Artificial Intelligence:

Propositional Logic:

Rules of Inference and Natural Deduction

Prof. Deepak Khemani

Department of Computer Science and Engineering

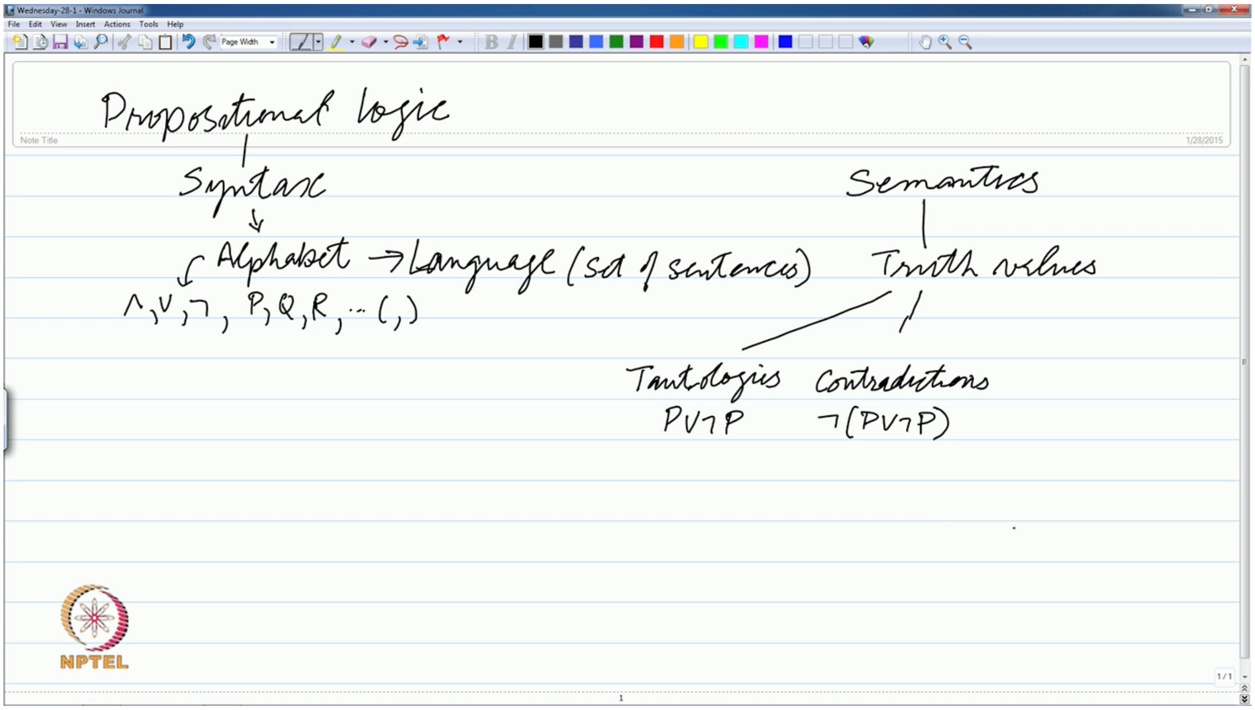
Indian Institute of Technology, Madras

Module – 02

Lecture – 04

Okay so let's continue our study of propositional logic. So to do a very quick recap we first looked at syntax and in the syntax once you define an alphabet you get a language. So the alphabet includes if you remember things like connectives and propositional variables and so on. So language is basically a set of sentences and alphabet is things like and or not P Q R without going into detail these are the kinds of thing we have. And then we have semantics. So as far as propositional logic is concerned by semantics we mean truth functional semantics we define the notion of truth values and we saw that there are three kinds of formulae or sentences in any language. One is tautologies these are sentences which are always true. When we say always true we basically mean they are true for any valuation of the atomic propositions so whatever the atomic propositions are P Q R S T and so on whatever valuation you give to them which means true or false these sentences will always be true and example of that was P or not P. then we have sentences which are contradictions and these are sentences which are always false. It means any valuation you choose it will be always false. And an example of a contradiction is this sentence which is the negation of the tautology that we have seen. I have particularly chosen this example because there is this relation between contradictions and tautologies. That if you have a tautology and if you put a negative sign before that it will become a contradiction and vice versa.

