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

Lecture - 19
Compactness & Analytic Tableau

So, let us start with one consistent set of proposition sigma. Then we could extend it to sigma prime, which we proved to be maximally consistent. This fact, that a consistent set can be extended to a maximally consistent set is called Lindenbaum Lemma. That may not happen in every system. In this system it happens. That means, Lindenbaum Lemma holds in PC. Next, we have proved some of the properties of sigma prime, this maximally consistent set, it just, essentially, capture the connectives. So, our last four properties, one for negation that either p or not p belongs to sigma prime, for any proposition p, and then the other three for the implications.

Those properties, if a set, holds; in a set if those three-four properties hold, then we would say that the set is a Hintikka set. There are many technicalities here. That means, we have proved that there exists a Hintikka set, which happens to be your maximally consistent set sigma prime. We are almost over in proving the completeness of PC. The formulation was if sigma is consistent then it should be satisfiable, right?

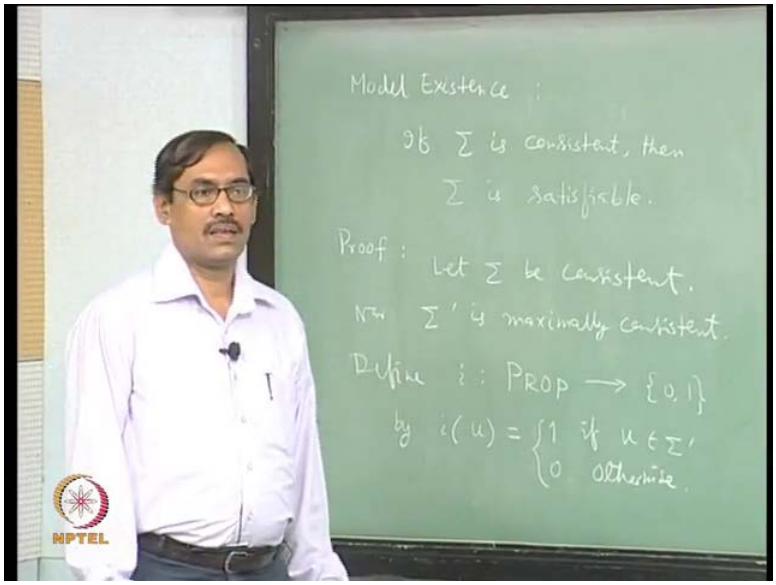
What about sigma prime? Sigma prime is consistent. Can we show that it is satisfiable? In fact, that is the technique, that is why we had extended sigma to sigma prime, and that sentence, that statement says that if sigma is consistent, it is also satisfiable; this is called the Model Existence Theorem. There exists a model for a consistent set. This is what we want to derive next.

This says, if sigma is consistent, then it has a model. I will say that then sigma is satisfiable. Its proof is easy now. What we have to do is, find a model for sigma. Here only we are connecting PC to PL, consistency is a PC concept, coming out of proofs, this depends on proofs, and satisfiablity comes from the semantics, truth and falsity; here we are connecting now. So,

What we do is, let sigma be consistent; then consider sigma prime. Now, sigma prime is maximally consistent and it, along with it, it satisfies all those properties. What happens, you define one function, define some function i from the set of all propositions. Now, in the set of

all propositions, we do not have those connectives: and, or, if and only if, and then we do not have top and bottom also, right? There is no, not these symbols in PC, that is our PROP now, which is appropriate to PC.

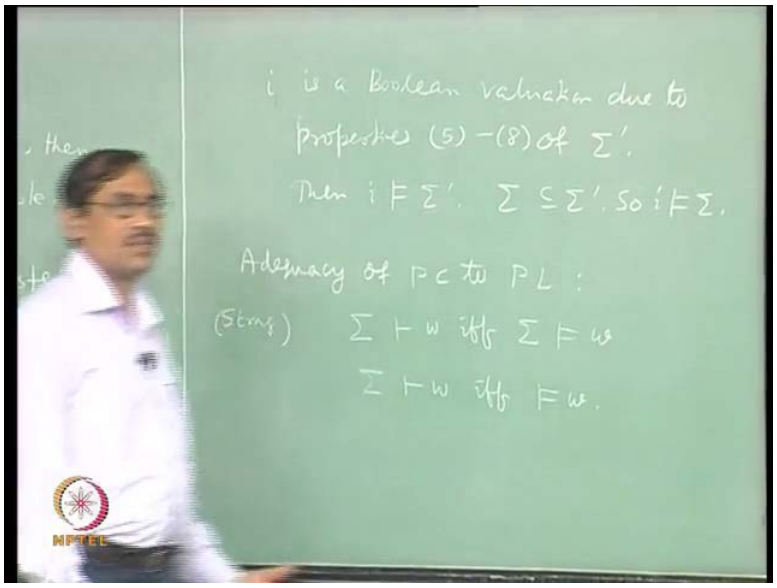
(Refer Slide Time: 02:05)



Define from the set to 0, 1, a function by taking i of any proposition, say u equal to 1, if u belongs to sigma prime, it is 0, otherwise. Now is it not obvious? Let this i be a model of sigma prime, it is because by definition it is 1, but we have to verify that it is really a Boolean valuation. It is a function that is fine, but it has to be a Boolean valuation in order that it is a model, that is why we have done all this work.

Now, it should be obvious that it is also a Boolean valuation because of this 4 properties. If we take any proposition u, either u is in sigma prime, or not u is in sigma prime, right? So, either it is given 1 or its given 0 with respect to the negative sign, negation sign; and what about other properties? The other three properties capture the implication, right? If q belongs to sigma prime, that is, if i of q is equal to 1, then p implies q belongs to sigma prime. So, i of p implies q is equal to 1, right? That is the first property of implication. Next, if not p belongs to sigma prime, that is, if p is given the truth value 0, then p implies q is also given a truth value 1, it belongs to sigma prime, right? That is the next property of an implication. Then you see that if p is true q is false, that is if p belongs to sigma prime and q does not belong to sigma prime, then p implies q should be false, that is also true: p implies q does not belong to sigma prime, that is your last property, right?

