Artificial Intelligence: Knowledge Representation and Reasoning

Prof. Deepak Khemani

Department of Computer Science and Engineering

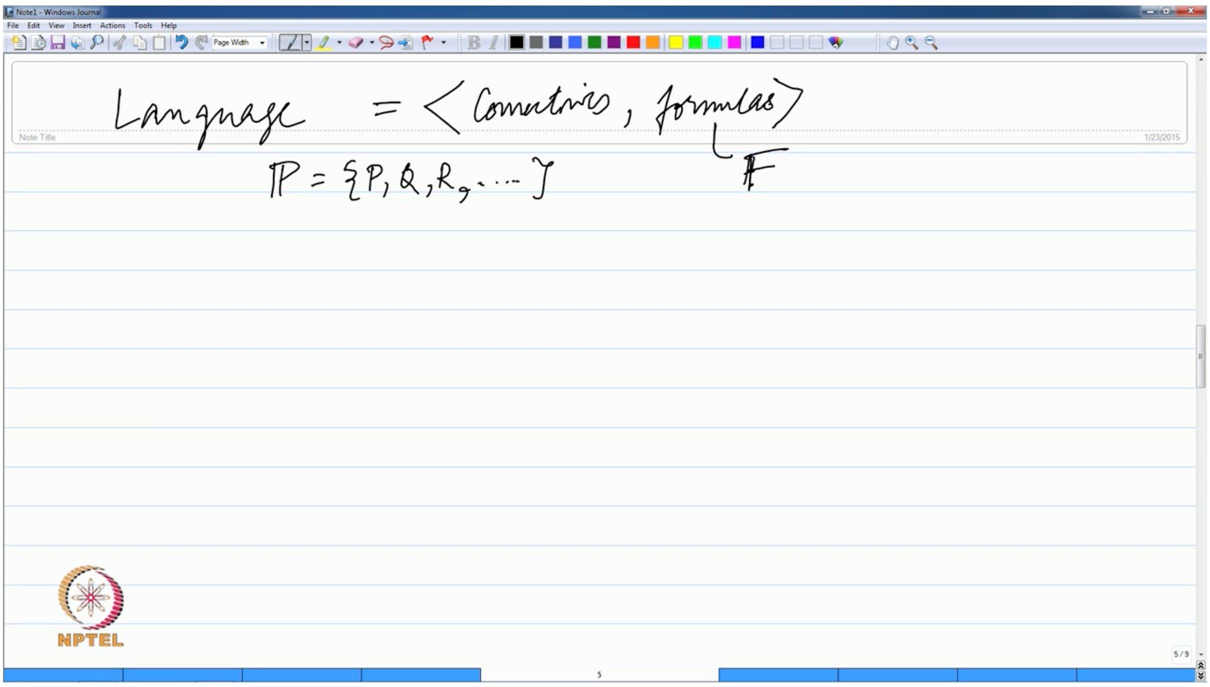
Indian Institute of Technology, Madras

Module – 02

Lecture - 02

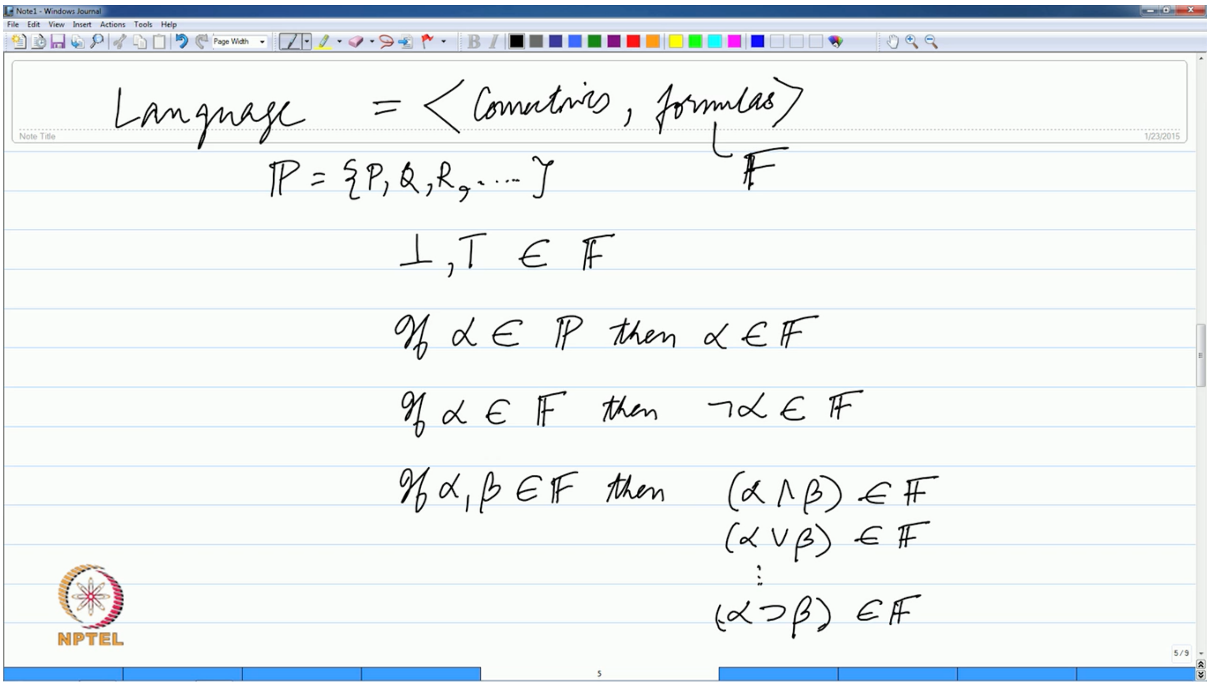
We were looking at the language of propositional logic while looking at the syntax of the language and we had seen in the last class that the alphabet of propositional logic consists of two parts in fact the alphabet of any language consists of two parts; one is the logical part which is kind of domain independent and common to all logics and the other is non logical part which is; which is the part which I stalking about the domain essentially. In our case since we are talking of the propositional logic language the domain dependent part is basically a set of symbols which will stand for certain statements in the domain essentially; and these symbols are the atomic sentences. So when we look at the language, we define this as a pair where we choose a set of connectives; as I said you can choose the subset of connectives and a set of formulas. So the set of formulas is chosen by first choosing a set of atomic formulas and then defining the rest of the language essentially. So let's call the set of formulas F we want to define what this set it essentially. And let's call this set of atomic formulas as P; that's given to us; that's the basic unit essentially.