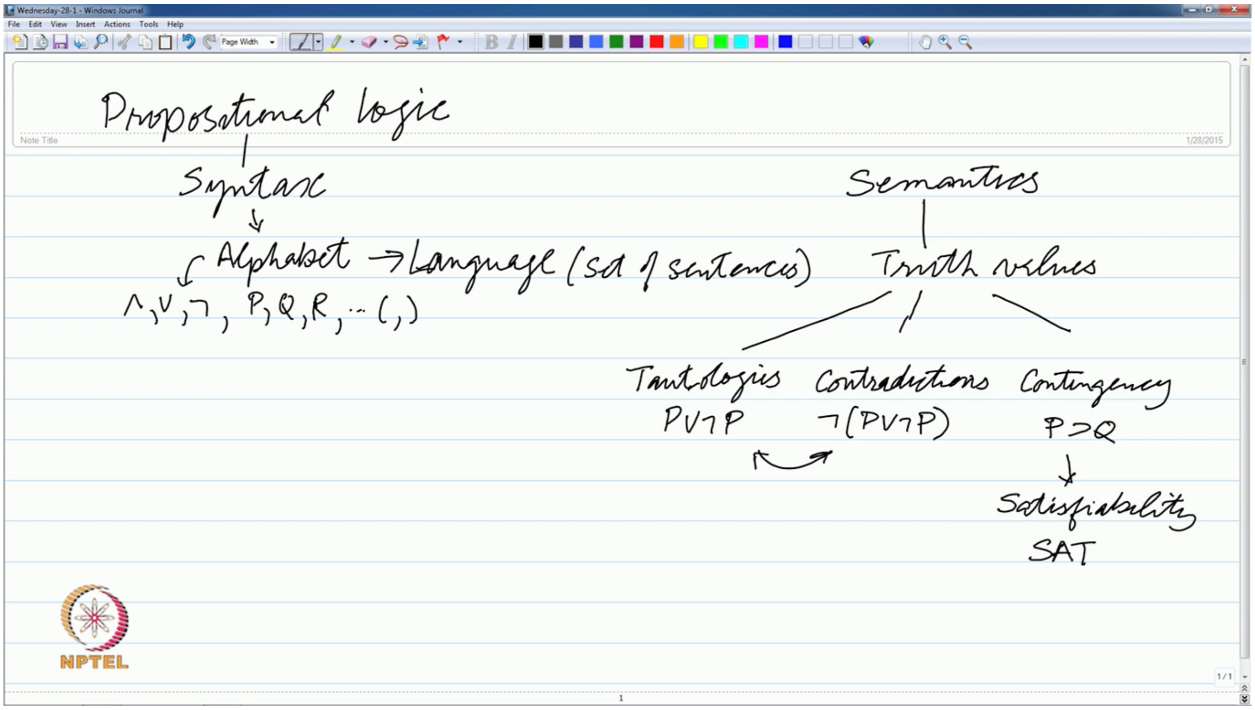
(Refer Slide Time: 3:16)



So in this case we have taken a tautology P or not P put a negation sign in front of that and it becomes contradiction. And the third kind of sentences are contingencies and by contingency we mean that the truth value of the sentence depends upon the valuation that you are choosing. So a sentence like P implies Q for example if you choose P is equal to true and Q is equal to true. But it would be false if we choose P is equal to true and Q is equal to false. So its truth value is not independent of the valuation it depends upon the valuation. So we would be more interested in tautologies and contradictions because we are more interested in proof systems. We want to show that whenever certain formulae are always true we have a way of reaching to those formulae. So before we do that so when we talk about contingencies we talk of satisfiability. And the well-known SAT problem which is common in many areas of computer science is basically addressing the satisfiability problem.

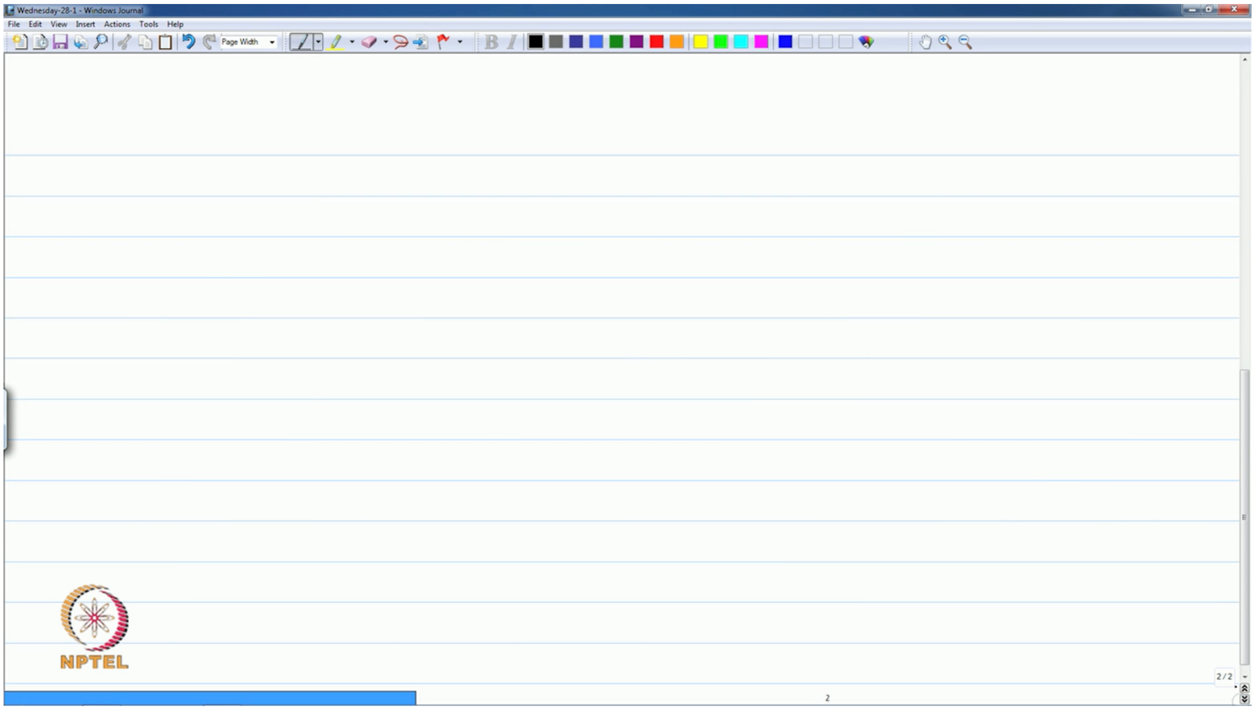
You are given some sentence you are asking the question if there some valuation which will make this formula true or which will make this sentence true. So that's a different kind of a game altogether we will not get into that. We are more interested in tautologies and contradictions and we will see what we talk about.

