, this not is, again, with for all.

Student: Take Y

So you have to take it inside.

Student: Not Y.

You can take not Y also. RAA will give for each x X, not Y entails for each x not of X implies Y twice; ha, we have used it twice, right. Not Y comes here; which is inconsistent, right. Then, I take it to the other side by RAA again. So, for each x X not Y entails this; clear. Now, probably this will be easier to show. One, you have for each x X; then second, rather third line, use axiom four; for each x X implies X; next use modus ponens to get X; both will be used; is that ok?

Second line will be for each x X implies X; that is your axiom four. Then modus ponens on this two, to get X. After you get X, you have not Y, premise. Now, what is your target? For each x not of X is there; so this can be obtained by universal generalization: UG. You want to get this. Now, that is propositional; which is your Example 1, right? So, use Example 1 as a theorem, previously proved theorem; then use modus ponens with X, you get not Y implies not of X implies Y. Again, modus ponens with not Y, right? You get not of X implies Y.

Student: What is this, we din not know one example.

Hm, you proved it.

Student: Sir in exam how are we.

You will have to prove it; no?

Student: We will.

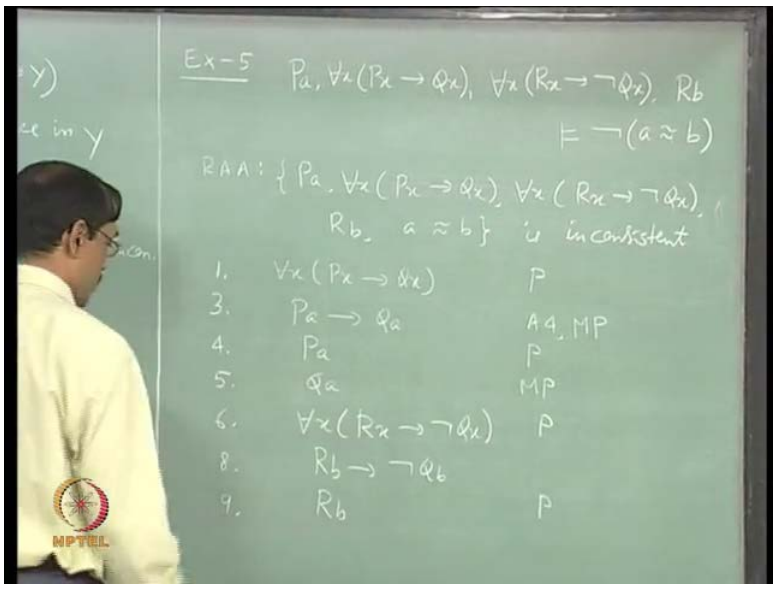
So, once you see that X and not Y gives not of X implies Y; right. That is really a PC theorem; it is a PC consequence; so you can always bluff it as PC. And then continue, fine. But then it need not be a theorem; so better show it, is that ok? But, it can be done some other way; your bluff might be correct.

See, there is a, there is one nice thing. How to prove the trigonometrical identities? There is a nice algorithm to prove them; which we never know in our high schools. Otherwise, you would have scored hundred percent. The algorithm is, suppose there is a trigonometrical identity, left side equal to right side. You know it is an identity, right. It

is given in the book, so it must be identity. So, what you do is, start from left side, writing whatever identity you know; go on applying it; and here from the bottom you start from the right side; whatever you know, go on filling it; leave it there, right. And it is not wrong; whatever you get in between that is correct identity. Is that ok? So it depends on how much you have to spare, in the paper.

The same way you can leave it here. You know that X, not Y should give you not of X implies Y, semantically. Because X, not Y both are there. It is equivalent to X and not Y and not of X implies Y is equivalent to X and not Y. So, it should be provable, because it is complete system; PC is complete. So, there is a proof of it; right. But what the proof exactly is, we do not know. Leave it as a theorem, PC theorem. But better to prove it. And we have proved in Example 1. That means, it will give, after some steps. Let me write it. Some eight star. I do not know whether its eighth or ninth, ha. There, you get not of X implies Y; then we use UG; that gives you for each x not of X implies Y.