(Refer Slide Time: 02.19)



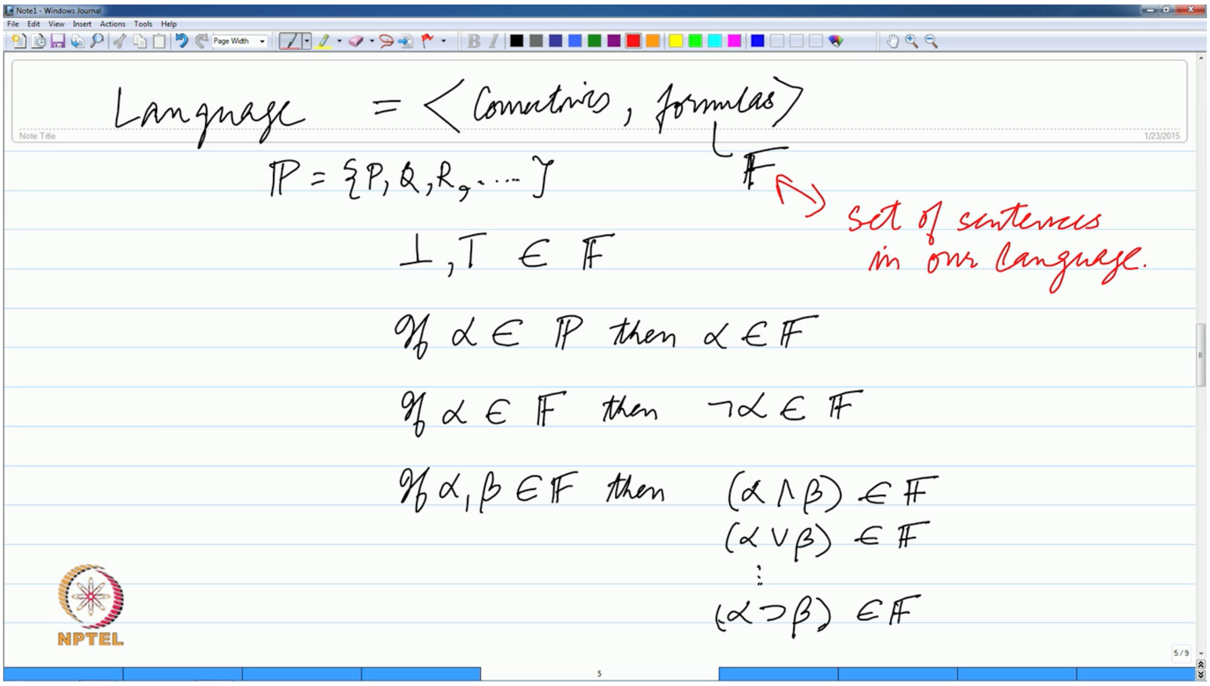
So this set F is defined as by structural recursion as follows; that these two symbols propositional constants that we had they will belong to the set of formulas of a propositional language F; by definition they are always there; then if anything belongs to the set of atomic formulas than that thing belongs to the set of formulas so in other words all atomic formulas are formulas and all atomic symbols are formulas. And if alpha belongs to the set of formulas then the expression not alpha also belongs to the set of formulas so this symbol, this unitary connective we are talking about; we will call this as not, and finally if alpha and beta belongs to the set for formulas then; we often use brackets but we don't always need to, so for every binary connective that we have in our language we can take up two formulas and connect it using that binary connective and produce the new formula essentially so in this way the set of formulas is constructed. What is the size of this set supposing the size of P is five; that means I start with five statements and I want to do some reasoning with those five statements. The set of formulas that you can construct is essentially infinite because you can always keep constructing bigger and bigger and bigger formulas which will combine them using different.