(Refer Slide Time: 4:52)



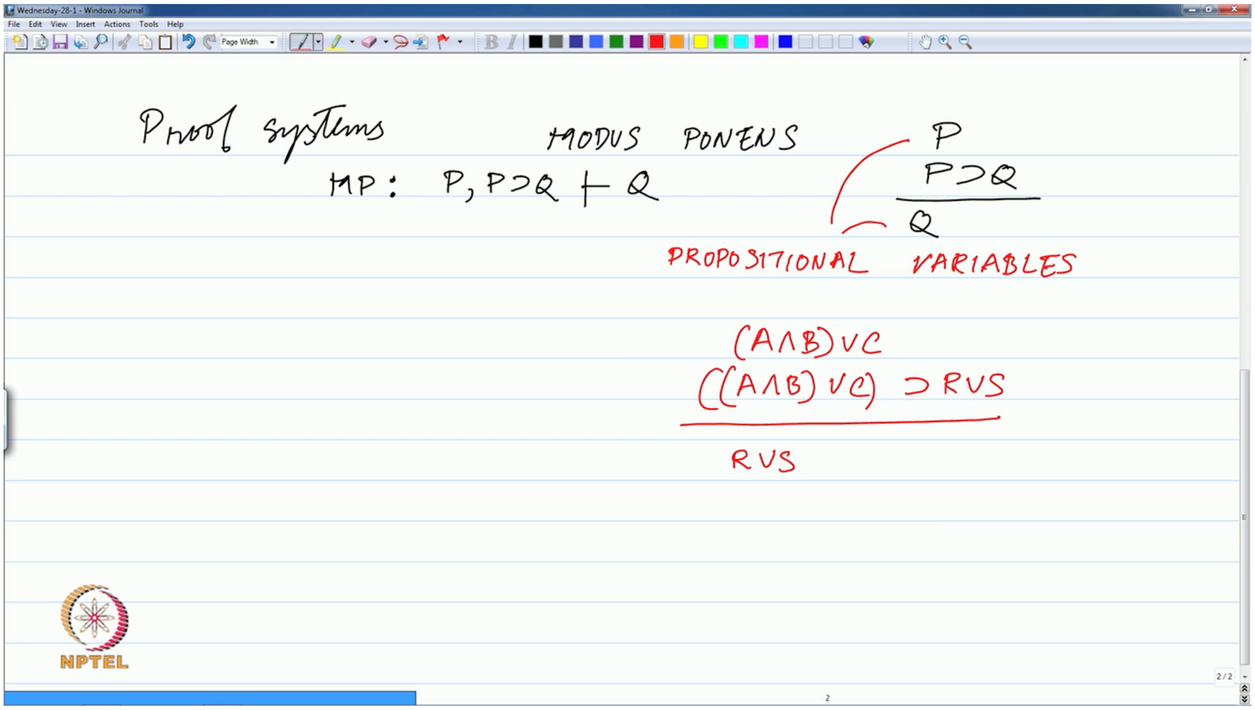
Then we define the notion of entailment so if you remember the notion we said that a knowledge base KB which is set of sentences that are given entails a sentence alpha whenever alpha is necessarily true as long as KB is true. So whenever KB is true alpha will be true and this is the notion of entailment. Correspondingly at the syntactic level we have this notion of provability the notion of proofs and that's the concept we will be exploring in most of this course. And we write that a knowledge base proves a sentence alpha and we are interested in that kind of logical machinery where provability implies truth or entailment and entailment implies provability which means you want to be able to prove all true formulae and we want to be able to prove only true formulae. Once we have such machinery we can throw away such semantics and we can just work with the syntactic machinery which is the proof machinery. Now for a proof system to work we need a language that the syntax gives us plus we have rules of inference. So let's look at proof systems

(Refer Slide Time: 7:04)