(Refer Slide Time: 18:27)

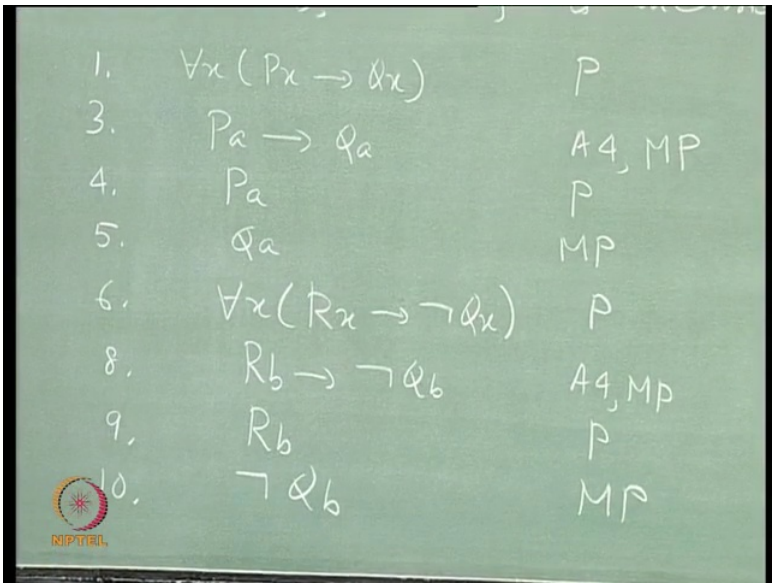


Then, let us see this. See, with not of a equal be b, it is not easy to get, right? So, use RAA from the beginning. By RAA, we try to show P a, for each x P x implies Q x, for each x R x implies not Q x, and R b, and a equal to b, is inconsistent. Now, with a equal to b, it will be possible; right? But, there are constants a and b. So we do not know which one we will instantiate. If you instantiate wrongly you may not get that, fine. We do not

know exactly where we will get not P a or not R b or even or Q a or not Q b. Something which will bring the inconsistency, that is not very clear.

Now, that always happens whenever there are instantiations. Earlier what happened? Earlier cases of RAA, there was no instantiation; they were almost propositional; everything should be possible. But, now there is instantiation and a equal to b; so I do not know which one I will be able to derive: not Pa, I will be able to derive or not R b. Which will contradict Q a not Q a, Q b not Q b; I do not know which one. So, let us keep it in this one and continue towards deriving inconsistency, right. Now, you start from, say for each x P x implies Q x; this is a premise.

(Refer Slide Time: 22:17)



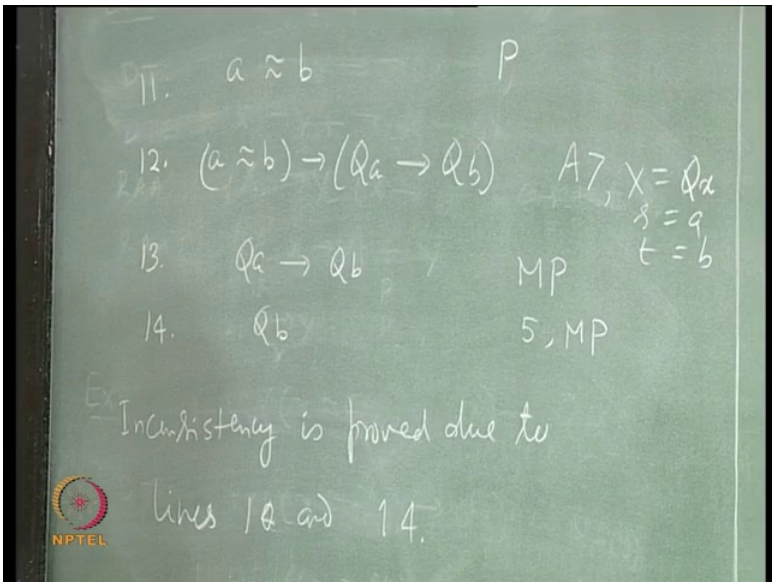
So, third line, second line, I want to use my P a, right. Since, I want to use my P a. I want to instantiate it. So, write in the second line: for each x P x implies Q x implies P a implies Q a right. It is axiom four, A4. Then use modes ponens to get in the third line P a implies Q a, right. So, A4 and modes ponens. Then I can use P a as the premise and get by modes ponens Q a, right. But, here I have R a, not Q x. Anyway I can use RAA still, because I have already Q a, or for R b; I can use b also. Any one of the instantiations will do; which one we will go for, a or b?