(Refer Slide Time: 4.42)



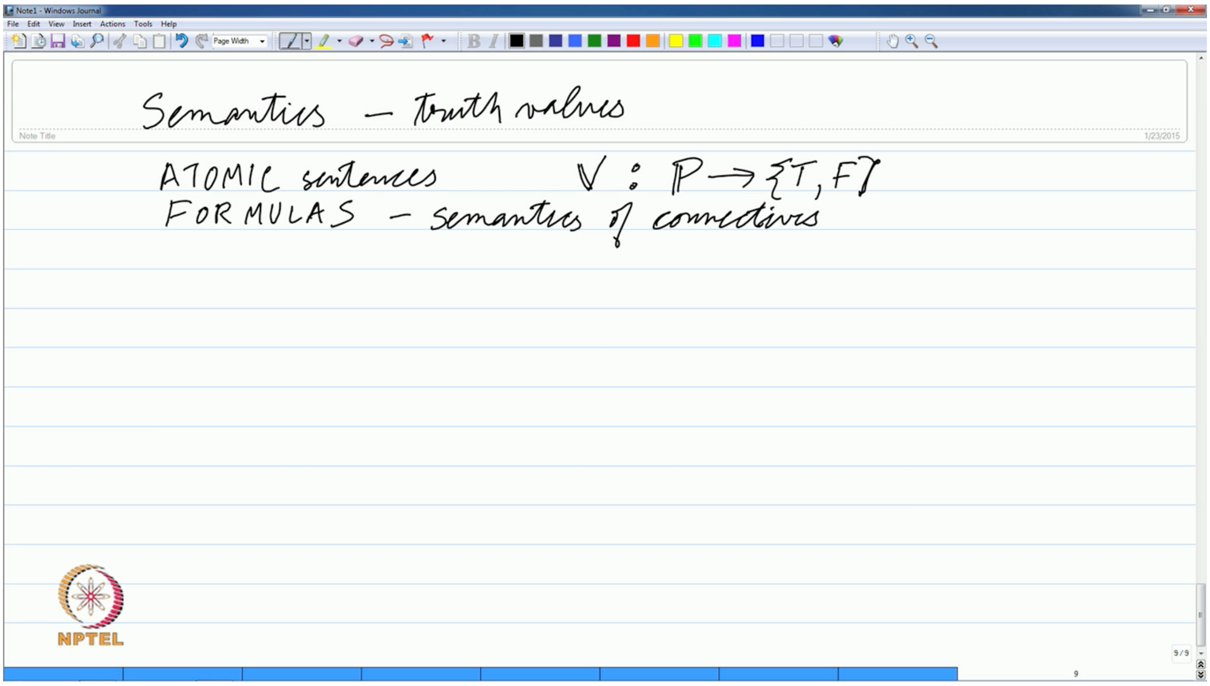
So this set F is essentially is the set of sentences in our language and we are interested in knowing which of these sentences are true and which of these sentences are false.