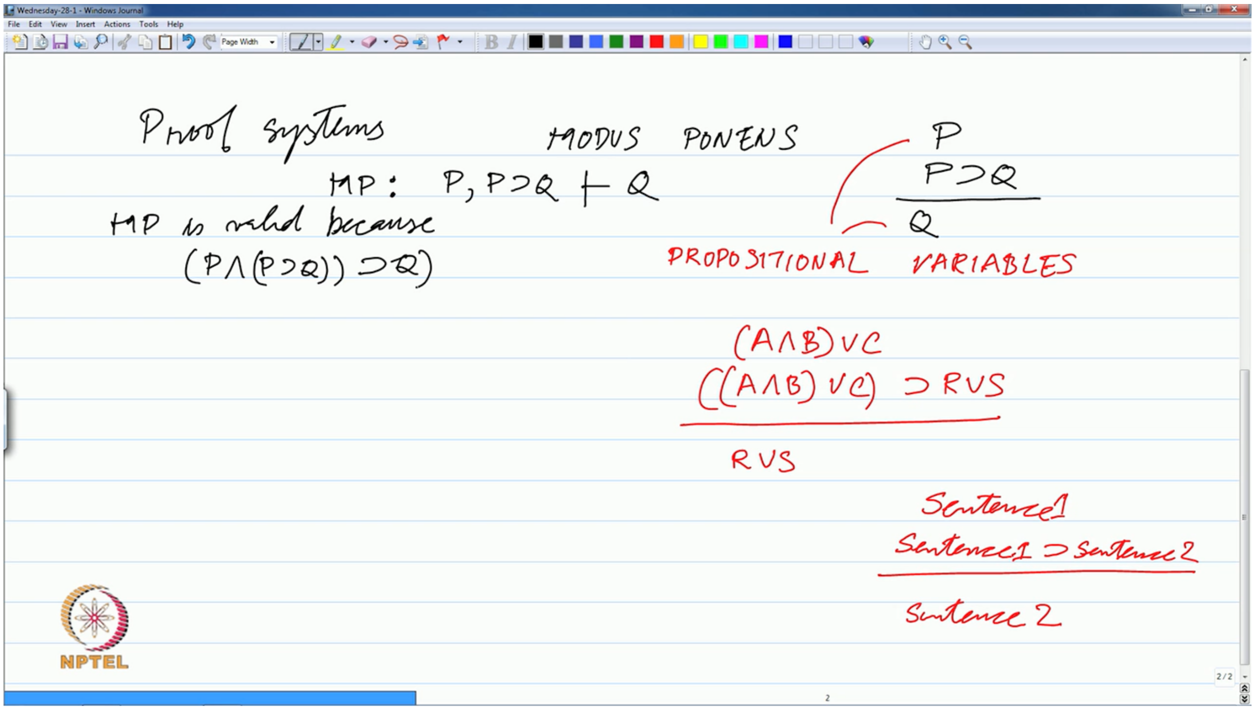
So let me start with a example rule. The rule that we have been we always start with whenever we look at logic and that is the rule of modus ponens which we have seen earlier and the rule is written most commonly as P and P implies Q Q. so the sentences above the line are given or antecedents and the sentence below the line is a consequent or what you derive. We also write this if you remember as P comma P implies Q and we use this symbol that we have and this rule we will call as MP which is a short form for modus ponens. Now the first thing to emphasize which I have done earlier also just to repeat that P and Q are propositional variables which means you can plug in any sentence in place of P and Q. so for instance so what this defines is only the form of the argument. So I could have written something like this A and B or C even I can write A and B or C implies R or S implies.

(Refer Slide Time: 9:13)



So if I do this I am still using the rule of modus ponens because what I have is something a sentence of any kind which in the original notation we said P but in this particular example the sentence is A and B or C and then we have another sentence which says A and B or C implies R or S. and from that if we can add R or S then we can this is also an instance of modus ponens. So this is just to emphasize the fact that when we say P or P implies Q these are propositional variables and you can plug in any sentence doesn't even have to be in propositional logic it can be in first order logic or modal logic or any other logics. What really we should write is something like this that if you have sentence one whatever that sentence is in fact in any language and if you have another sentence which is of the form sentence one implies sentence two then we can infer sentence two. So this is the rule of modus ponens and we saw that there are other kind of rules. So what do we mean by rule of modus ponens that it's a form of argument that we are willing to accept which says that whenever we have two sentences which match this pattern that one of the sentence is the kind P and the second sentence is of the kind P implies Q then we can make an inference of the third sentence which is Q. whenever this form is conformed to the argument will be accepted as valid. And if you remember that we had said that MP is valid if or rather I should say because this sentence P and P implies Q implies Q is a tautology.

(Refer Slide Time: 11:44)