(Refer Slide Time: 06:00)



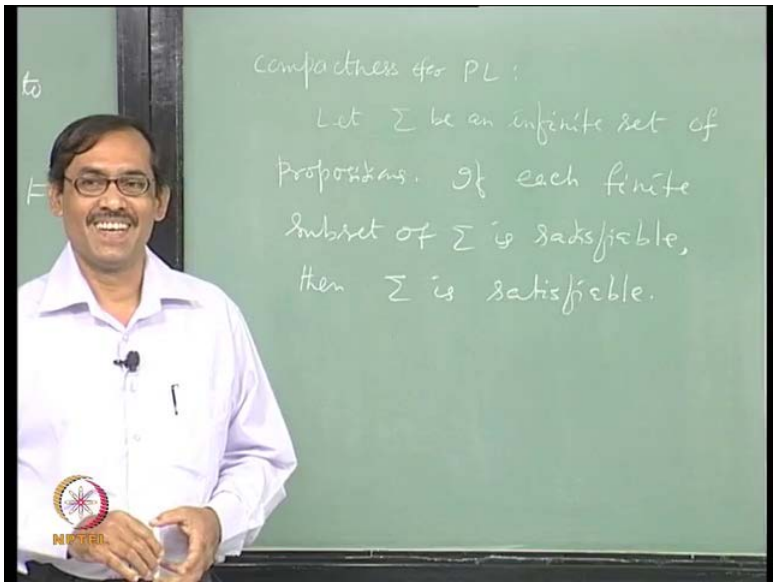
We just see from these, that i is a Boolean valuation, due to last four properties, which were that 5 to 8 of sigma prime? And that is all, that is the end of the proof. i is a model of sigma prime and sigma is a subset of sigma prime, so i is a model of sigma. i satisfies sigma prime, sigma is a subset of sigma prime, so, i is a model of sigma. Easy consequence. Once you have sigma prime, there is nothing to prove, and there by you have also proved adequacy of PC to PL. In fact, strong adequacy, right? Let us mention that. So, adequacy really means what?

Sigma entails w if and only if sigma semantically entails w. This is your strong adequacy. If you do not have strong-ness, you say w is a theorem if and only if w is valid, without any sigma's, right? Its proof is easy, because this is, this holds if and only if sigma and not w is inconsistent, and this holds if and only if sigma union not w is unsatisfiable. Now, consistency and satisfiability are equivalent. Model existence theorem says one part, soundness theorem says the other part, right? In fact adequacy means soundness plus completeness, both the things are there, is that clear?

There is a nice consequence of it; it says something like: you start with one infinite set. Suppose that it is unsatisfiable. Then you can prove that there exists a finite subset of it which is unsatisfiable. That is what compactness means, right? Which means. Same construction will prove it, but we do not have to prove it once more, because you have adequacy, and in PC we have finiteness. Immediately you can come to it. So, if you want see, for, this says, it looks simple now for all these, but it is not a very simple concept.

We just start with one infinite set, infinite set of propositions. Now, you take all its finite subsets, right? Suppose, if you claim that each of the finite subsets has a model, right? Can you construct a model for the whole infinite set? This is the problem. Do you see it? See, as a a particular case, suppose you have sigma which has w1, w2, w3 and so on. Now, for w1, you have a model for w1, w2, you have a model, w1, w2, w3 also have a model and so on, right? But not only these, we are assuming something more. There are, assume every finite subset of it has a model. Now, how to construct a model for the whole set? The clue is, when you say that every finite set, subset, has a model, it is not necessarily the same model, it is not the same i, okey? It can be different i, it need not be an extension from a subset to a super set of it, it can be completely different. Still from that you can always construct one model for the whole infinite set. That is what compactness says, right?

(Refer Slide Time: 10:48)



Let us write that compactness for PL. It says: let sigma be an infinite set; if each finite subset of sigma is satisfiable, then sigma is satisfiable. The proof is easy. What you have to say is, proof is by contradiction; if sigma is unsatisfiable then there exists a finite subset of sigma which is unsatisfiable. Which finite subset, we do not know, but we can show there exists one. So, suppose sigma is unsatisfiable. Then by strong completeness theorem or model existence, whatever you take, we want only completeness; you see that sigma is also consistent. Now, you are trading between PL and PC. If sigma is satisfiable then sigma is also consistent. You will have : if sigma is unsatisfiable, then it is also inconsistent.

Now, suppose sigma is inconsistent; so that means sigma entails p sigma entails not p, but some proposition p. Now, again you tread between PC and PL. You say, sigma semantically entails p, sigma semantically entails not p. But, wait a minute. From the PC itself, when you say it entails, you have a proof of it, but that proof contains only finite number of propositions. So, both the proofs have only finite number of propositions, each. So, there union is also a finite subset of sigma, now you concentrate on that finite subset and tread between PC and PL. Call that as sigma 0, which is the union of all those used in both the proofs or either of the proofs rather, right? You have to take the union. Then sigma 0 entails p, sigma 0 entails not p, semantically, sigma 0 is unsatisfiable. That is the proof, yeah?