Student: b

With b, fine. Then sixth one, introduce the premise for each x R x implies not Q x. Then again, eighth line will be R b implies not Q b. Then you have ninth line as R b. That is a premise. So, seventh line, I have left it, there is axiom four. So, it should have justification here as A4 and M P, fine. You have to fill it up; do not leave it like this.Then ninth is R b; I get in the tenth line, not Q b, by modes ponens, fine. Then a equal to b has to be used. So, eleventh line is, a equal to b, premise. Twelveth.

Student: Q x.

(Refer Slide Time: 22:44)

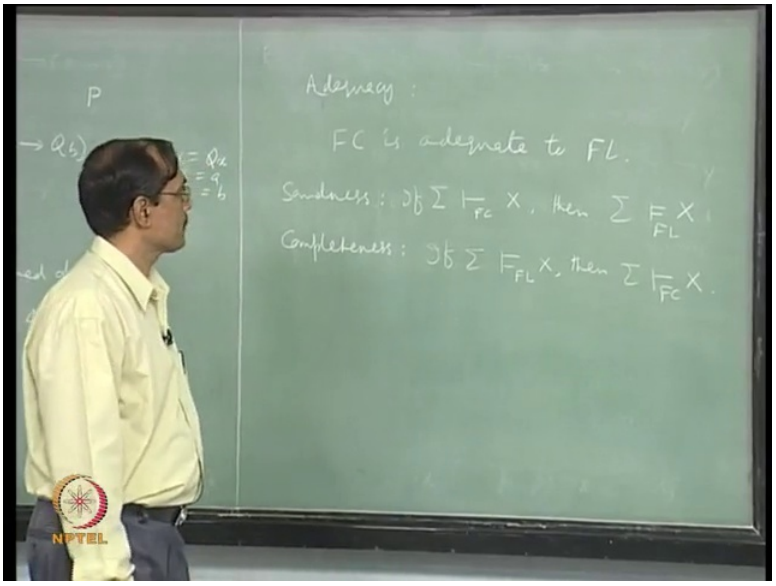


We have already Q a; we have not Q b, a equal to b; that should give the contradiction, right. That should bring an inconsistency. We have to use A7, a equal to b implies Q a implies Q b; X, s equal to t, X x by s, X x by t. So, our X is Q x, fine. This is A7. Then modes ponens. That gives Q a implies Q b; and we have already Q a. Q a is which line, fifth line. So, you just have Q b. That is the end of the proof, right? You have to write a line. Where is the inconsistency? So, inconsistency is proved due to formulas in lines: not Q b is in 10, and 14 clear? It is easy now, how the proofs will be going.

The next concern will be, whether this system really captures FC or not? It does capture, and we will not do the proofs. Now, proof again follows the same line as that of PC to PL, right. It is sound; it is easy to see. Because all the inference rules are the valid consequences in FL, and all the axioms are really valid formulas in FL. Therefore, it is sound, by induction. Soundness is easy; completeness will be a bit difficult; because here

you have to instantiate with all the terms. So, there is potentially infinite number of terms. For all instantiations you have to consider the formulas; whenever there is some for each x X you have to really consider X substituted by all possible terms. All those formulas have to be extended in order to get one maximally consistent set; because there is our approach to go to a maximally consistent set. To get that, you have to really add all these instantiations. That makes it difficult; but then it can be done. Because all of them are countable, so they can be enumerated the same procedure still can be used.