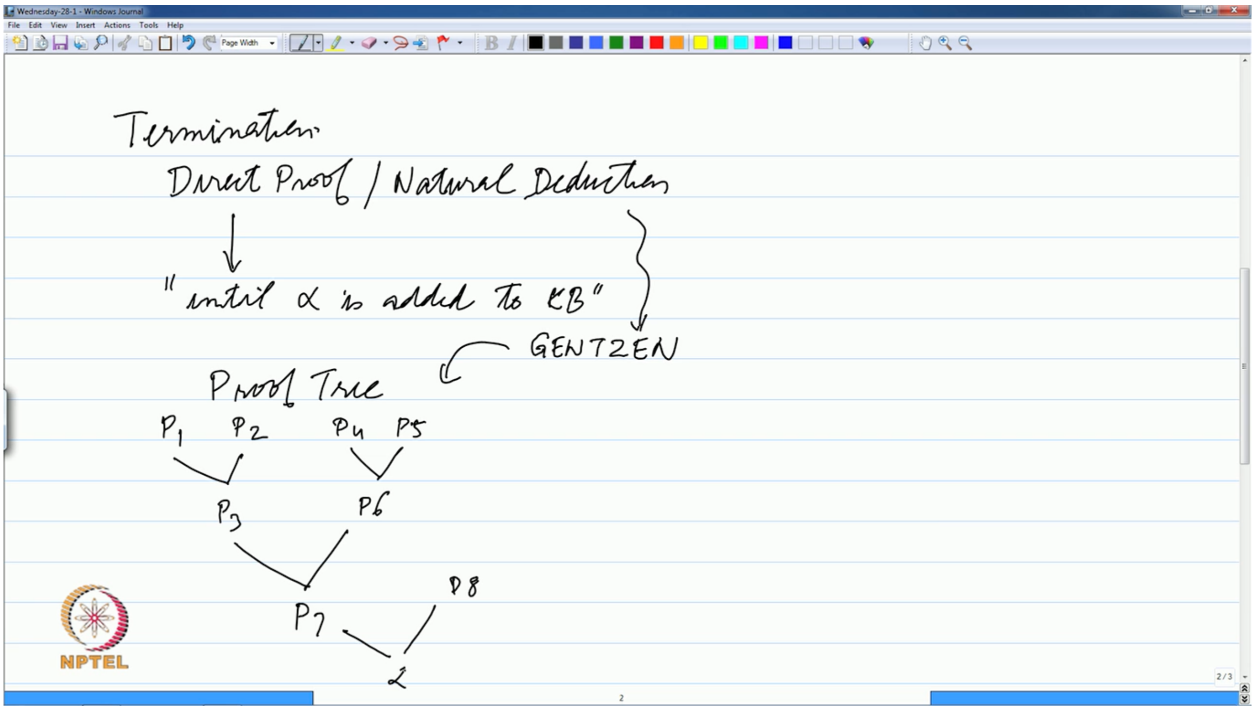


So every rule of reference is based on the tautological implication we had seen that in the last class this is just a quick review of what we had seen. So what is a proof system a proof system does the following that it's an algorithm you might say that pick some rule and when I say rule I mean rule of inference with matching antecedents which means in our knowledge base there are certain formulae which match that antecedents then add the consequent to the knowledge base and put this whole thing into a loop until some termination happen we will look at this. This is by and large the outline of any proof system. A proof system is a sequence of applications of rules till certain criteria is satisfied. We will look at all these criteria. Another important point to note is that this notion of some is really a key question. Because once you pick a right rule you will get a short proof if you pick wrong rules you will produce formulae which is not of any use. So that some is going to be a critical.

So let's talk about the termination criteria.so by a large we tend to distinguish between two kinds of proofs one is called direct proof or we use a term often natural deduction. So we have thrown in the term deduction here so at this moment you will simply say that deduction is that form of inferences which use valid rule of inferences. And what we do in direct proof is the termination criteria is until alpha is added to the KB where alpha is a sentence you are looking for, you want to prove a certain statement and we use a term theorem. In direct proof you keep adding formulae till you get the formula you are looking for and that's the most natural form of proof and that's why we often call it natural deduction. This name is also attributed to a logician called Gentzen and in fact there is a whole system of proving things which is known as gentzen systems which are basically natural deduction systems and in gentzen system we have a proof tree where the leaves are what is given to you so P1 and P2 if they are sentences they may give P3 and then

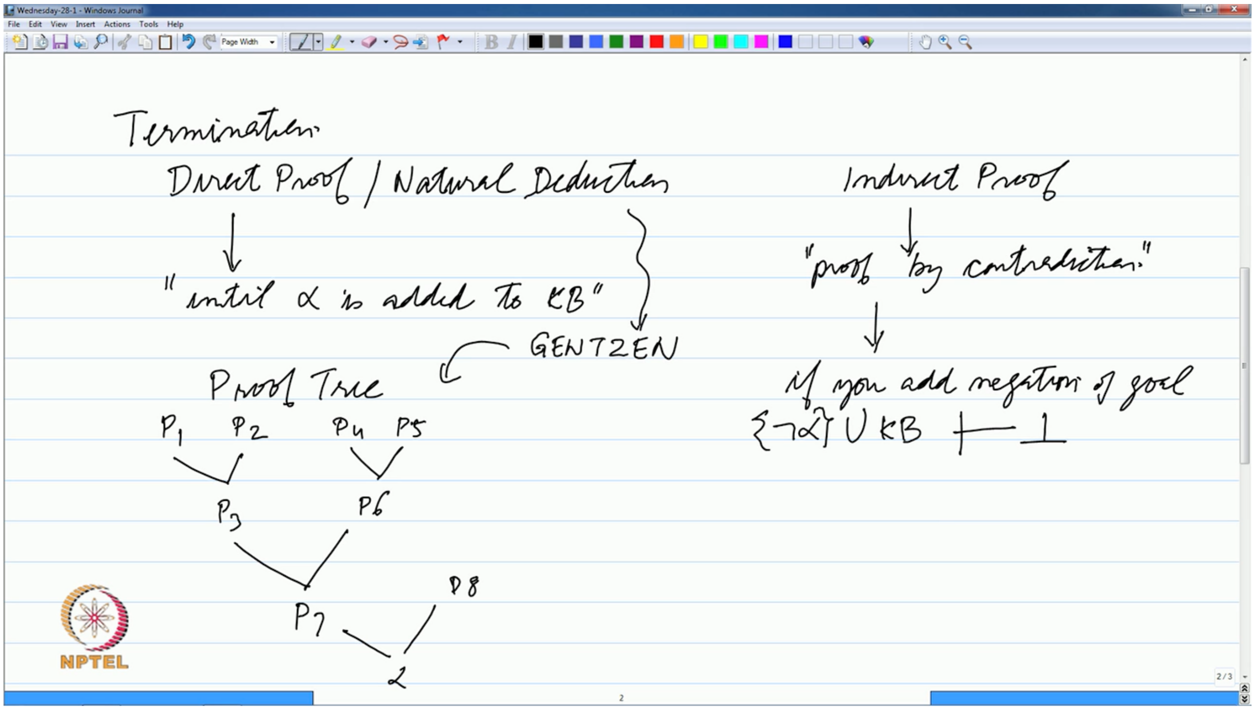
P4 P5 may give you P6 and this may give you P7. And some formula P8 and this will give you alpha. Alpha is what you are interested in.

(Refer Slide Time: 16:22)



So gentzen systems are natural deduction systems where the leaves are what are given to you and internal are the consequents we have added and you terminate when you add the consequent which is alpha. The other form of proof is called indirect proof and in indirect proof most of the time we will look at two systems one is the tableau system and one is a resolution method. In both these methods you try to its kind of proof by contradiction. And what do we mean by this that if you add negation of goal so we often use the term goal for the formula or sentence that we want to prove which means negation of alpha to the knowledge base then this results in a contradiction which we write as. So remember the symbol I have written on the right hand side is the symbol for bottom it's a sentence which is always false and what you are doing in a proof by contradiction is that if you add the negation of the goal then you end up with contradiction which means the negation of the goal must be false which means that the goal itself must be true.