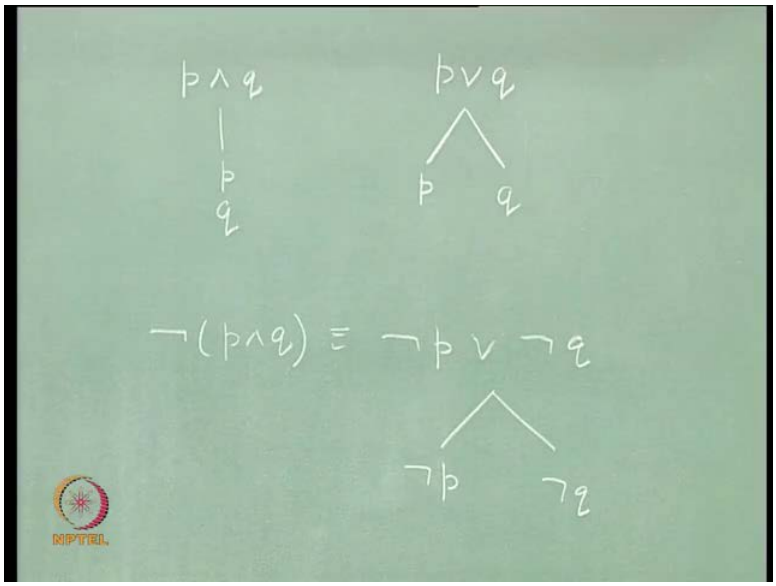
All the properties are not compact properties, that is why it becomes non-trivial. For example, suppose in natural numbers, you take any finite subset of natural numbers, it will have the maximum, but the whole set of natural numbers does not have a maximum. So, taking a maximum is not a compact property. But here it is; satisfiability is a compact property. That is what it says; that is why it becomes non trivial. Though in PC it is very trivial, that definition is through a finiteness property, the proof is a finite sequence. That is why when you say PC and PL are adequate, it gives some more information, it is not just I have started with some axioms and proved it, fine? If we get some time, we will do some applications of compactness. Right now, we will not, we will resist the temptation.

See in PC, what happens, you are looking at the proofs now. Construction of proof is not effective, as we have seen; but but not proved it. Why it is not effective? All that we have seen, from our experience, that the idea of proof, notion of proof is effective. You can always check whether some sequence of propositions is a proof or not, but how to construct a proof depends on your cunningness. It is not mechanical. Can you have a mechanical method? Something like that, resolution; it was somewhat mechanical; when? But you had to use cnf conversion, that is again mechanical. So, there is a hope, mechanically something can be done, right? And it is mechanical from the beginning.

Suppose, you have a finite set of propositions, you can always find out whether it is consistent or inconsistent, mechanically, we have very crude procedure, construct its truth table, right? It is very long, it does not matter, there exists a method. That, that concept is called decidability of a theory. So, in propositional logic the thing is decidable, whether something is consistent or inconsistent is decidable, provided you have that for a finite set. For infinite sets, you do not know. What to do?

Now, we will see that there is still, there are still some other methods, which can be very easy, and will present one such method. In fact the method is so easy and so intuitive, at the same time it is so formal, that logicians become addicted to it. Once you say this is consistent or not, immediately they will go for that. Yes? We will see why is it so. it is very simple observation.

(Refer Slide Time: 16:30)



Suppose, I take p and q, one proposition; now, how do I decide whether this is true or false? Well, first thing is, p and q is true, if both of them are true, right? So, you just construct a tree, if p and q both are true then p and q will be true. Suppose I take p or q, a similar construction, p or q will be true if at least one of them is true, right? I have now two cases, just the way you did from your childhood. There are two cases, one of the cases will succeed, then I say that p or q is true, right? So, I will branch into two parts p, q; I just keep it, I do not know what will happen to this p or q, but this can happen, right? This gives a method, how to proceed, but this says that preliminary conversion is required for and, or but we will not do that way. What we will do is, we will try to identify the forms of propositions.

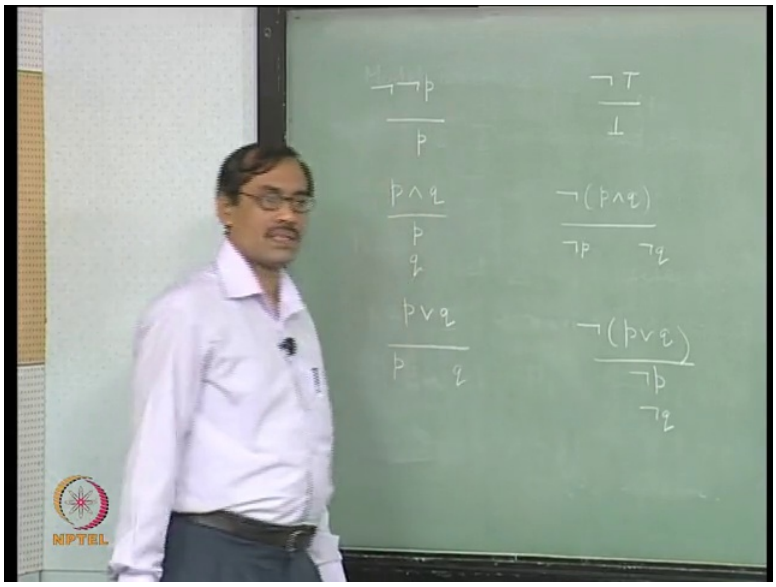
The propositions can be in which form? Suppose, all the connectives are used. Then, let us see what can be the forms of the propositions. This is required because by De Morgan, say, not of p and q, though I know p and q is true. When both of them are true I can make a tree with both p and q in this same path, right? But if it is not p and q, now it is equivalent to not p

or not q. I know it is equivalent to not p not q, then I should have crossed out with one as not p, another as not q, which says not of p and q is true, if one of this is true.

Now, you see even if and is there, because of negation sign, some proposition can be just and of something or negation of and of something. Similarly, from each connective or; the four connectives, you will get eight such cases, right? A proposition can be in that form or negation of that form, is it? And then, you have top and bottom, you have to do something for that; you have double negation. It can be not not, any number of not's can be there, right? So, recursively you should have a rule for not not also. Then this expanding propositions in such a fashion is called a tableau; you go on finding the tableau. How it is going, finding out the cases when it can be true, and so on.

These types of trees are called Beth's semantic trees. Sematically, we are just completing a tree, making a tree, finding out what are the possibilities when it can be true, right? Propositions at the root can be true, when, that is what the trees answer this. Trees have been modified to make a proof method, that is what we are going to present. These tableau will have certain rules depending on the forms of the propositions; we just write the rules first.

(Refer Slide Time: 19:48)



Let us see, we will have rules of this form. If it is in the form not not, you get p. Again we will write in this fashion, fractions. This means, if not not p is there, this will be true. If this is true, semantically it looks like, but when we construct a true proof system it can be semantically motivated where there is no trace of semantics there. It is mechanical; that is

what do; we want to make just like our PC. So, you just construct them as rules. Now, think of them as rules for getting the semantics, whenever need will come, we connect them, right?