(Refer Slide Time: 5.21)



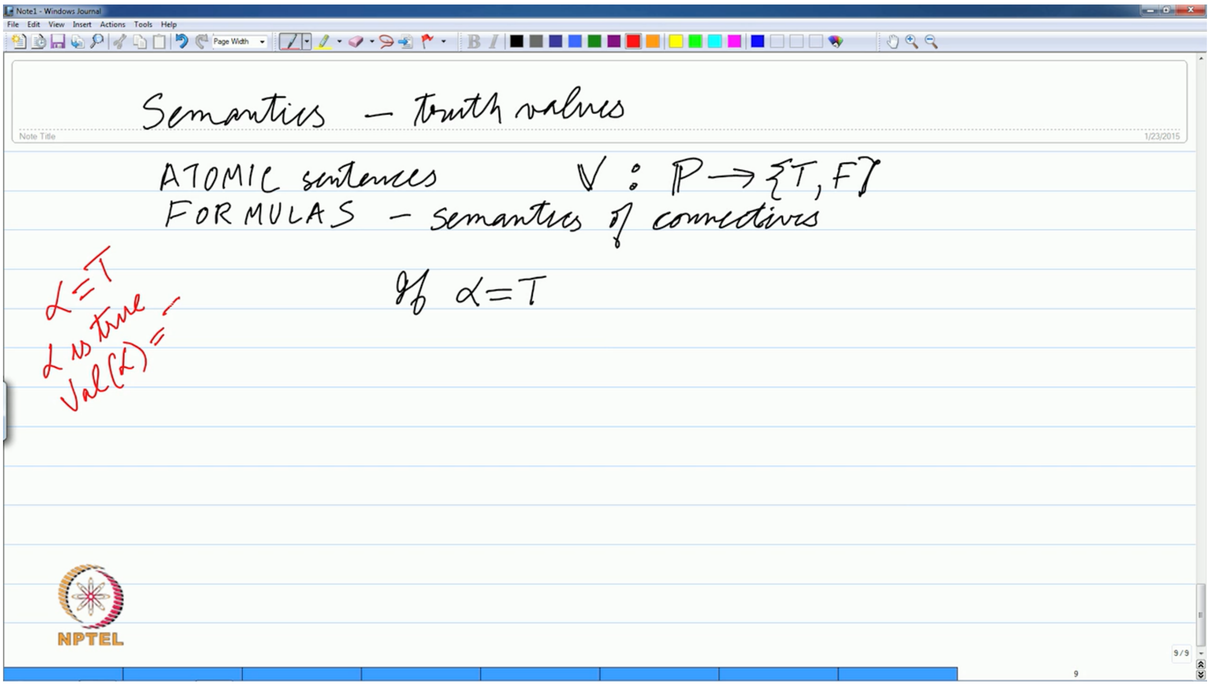
So let's talk about the truth values of sentences or semantics which is concerned with truth values. As we said a little while earlier that there is one notion of semantics which is as to what does is mean but in our case of propositional logic there is nothing really complicated like that because we have said that these are propositional variables and you can actually plug in anything you want. He can stand for anything you are interested in essentially. What we are interested in is that when we construct larger and larger compound sentences which of them will be true and which of them will be false. So for atomic sentences propositional variables we have a valuation function; let's call it v which maps a set of atomic formulas; two values set, so let us use this set T and F so throughout the course we will use this two values set to stand for truth values. As far as between us we will call the first element as true and the second element as false. And we will try to associate these two values with our domain. So whatever we think is true in our domain we will want to map to this symbol T and whatever is not true we will map to the symbol false essentially. So for atomic sentences the valuation is given externally; somebody tells you which statement is true which statement is false. So you cannot determine it.

(Refer Slide Time: 8.31)



But for rest of the formulas we determine the truth values using the meaning of the connectives. So we will make statements like this that if alpha is equal to T. This is the short form; I don't know whether we should use or not; let's assume we will use this short form; so this short form alpha is equal to true essentially stands for alpha is true. To be more precise we should have said something like val alpha is equal to true.