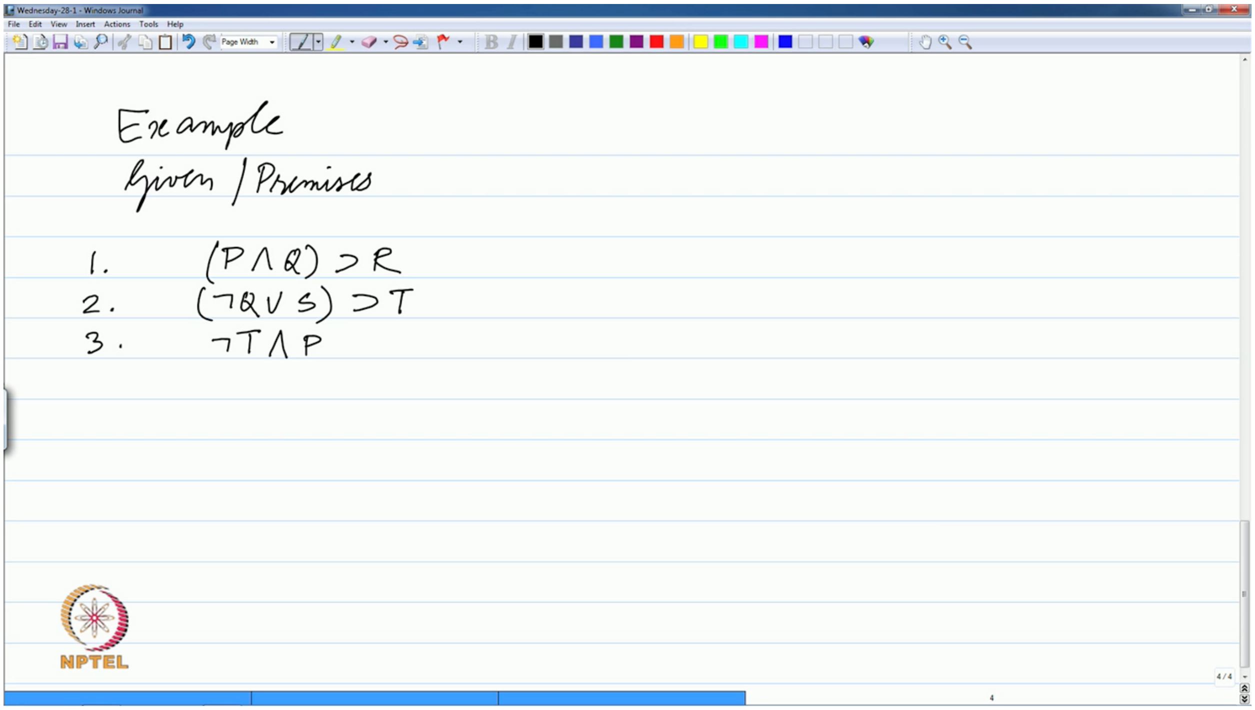
(Refer Slide Time: 18:12)



So we will look at this in little bit more detail. Let's focus on the direct proof to start with and keep in mind that our goal is to be able to write programs which will do all this work for us. So this whole exercise is called theorem proving and we want to write programs which are often called theorem provers. And what they will do for us is that we will give them a set of things which are given or true and we will say that show that this thing can be derived and the theorem prover would produce a proof for that. That is our interest. So we would prefer those methods where it is easy to write algorithms and we will see as we go along that indirect proofs are easier to implement than direct proofs because very often direct proofs involve certain amount of guess work certain amount of creativity if you want to say including the rules and things like that.