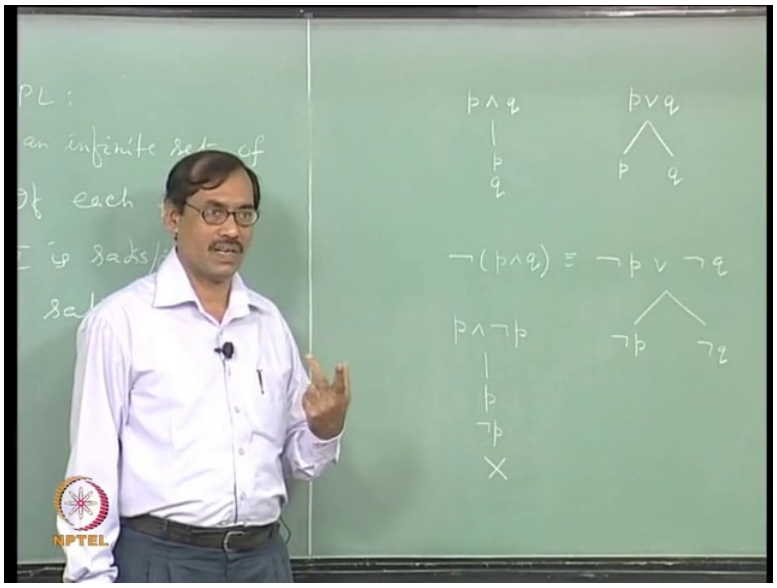
Now, if it is in the form not top, we can write a bottom. If it is p and q, we will have p, we will also have q. If it is not of p and q, then we will have two children, one is not p, another is not q. If it is p or q, we have again two children. This can be written as a fraction, this is a straight, this way, when you construct the tableau we will be constructing the trees. Then p or q will have p on one side, q as the other side, right? See if you do not like the fractions, you can simply connect by vertical bars or slanted bars. It does not matte; we have not of p or q, which will be not p, not q; both are in the same path, right? Next, you may have p implies q, which will have, again, two children, one is not p, another is q. See the semantics, p implies q is equivalent to not p or q. Next you have not of p implies q, which is equivalent to p and not q, right? So, we will have p, not q in the same path. Next, if and only if, p biconditional q, this is true when both of them are true or both of them are false. So, we will keep in one branch p, q, both, on another, not p, not q. Then negation of p if and only if q. That is equivalent to either p or q, right? Exclusive or, this is true when one is true, the other is false. That gives rise to p, not q, not p, q.

These are all the tableau rules. Any proposition can be in these forms, or there can be another form like bottom only. We have to give some where this rule. We have take care of that, how only bottom can be handled, okay? That is only missing here, fine?

Now, as it says, as our heuristic says, if a proposition is given, you are going to apply the tableau rules and then branch out, construct the tree. Construct the tree as we can, as far as you can go, continue. After the construction, what are you going to do? That is the point where we have to write something about the rules. How to use the rules; that we know now, for the construction of the tree, for the expansion; then after you expand what you are going to decide?

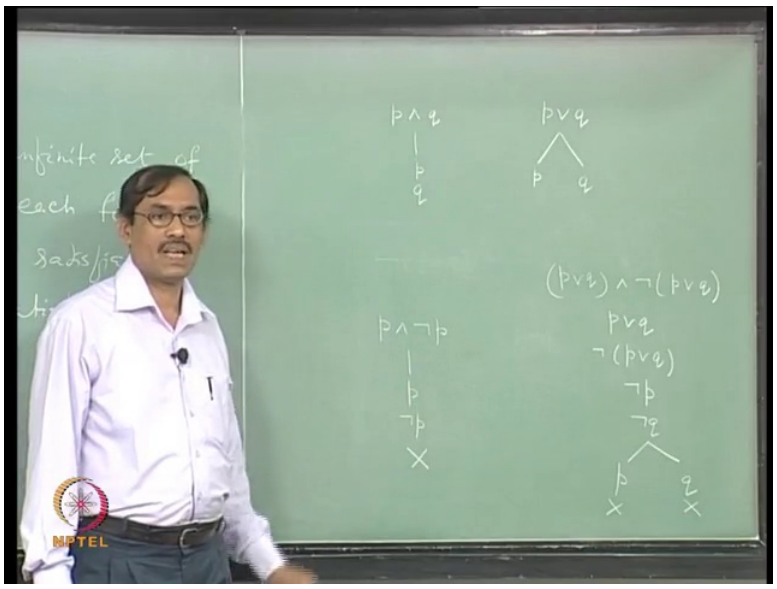
Well, it is also easy. For example, I take p and not p, I know this to be unsatisfiable, okay? Now, for the tableau rules, it gives me p, not p both. Semantically this says, in order that this will be true, this will be evaluated to 1, both p and not p should be evaluated to 1, which is not possible, right?

(Refer Slide Time: 23:50)



As a child, if it is not possible I will cross it, right? It is not possible. This is what we are going to introduce in the tableau. If, in a path of a tableau we find that there are two complimentary pair of literals, there is a pair of complimentary literals, then you close that path with a cross, provided that you had applied the tableau rules on every situation wherever it is applicable. Then you will end in literals. But you may not end in literals. If you end in a proposition p and also its negation, then also it is fine, whatever way we expand, they will close, is that clear?

(Refer Slide Time: 25:03)



For example? Let us see. I have p or q and not of p or q. Then once I apply the tableau rule, it will give me p or q, and again not of p or q. Now, not of p or q gives me not p, not q. And p or q gives me p, q as different children, okay? Now, I see the path. The path you can see either from the root to leaf or from leaf to root, does not matter. Let us take root to leaf as usual. So in this path, you see that p is there, not p is there. So, this path closes. In the other path, I see not q is there, q is there; that also closes. Now, the question is why should I go this far? I have already got the proposition u and not u. Whatever way I expand they will close, right? This is the shortcut we will be going to work with. Instead of going to the level of literals, whenever you find the proposition and its negation, stop there, close the branch, is it okey?