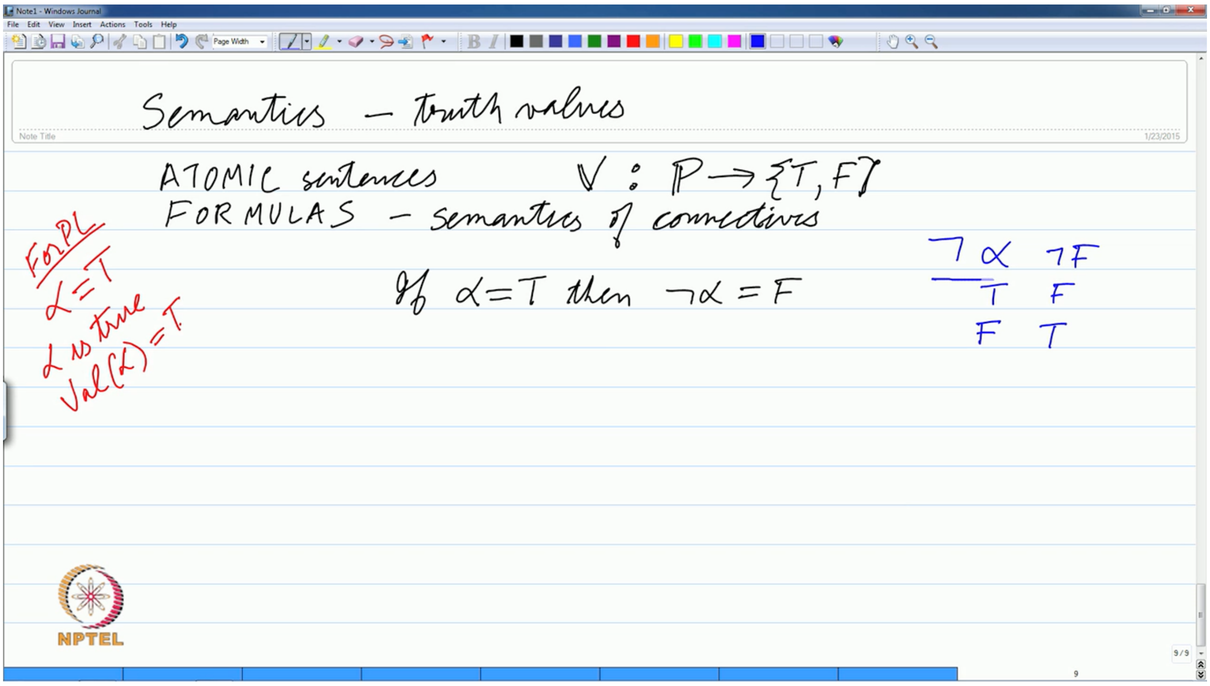
(Refer Slide Time: 9.17)



But since we are dealing with the propositional logic we will use the short form at least for the time being. When we will move to the first order logic we will see that the equal symbols has its own meaning that will be the part of the language and therefore we have to you known give a different.

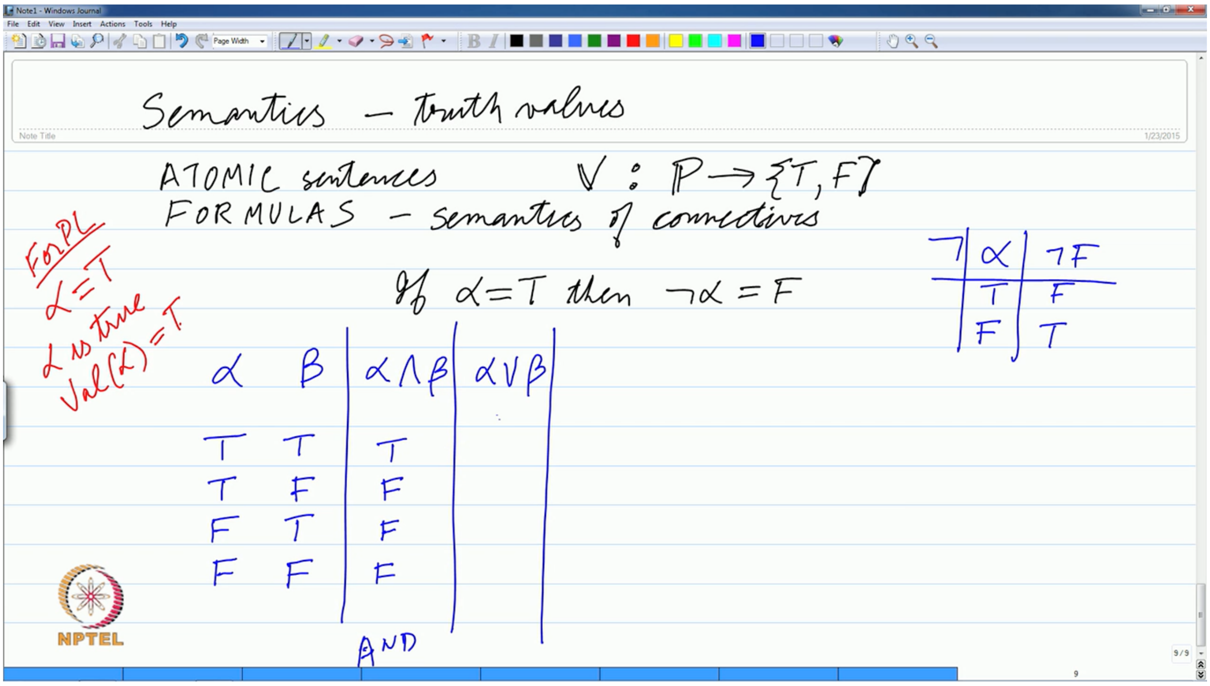
For this we will do for PL only. If alpha is equal to true then not alpha is equal to false. So this gives us how to arrive at a truth value of a sentence which is made up using the negation sign. You are familiar with this idea of truth tables so we can construct a truth table for negation, so we can say alpha is true or false then not alpha is false or true