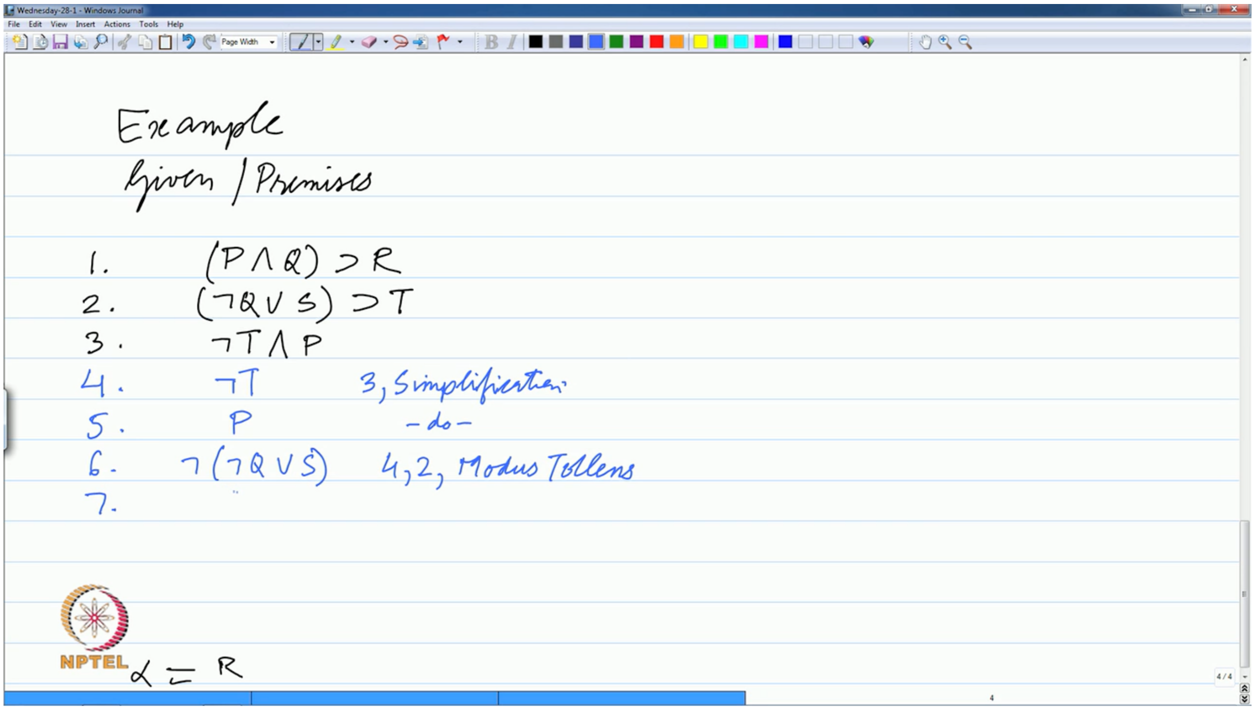
So let's look at an example of the proof. So let us say some sentences are given to you. So which we use the term given so remember your school geometry school algebra you said this is given and to show that kind of thing or we can use a more formal thing which is premises which you accept because of some reason. It could be because of the domain you are talking about a sentence pertain to certain domain so you would say P is true for example or P and Q is true or something like that. So let's say we are given some sentences I will just I am just choosing some random sentences here so let's say P and Q implies R that's given to you. You don't question why it's true it's something given to you and let us say not Q or S implies T. you can plug in any sentence for P or Q or R. so for example P stands for Alice likes mathematics. Q could stand for Alice likes Physics then R could be Alice would study science or something like that you know so if Alice likes math and Alice likes physics then Alice will study science or something like that. So can plug in virtually any sentence you are not really interested in what you are talking about you are not interested in content we are interested in form of sentence. Let me just take a very simple example. Not T and P. let's say these things are given to us.

(Refer Slide Time: 21:09)



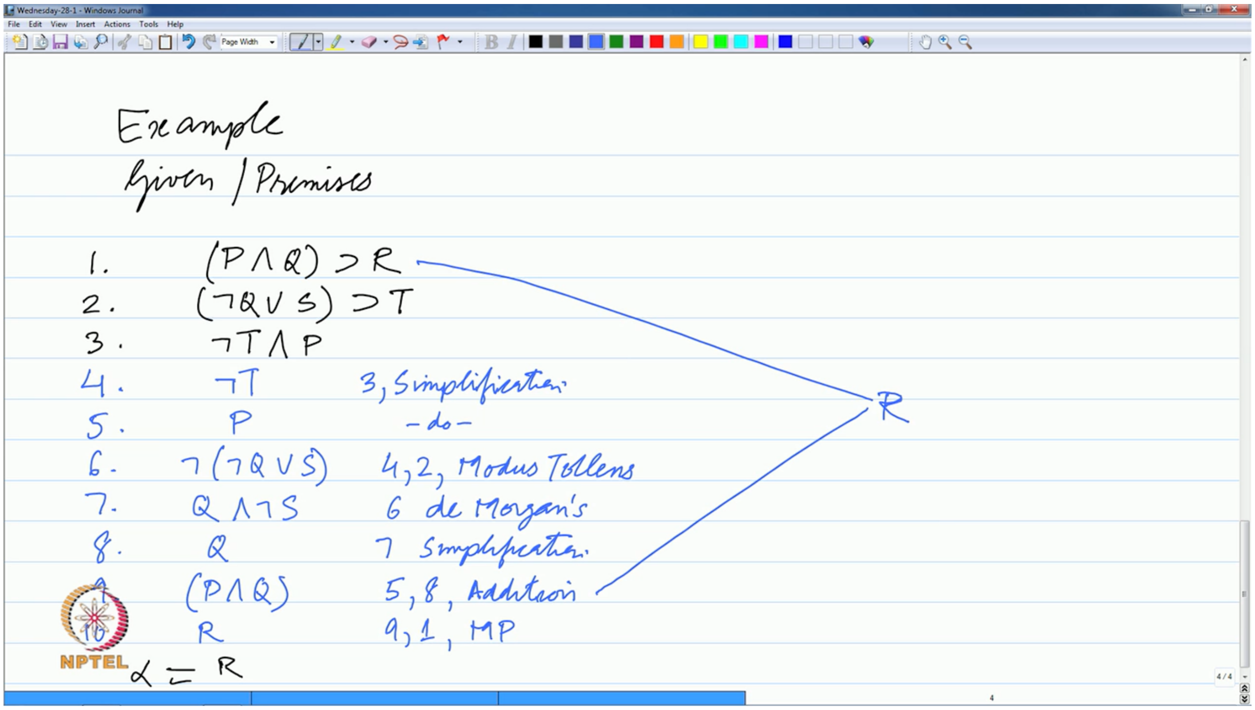
I hope this works and we want to show let me write it here alpha is equal to R. so we are saying that if the first three sentences are given to us does R follow logically from this. So one way you have seen earlier is to construct a truth table just construct a large formula based on this and see whether R turns out to be true or not. but we are interested in proof methods right now. So let's do a proof of this. So what I am writing in blue if you can make out are sentences from this. So for example not T so this comes so typically the way we write proofs are that how did you get this not T we got it from sentence three and a rule called simplification we have to give a justification for any sentence. So we can only use valid rules of inference. five P ok again similar three and simplification. Six negation of not Q or S how do I get this I get this from four and two and a rule called modus tollens one of the rule set. So basically a proof system will have a set of rules of inferences and a lot of time is spent by the logicians in trying to figure out what rules of inferences do we need for a system to be complete. So if you have more rules of inference you can possibly get shorter proofs but then the number of choices become large larger at the same time. So there is a tradeoff between having more rules and less rules.