(Refer Slide Time: 25:49)



We will not prove that; but will mention. FC is adequate to FL. Which really means, you have two results: soundness and completeness. Soundness you can say sigma entails in FC X, then sigma semantically entails X in FL; that is your soundness. And completeness is the converse of it. If sigma semantically entails in FL X, then sigma entails in FC X, for any set of formulas sigma and any formula X. Which you can write also with reductio ad absurdum in a different way, right. Once you show that it is adequate you can prove something else; like, can you generate all the theorems in FL, all the valid formulas in FL.

Suppose first-order logic is there; you know what are the formulas; you have the laws. But, they are only the basic laws; then there are so many others, infinitely many valid formulas are there. Can you generate all the valid formulas by using the laws? That is no completeness there; we are in a semantic domain. Now, well since FC is complete, it is

enough to say by using the axioms and the rules of inference of FC. Can you generate all the valid, provable formulas? Yes, because we have the definition of a theorem, definition of a proof. So, what you have to do is, just feed all these things to a machine and program. Write a program; just go on generating by following the rules, whatever it is, right. We are not asking you to prove a particular theorem; we are just asking it to go on generating all possible theorems, right. So, it generates one by one, it gives you right; so that means the set of all theorems of FC is recursively enumerable. There is a program to generate all these things; it is an output of a Turing machine, right. So, it is recursively enumerable since it captures all the valid formulas in FL.

So, all the valid formulas in FL, set of all that, is also recursively enumerable, right. There is a program to generate all the valid formulas in FL; the program is just programming of PC, implementation of PC, FC, right? Is it clear? It is the next strong question we have to ask, because it does not say when you will get your theorem, when you will get your valid formula to be proved. It just goes on generating on its own way, the way you have written the algorithm, right. But, problem is suppose you want to have a proof of something like this, right. That is a targeted one. It did not have any target till now. You just say, go on generating all the valid formulas. Now, you say whether this formula is valid or not? Give a particular one, that algorithm may not work, right? If it is, then it will work right; so that is the problem.

Suppose you know that something, some formula is valid. Now, your algorithm generates all the valid formulas, so at some finite time it will be generated. And once it is generated it stops there, you can write another module on it, right. Use that as a module, and once this is generated, stop there, fine. So it stops it gives “yes I have got a proof of it”. That means, if your conjecture is true, you get a proof of it. If it is false?

Student: No idea

No idea. It may not stop at all; it can run on infinitely, right? So that is the idea of recursive enumerability, fine. If there is a targeted theorem to be proved, we do not know when we will get the proof, but we will get it if it is a theorem. If it is not a theorem, it does not say anything, the algorithm need not terminate, fine. It is one side of our strong question, that given any formula, can you decide whether it has a proof or it does not have a proof? It only answers it partially, right. And the undecidability theorem for the

first order logic says that, the answer to the next question is negative, right. It is not decidable at all. It is recursively enumerable, but not decidable. Means, given some valid formula you do not know whether it is a theorem or not. Once it is a theorem you have a proof of it. Once you have the undecideability result you may say this recursive enumerability is a semidecidability result rather, because one part of it is decidable, the other part is not decidable. That is why they are also called semidecidable problems. So it is giving some more information to recursive enumerability. It is recursively enumerable, but not decidable; they are called semidecidable. Undecidability, we have not yet proved; we will see that once time comes. But, recursive enumerability, it gives; that is all.

What about your Herbrand interpretations? For example, suppose you do not use FC, you just go to the Herbrand expansions. What happens there, you are given with one formula, FL formula; then you compute its Herbrand expansion. Now, it is potentially infinite set; but infinite set of propositions. Suppose it is unsatisfiable. We leave the question of validity, because by duality it is enough to consider unsatisfiability. Now, suppose the formula given is unsatisfiable. Then that means, its Herbrand expansion is unsatisfiable. Once it is unsatisfiable, by compactness theorem for propositional logic, there is a finite set which is unsatisfiable, right? Now, let us write an algorithm how to prove unsatisfiability from the Herbrand expansion.