(Refer Slide Time: 10.37)



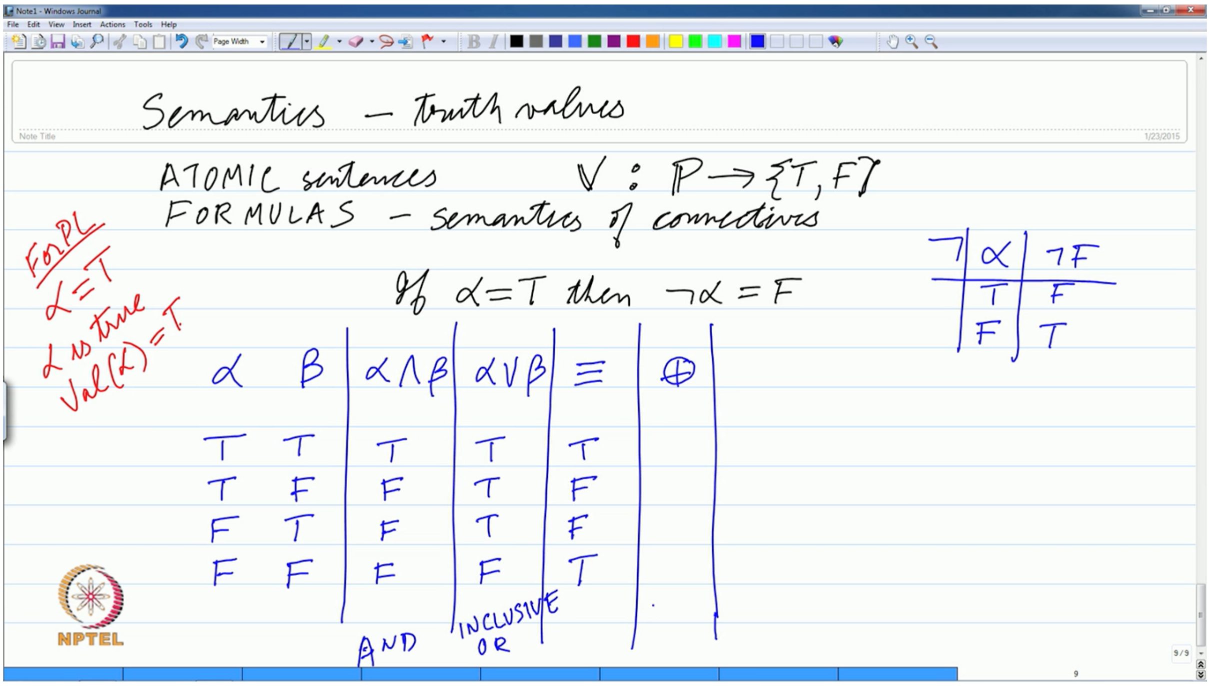
Likewise, we can construct truth tables as you are familiar with for other connectives so let me do a few examples so the other connectives are binary connectives so they take two values. So let's say alpha and beta and the values are either true true false false or true; these are the four combinations of values that we can look at and we can define different connectives so what is the meaning of alpha and beta so no doubt you are familiar with this. We read this as and the semantics is defined as both the constituents must be true so whenever both are true this is true, in all other cases this is false.

(Refer Slide Time: 12.00)