That is why we did not have a rule for the bottom; this is what we are going to write. If you find in a path a proposition or its negation, you close it or if you find in a path bottom is occurring, close it; because that will tell you: in order that the propositions in that path are true, bottom should be true; it can never be. So, that path also closes, right? That is how we are tackling this bottom; all others are given in the rules.

Our rule will be: if in a path you get a proposition and its negation, okay? That is one case. Second case is: if you find a bottom then close the path. Now, suppose you take a proposition and construct a tableau for it using the tableau rules. You see that every path closes, then what do you conclude? It is unsatisfiable. If every path does not close?

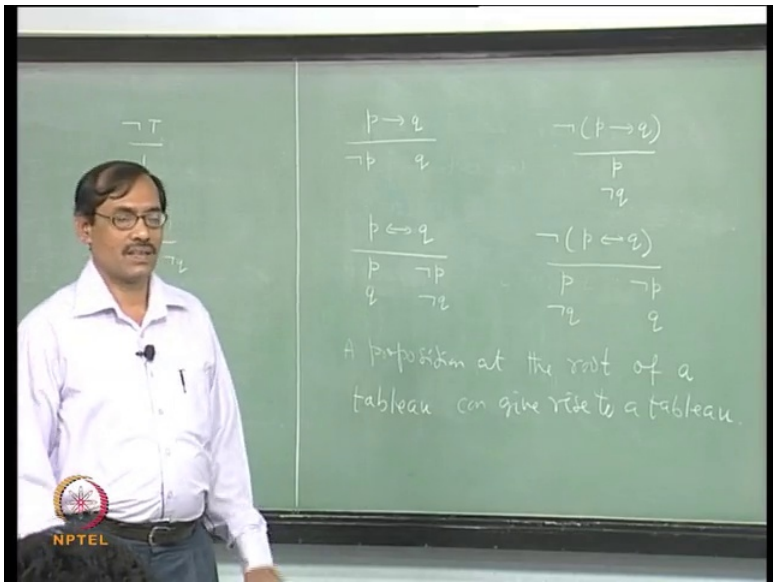
Student: We are getting the literals.

See from the begging itself, I could have told that p and not p are satisfiable. I do not expand it, I become lazy. p and not p, I do not expand it, I keep there. Now, this also a tableau by definition, it is open, it is not closed. So, it is really satisfiable, but that is not the case, right? That means to take a decision that one open path really gives rise to a model which will make it satisfiable, you have to see that all the compound propositions have been expanded. So, compound means all those propositions in the forms of the numerators in our rules, right? They are the compound propositions.

So, you should have applied a rule on the compound propositions, in that path; if it is done, still the tableau remains open, then it will be satisfiable. We have to make certain definitions for all these situations; to take care of the situations, then it can start. Let us give a definition for whatever we need there. First thing is, you have the tableau rules, then you have to say

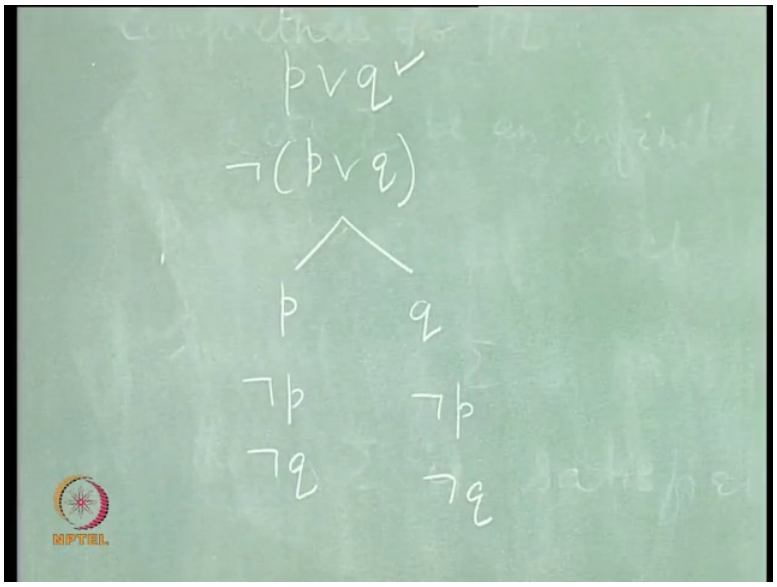
how to expand the tableau so that a proposition at the root of the tableau can give rise to a tableau.

(Refer Slide Time: 29:05)



These were criptically written; it says start from a proposition, then apply the rules if it is compound, and continue; that is what it says, recursively, right? But then there is one simple question. The question is this: let us look at the tableau, we have constructed here from the third proposition, we got not p and not q instead of the third, suppose I go for the second.

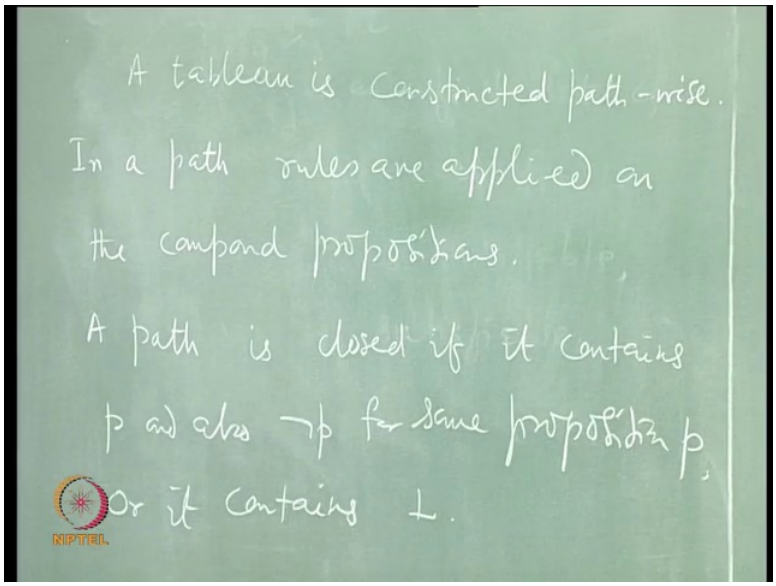
(Refer Slide Time: 30:11)



What will happen? I have p or q, I have not of p or q, now I first apply or rule, I would get p, another as q, now I want to apply this. See that. that is the point. That is the subtle point, while you are completing the, going for the tableau expansion. Once you take one proposition at the top level, which is beyond all these branching and you want to apply tableau rule on that, then you have to add on every leaf. Because, tableau are expanded path-wise, you have to concentrate on each path, because we are going for a construction of a model along that line, right?