We know, there is a finite subset; so start first one, is it unsatisfiable? Yes, stop there. If not, take the second one, ask the same question, is it unsatisfiable? If unsatisfiable, stop; otherwise add the third one, continue. If it is really unsatisfiable, it will stop somewhere. And that is one finite set, you have got, which is unsatisfiable. Compactness theorem guarantees that there is one finite subset, once it is unsatisfiable, right? That is why the algorithm will terminate; is that right? But, if it is satisfiable, again, algorithm will not terminate; it will not be able to show that it is not unsatisfiable, right. Again the same thing is coming there. Unsatisfiable sentences or unsatisfiable formulas in FL are recursively enumerable. So, you can get the same result from the Herbrand expansions also. But it does not say anything about the other side, undecidability, fine.

Suppose that you need some more results like that; you start with a Turing machine, for example. Now, any Turing machine, its operation you can stimulate by one first-order sentence. Let us say first-order sentences. Just the way we are translating some natural

language arguments. What you do there? Say, all horses are animals; therefore, legs of horses are legs of animals. In this argument what you do is, you take anybody is a horse H x, anybody is an animal A x; go on symbolizing it. You get one consequence in first-order logic, with your translation of those English sentences or English phrases to predicates. And similarly definite descriptions to functions and so on. With that you do the translation. Now, suppose it is a Turing machine. Then you have to see what are its fundamental propositions or fundamental predicates. One is, a Turing machine can go to the right, come to the left, and so on.

Suppose you number the tape, one, two, three, four, and so on, start from the finite left, left side cut, to right side only it is just extended, not both the sides; that always can be done. Starting from there, (even if you take both the sides you can again manage with negative integers, right) starting from zero. So, let us try one to infinity. Now, what do you do? If it is in some particular place, then you say where the Turing machine is. So, you say W5 means, now it is scanning the fifth square. Next what you do, if it is just fifth square and some transition is applicable, which says that “go to the right by reading this symbol being in this state” right. Now, you have to say what is the state of the machine, what symbol it is scanning. So, it is in W and it is scanning so and so symbol. So, for each symbol you have predicate S of something, what is being scanned. In which state is it, you write T of something, so T of that. Now, you say W have this, S of this, t of this implies? If there is a transition with that thing, so, you have to introduce a predicate “if the transition is such and such”. Then action of the machine means after the next step the machine will be in this configuration, right. So, it will have this state; it will be scanning this square, and so on. This can be again represented by the predicates, fine. Each step of the Turing machine can be represented by a predicate, by a formula. If such and such predicates are satisfied then such and such predicates are satisfied, right. That is how you will be formalizing the whole Turing machine.

Once the Turing machine, its functions are given, you can just formalize, get a consequence. Now, you ask the question whether this Turing machine will halt or not. So, that means the halting state you have to translate now, right? It is in particular state h, that is again another predicate with h as the argument. So, halting of a Turing machine can be translated. Therefore, you have a consequence: this Turing machine entails halting state, right? So this translation is done now. Once it is done, that means halting

problem for Turing machines is translated as a consequence. But, halting problem is undecidable; so consequences will be undecidable. Particularly, that consequence, that set of consequences what you get from the Turing machines will always be undecidable, fine. So, first-order logic is undecidable.

In fact, this is the main thing why Turing started his Turing machines, to prove that first-order logic is undecidable. That is how we got the computers; it came from logic. So, that is the undecidability result of first-order logic. There is another way of doing it like you may be using so many predicates and other things for translating the Turing machines. You do not need so much; you may need only one binary predicate. If there exists a binary predicate in your language, first-order logic language, it is enough to give undecidability result. In contrast, if you have a language where only the monadic predicates are there, then it is not undecidable; it is decidable.

For example, if you have all the, all your consequences of this type where you look at the predicates, there is one predicate P which is unary another predicate Q which is also unary. Another predicate R which is also unary; all these are unary predicates. So, suppose you get all consequences which use only unary predicates no functions involved, right. That is your Aristotelian logic, monadic logic of Aristotle, like your Aristotelian sentences; now all men are mortal, Socrates is a man, therefore Socrates is mortal. And all those sentences, they are all monadic first-order logic sentences. Every predicate there is unary; it is also called monadic, unary or monadic. (Similarly, binary or dyadic there are some different languages.) Now, what happens, if that is so, then it is decidable. But, if there is at least one binary predicate or one function symbol, then it will not be decidable, fine.