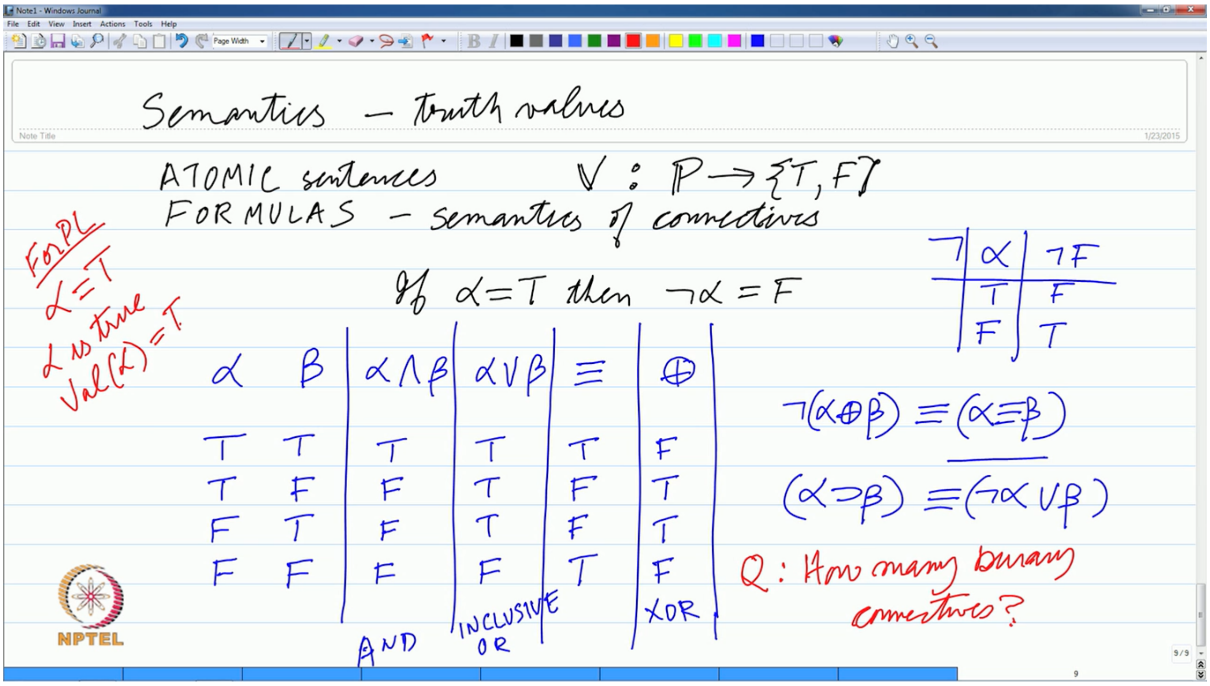
Alpha or beta this is or and it's the inclusive or so if either one is true it is true else it is false. Let's do a couple of more, equivalence, so when I write equivalence it means alpha equivalent to beta so both must have the same value in which case it is true and both have the same value it is true and otherwise when they have different values it is false . Exact opposite of this is the symbol or ex or or exclusive or which is true only when they are both different and its false otherwise so you can see that between xor and equivalence, they are the complement of each other.

(Refer Slide Time: 13.02)



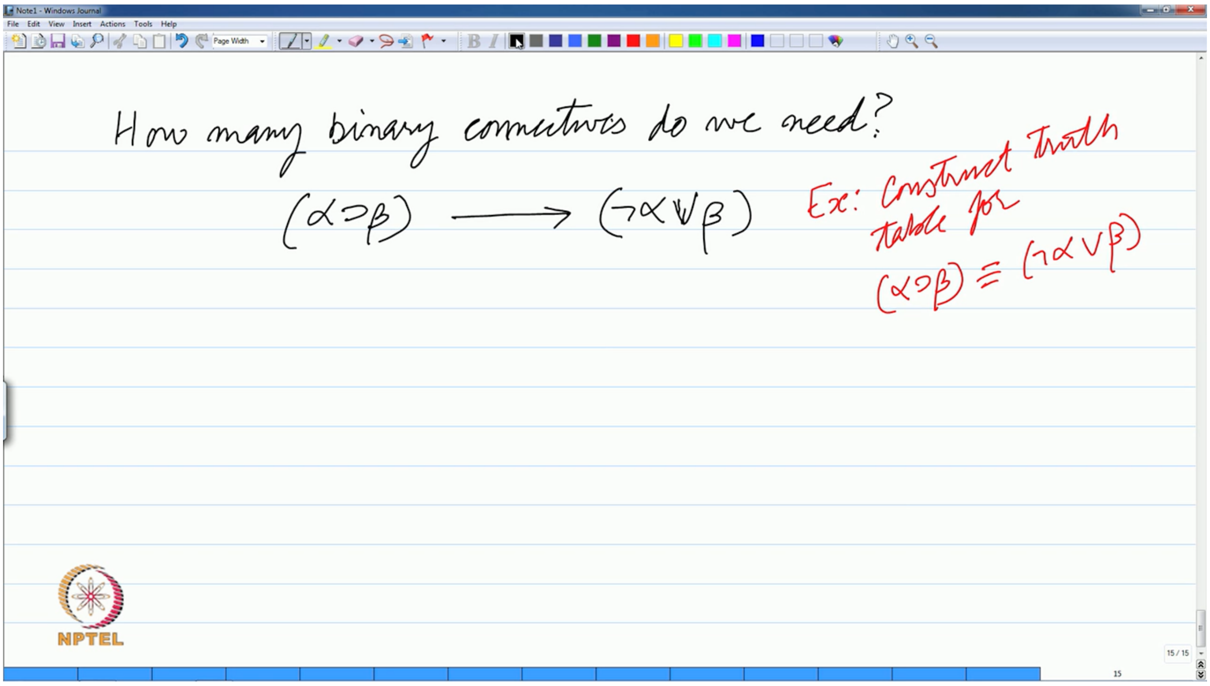
From this you can kind of guess that if you are having the formula of the kind not of alpha xor beta then this must be logically equivalent to alpha equivalent to beta so the point of this exercise is to show that you can say same things in different ways even in the language like logic. You want to say that both are; both alpha and beta take the same truth value you can either use this simple expression which says that a alpha is equivalent to beta or you can say it in a slightly roundabout way which is that not alpha xor beta. One equivalence that we will often use is that instead of saying that instead of saying that alpha implies beta we can say not alpha or beta. So I have not defined implication here but I will leave it as a small exercise for you to define implication and the other this thing. So the question we can ask that how many binary connectives are there? Because you can fill in this column in sixteen different ways we can actually have sixteen different binary connectives and the next question is do we need all of them essentially?