So, in programming, it can be is easy, because you can expend all these things path-wise, continue, if it does not close, you stop there; if it closes you backtrack; that can be done. But while doing it manually, usually you apply it breadth-first way, not this way path-wise. Once you do breadth-first way, be careful about that. If you apply a rule, you think that I have already applied, but that is guaranteed if you have added it every where. Let us take it, I have not used it; to use this, I have to add not p, not q in both the branches. This is what it says. Now you can see all the paths will be the same, same literals here, same literals also here, is it clear? Now, you have to define all these things formally.

(Refer Slide Time: 32:11)

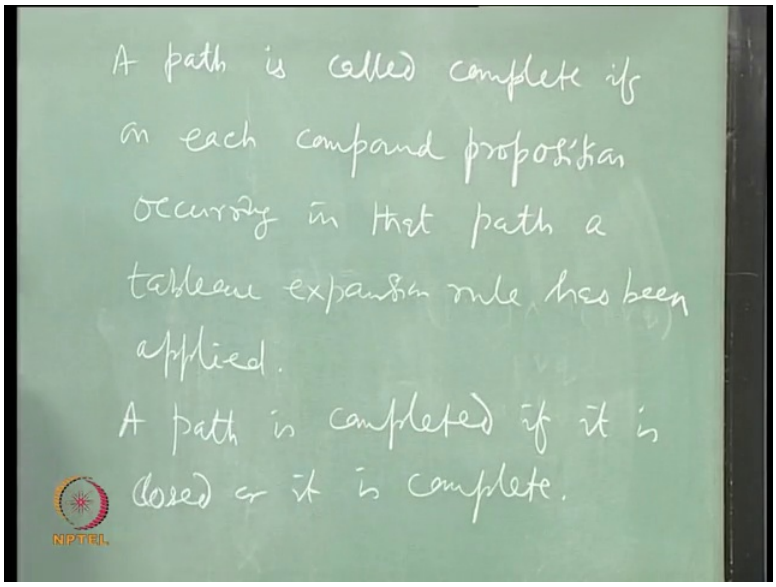


A tableau is expanded or it is constructed path wise, in a path rules are applied on the compound propositions. In this sense, the tableau rules are really tableau expansion rules; they are used to expand the tableau, right? That is how it is constructed throughout. Sometimes you need have to concern with the whole tableau; somewhere you find p and not

p is there, for some proposition p; you close it. There is no need to go further. If we want to go further, then you should give a name for that kind of tableau, right?

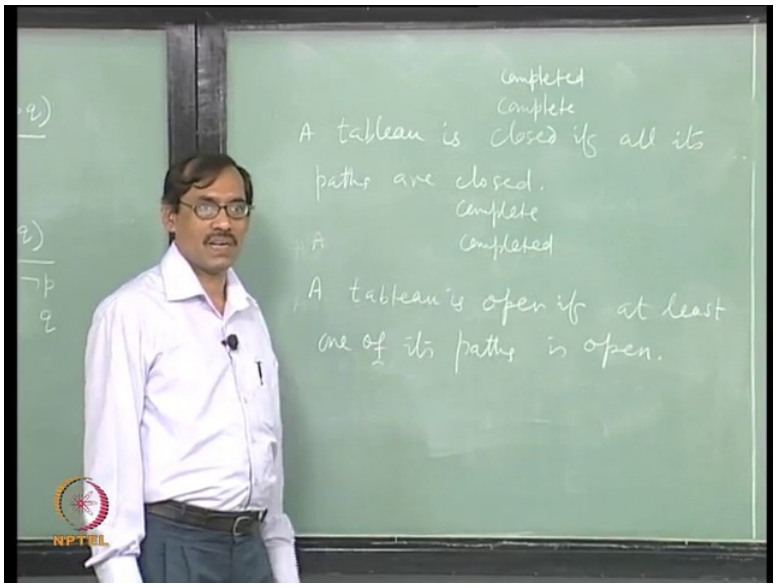
Say that, the path is, so path means path from root to leaf, wherever you are now, that is current leaf, a path is called closed if it contains p and also not p for some proposition p, or it contains bottom. Otherwise, you say the path is open. These open and close, they are dynamic; up to some level a path can be open, but once you apply another, one of its paths can be closed, even both of them can be closed later. It is a dynamic thing.

(Refer Slide Time: 34:17)



A path is called complete, called complete, if it can no more be expanded, right? That means on each of the compound propositions occurring in that path, a rule has been applied, right? If on each compound proposition occurring in that path, a tableau expansion rule has been applied. See, some times you say I do not have to complete the path, I can take decision before it. So, we will also include one term for that. We will say that a path is completed, if it is closed or it is complete. That means, it is essentially complete, right? Nothing can be done for it; you know it is closed, you have decided for it, or it remains open. Nothing more can be done by using the tableau expansion rules. In fact, we wanted this completed path to take any decision, right? Then you come for the tableau itself. Let us leave it here.