See its traces can be there in the Herbrand expansions. In the Herbrand expansion. If you have one function symbol f, then we get one infinite domain. So, its traces found there. That, probably that will be undecidable; that can be proved to be undecidable. If there is at least one function symbol or there is at least one binary predicate, then also you can show that it is undecidable.

We are not proving that strong result also. But monadic predicates, if the language only concerns monadic predicates or unary predicates, it is decidable; how do you show it? How do you show that if it is only monadic predicates, then it is decidable? FC

arguments do not show it, right? FC says it is recursively enumerable, fine. What about undecidability or decidability? Herbrand expansion will be finite, right? There is no function symbol, is it not? Suppose like this, you have a. There is a constant b, there is another constant. So you make the Herbrand expansion. What about the equality predicate?

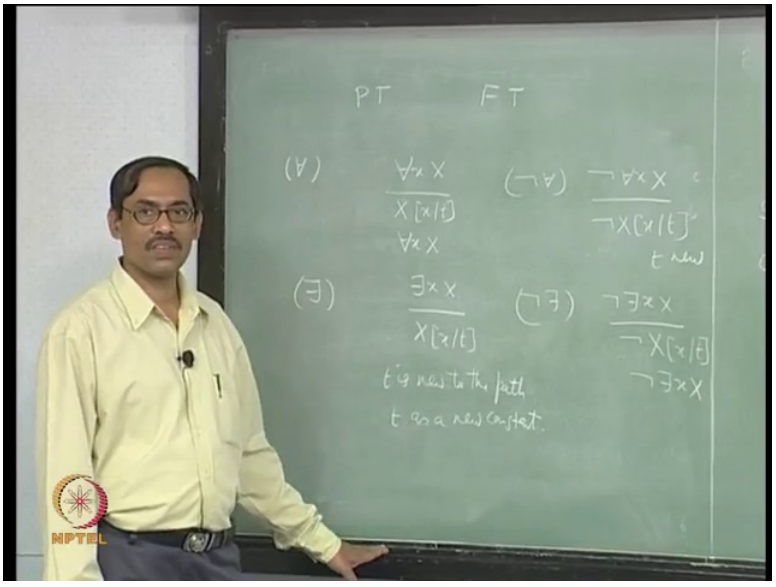
Student**:** Sir, binary predicate.

Hm. It is binary, that will create problem. So, in Aristotle's logic equality predicate is also not there, right. So, there is no problem. You get only a finite number of constants, maximum, in any consequence. So, you have got a domain consisting of those constants only. In that domain, you do it; transfer to Herbrand expansion, that is a finite set. So, it is decidable. Easy from the Herbrand expansions, right. Also the algorithm was given by Aristotle himself; there is another way of doing it, anyway we do not need it now, right. This is the story about decidability results. With this we stop our discussion on FC. We go to another proof method. So, what is our next proof method to discuss?

Student**:** Analytic tableau.

Analytic tableau, right. That is what we have not yet discussed. You have propositional analytic tableau and now we want first-order analytic tableau.

(Refer Slide Time: 42:06)



Again, it is an extension of propositional analytic tableau, all the connective rules are there, whatever we had earlier. Now, we need not confine ourselves to implies, for all, everything can be tackled there at a time, just like your connectives. All the connectives, even top, bottom symbols are there in PT. To extend it we need some more rules for the first-order logic, specific to it. One is, your for each rule, this says, for each x X you can derive X x by t. But then we add something else, which says that the same thing can again be used, because with this one instantiation you may not be able to prove it, you may need some more instantiations. In practice when we use it will not write this, but will remember we used it. In contrast, you have there exists. You can instantiate only once; after that you cannot use again. We will have there is rule, which says, there is x X, so get X x by t. It is not repeated; you cannot use it once more; but what is this term t?