(Refer Slide Time: 23:19)



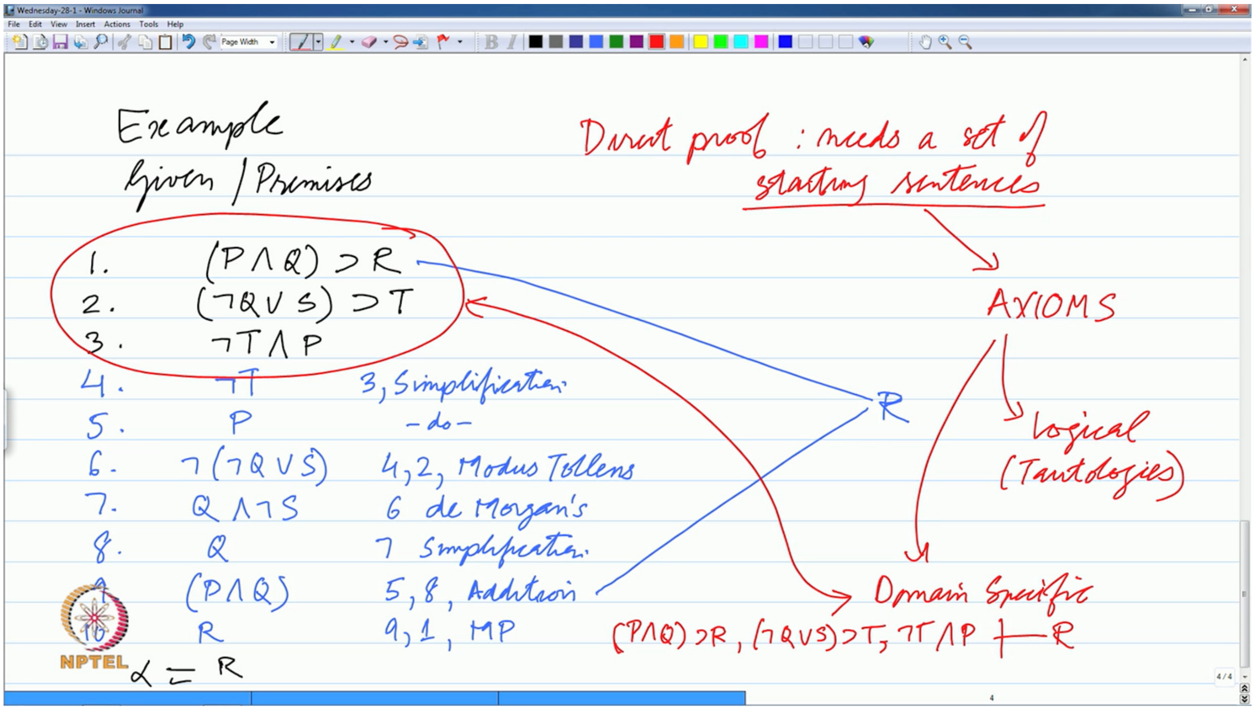
So then we get this so we push this inside we get Q and not S which is from 6 and de morgan's which I have not spoken about but any textbook on logic will tell you that. When you push the not inside which said that not not Q should be Q. from this we will get Q which is 7 and simplification. Then I can get P or R because I have P which is given in sentence 5 and I have Q which is given in sentence 8 and there is a rule called addition and finally I get R because we have 9 and 1 and our favorite rule which is modus ponens. So I have managed to add some formulae to knowledge base so not T P to knowledge base and in the end I ended up adding a formula which is the formula you were interested in which is R. so you can also write this is form of a tree as I said a natural deduction tree so for example at some point you would say that from this sentence and this sentence I would get R and other things you can actually construct a tree from the sentence which end up in R. so this is the basic idea of direct proof.

(Refer Slide Time: 25:18)



Now in this we assume that something was given to us and then from there we move forward. So one requirement of direct proofs is needs a set of starting sentences because if you know the application of the rule says that pick some rule with the matching data where is the matching data it must be there in your knowledge base so you must have some knowledge base to start with to which you can apply this. Now there are two kinds of so the starting sentences they are called axioms by axioms we met something that we expect as true. And there are two kinds of axioms one is logical which basically means that they are tautologies and other is kind of domain specific which are given to us by domains. So these domain specific axioms are the ones we started of with we accepted that this is true because somebody said this is true and in any case we are interested in the entailment question so what is the entailment question we are interested in is that this set of sentence which is let me write here P and Q implies R comma not Q or S implies T comma not T and P entails R. this is the question you are interested in. Of course this is not entailment.

(Refer Slide Time: 27:34)



This is a derivation but we are interested in those machines where derivations will always correspond to entails we have said that we are interested in machine which are both sound and complete. So if my machine produces R as it is done here then R better be true otherwise what is the use of having such a machine. We need a machine which is sound. Okay so in the next class we will look at some of the early logical systems where the axioms are not tautologies and we will show that even working with domains for problems like this can be reduced to looking for tautologies. That's why in the beginning of class I said we are interested in tautologies. If you can solve find all tautologies, we can find problems like this. Okay so we will do that in the next class.