(Refer Slide Time: 36:12)

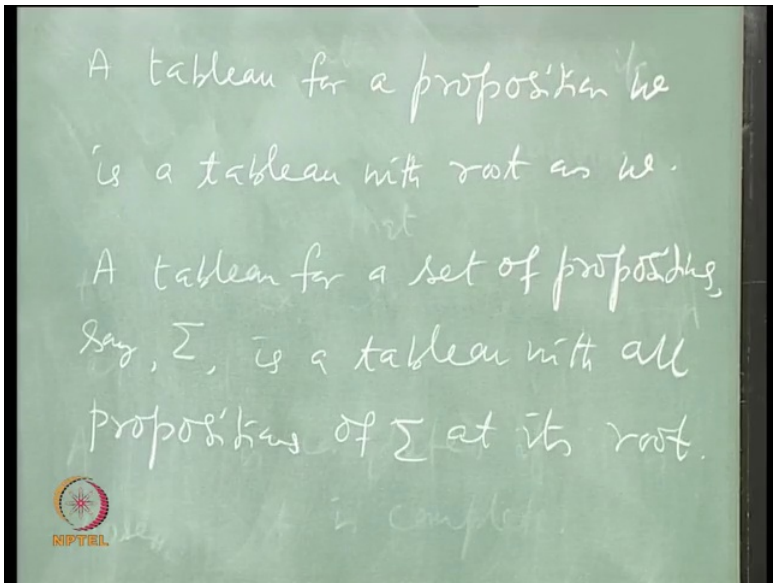


A tableau is closed if all its paths are closed. This is what, should give us an unsatisfiable proposition at the root. Similarly, we say a tableau is complete if all its paths are complete, not one, complete, everywhere compound propositions have been resolved, those have been expanded. Similarly, you say tableau is completed if all its paths are completed, right? So, closed or complete or completed, all these hold.

Now, comes the next one, a tableau is open if, here you require that, if at least one of its paths is open, not every, right? You do not need every, if at least one of its paths is open. For taking decision whether this is satisfiable, you may need an open completed tableau, right? It should be completed; if it is not complete, but only open, a decision cannot be taken. It can still, further be expanded and found to be closed, okay? So, we need a completed tableau for that. Then you should define the syntactic part of all those things. We cannot say satisfiable or unsatisfiable here because it has nothing to do with the truth or falsity, though it is semantically motivated, but as a proof system it may not have any semantic content, right?

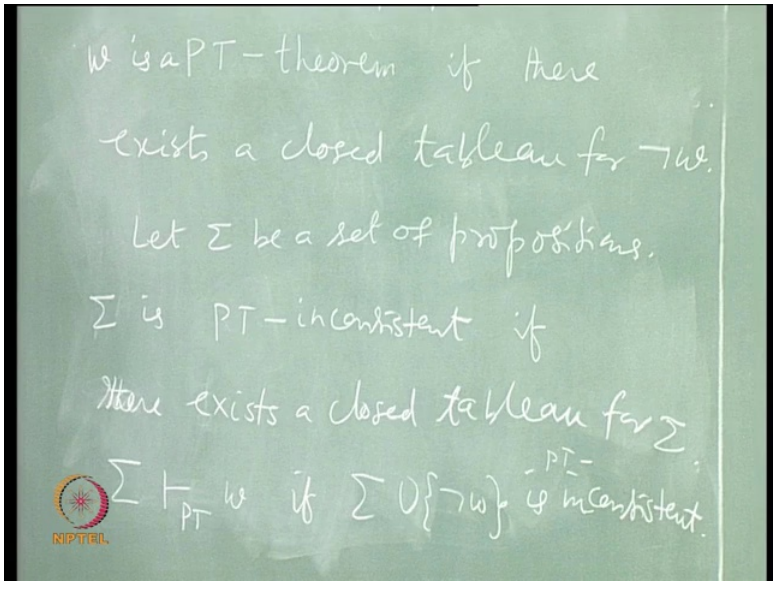
A tableau for a proposition w is a tableau with root as w. You have to define everything almost. Once you say a tableau for a proposition, that means you start the tableau with the root as w, then continue expanding it. Similarly, we can say a tableau for a set of propositions, say, sigma, is a tableau with all propositions of sigma at the root.

(Refer Slide Time: 38:43)



Now, we define the theorem, a tableau theorem or we say, propositional tableau, PT, for propositional tableau, is a, or we can say what is a theorem. We start with a proposition. We say w is a PT theorem if what happens, if there exists a closed tableau for not w. Now, you see reductio ad absurdum is inbuilt in the definition, okay? You start with not w, then expand it, find that the tableau closes. See, a tableau may not be unique; it depends on which order, we are using the rules, it can become very different.

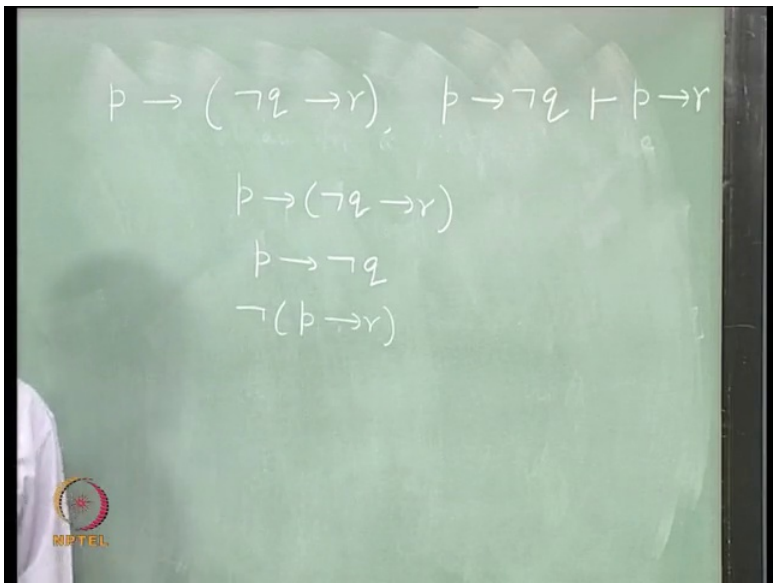
(Refer Slide Time: 40:08)



So, all that you need is there exists a closed tableau, fine? Then similarly for a set of propositions. We say that sigma is inconsistent, in fact we should write PT inconsistent, inconsistent in the system PT, if there exists a closed tableau for sigma. Start with all the propositions at the root and then continue by using tableau expansion rules, see that it closes; if it is closes, you say that it is inconsistent.