Student: Is that particular generated form of all the premises it should be.

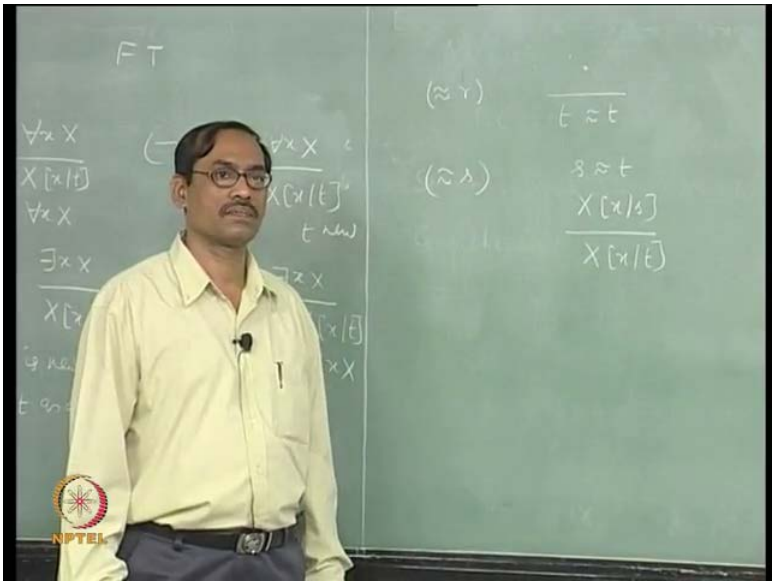
No. Suppose you have already Pc in the premises; you cannot write again c, right. In some context, in mathematics, you are doing some problem. You say there exists one number between zero and one, having such and such property; you say let that be alpha. Now, again you find there exists another number within zero and one having some other properties; you cannot say, let alpha be that, right? It may not be. It can be some other thing. So, that is what exactly done. This t should be a new term to the branch, to the path; whole path from the root to that place where it is being applied. It need not be new to the whole tableau till now generated. Because all the tableaux are generated only path-wise depth-wise.

We are doing it breadth-wise because of our convenience. By definition, tableau is only depth-wise. It concerns only through the path, nowhere else. When it is closed, we go to the next path; that is the way it is being implemented. What we do, we say that t is new to the path, new term in the path. We will again say what is the meaning of this new. Right now, we will remember this t, means it is constant; let us say it is a constant right. It is a new constant; that will simplify. It will need some experience to go for a generalization, where you can use terms also, not only constants, any closed terms any other terms even you can use.

Student: Variables or not variables

We will see that. There are some other constants, we will see slowly. Let us think of this t as a new constant for the time being, fine. Similarly, we will have rules for not for all. That will say not for each x X, you can get not X x by t; again t new. It is really existential, right. Not for each is really there exists not X. So, we will not write, it is equivalent to there exist not X, therefore this a tableau rule. It will proceed this way. Then we have not there is, which says not there is x X, from which you get not X x by t, for any term t. And also repetition of the rule, repetition of the formula; it can be used once more. So, these two are really universal rules, these two are existential rules.

(Refer Slide Time: 46:40)



And there are two more for the equality. In equality, you have reflexive property of equality, which says anywhere you can add t equal to t, just as in resolution. It is like an axiom; anywhere you can add it. And there is substitutivity, which says, if you have X, ok; let us start from equality; s equal to t, X x by s, then infer X x by t; fine. Here, it needs two, just like your MP; if both the things occur in the path, then you can extend the path by adding X x by t. So, these are all the rules, extra rules for FT comparing with PT.

Then, in same way we will be proceeding as in PT, extend the tableau path-wise. Then once you get negation and that formula itself, both in the path, close the path, right? If the whole tableau is closed, then the formulas introduced at the root are inconsistent. The set of all those formulas is inconsistent. So, you are proving only inconsistency; nothing

else. Then to prove sigma entails w; we show sigma union not w is inconsistent; RAA is a definition now.