Similarly, you say sigma entails PT w, right? If sigma union not w is inconsistent, PT inconsistent. So, inconsistent here means PT. Slowly, since you have become matured, we forget this PT, always writing, you will say inconsistent, PT theorem as theorem and so on. Once we are in the context of tableau, it will be interpreted as tableau consistency, not PC consistency, okay? Let us see one a example, at least how to show something is inconsistent or not. Let us see whether this happens or not. In fact, you should have PT here, as a subscript, in PT only, we are deciding. So, that means some times will say that sigma entails w has a tableau proof, that is also used.

(Refer Slide Time: 43:26)



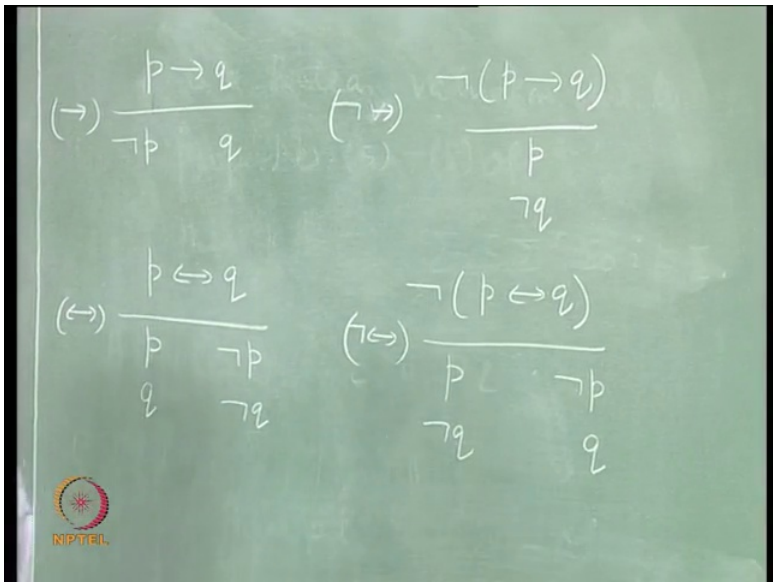
Now, how to proceed? You simply start with sigma union not w. So, p implies not q implies r, p entails not q not of p implies r. Suppose, if we write in different propositions vertically, because you want to apply the rules, right? They will be on the same path, so it is more logical to start with vertical things, than comma and then write it. Now, we have to apply the rules. See, I want to apply a rule on this first, why? If I start from this, then it will branch out

from the beginning; then at every branch I have to add this. If I start with this one it will be stalking, it is not branching. So, it will be helpful, okay?

In all those rules wherever you get two children they are called the branching rules, others are called the stalking rules. They are just stalking the propositions one after another, the other ones are branching out. Now, it is helpful as a heuristic that if there is a choice, go for the stalking rules instead of the branching rules; that will make the tableau shorter.

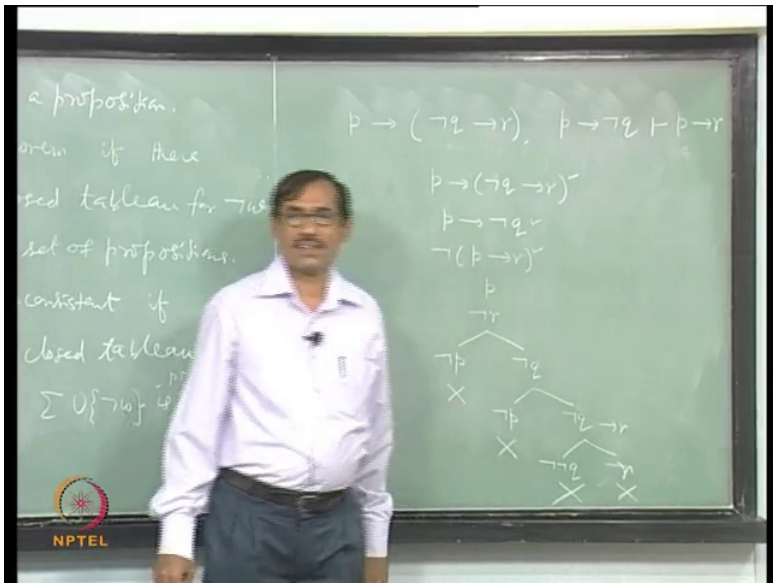
Now if I apply not implies rule, I can give names to all the rules. Let us call these rule as not not, that is the form of proposition, we will be meeting. This one will be not top, this is and, this is not and, this is or, this is not or. Similarly, this is implies, this is not implies, this is biconditional, this is not biconditional. That says the form of the proposition directly.

(Refer Slide Time: 46:02)



Now, if I apply not implies rule it gives you p not r, is that right? Next, which one? Well, I can go for the middle one, instead of these, that will I can branch out. So, this gives implies rule, I use. That gives, what? So it is often easier for you if you go for a book keeping device. Once I use this, I tick it; it will not come further. Then, what about p implies not q? It will branch out to not p, not q. Now, I close it. Instead of applying this, I will close it here, because I have already got p and not p in this path. So, I close it. And on this not q, the other path, not q, not r, p and so on. And I have one on which I have not applied a rule, the tableau expansion rule. This, I have to apply, implies rule, so that gives me not p and then not q implies r, they come in this path; I find not p, not q, not r, p.

(Refer Slide Time: 46:20)



So, p and not p are there; that path also closes. Then I look, what, q implies r, all the things have been used in this path, there is another compound proposition where I have not used the rule. So, I branch out to not not q and r. If we use double negation it will go to q, but I do not need that. I have not q, here not not q. You can further take q, but does not matter. That also closes. And here not r and r, also closes. So it is closed. Therefore, the set of propositions is inconsistent. We do not have to say that, we just say, this is true by tableau method; because our definition says: if there is a close tableau for sigma and not w, that we have taken directly.

Now, as an exercise, you can try all the axioms of PC and MP also. See, how do they go in tableau rules? Slowly you should get acquainted with this; very quickly you can decide what happens, that helps. So, today we have done tableau. No, we have done also completeness of PC and compactness and then the tableau rules. We will discuss again tableau next time.