(Refer Slide Time: 15.10)



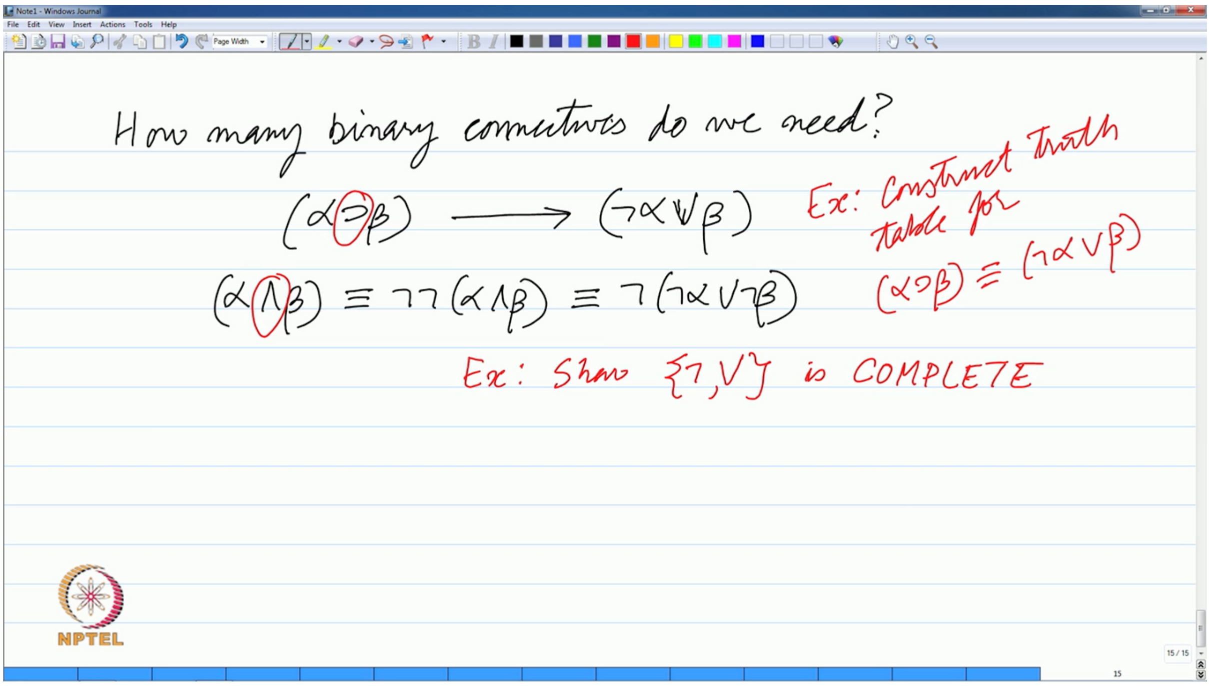
So the question we are asking is how many binary connectives do we need? SO we just saw an example in which we said that if you have alpha implies beta than you can actually replace it with not alpha or beta. They will have and you should do as a small exercise, construct a truth table

(Refer Slide Time: 16.30)



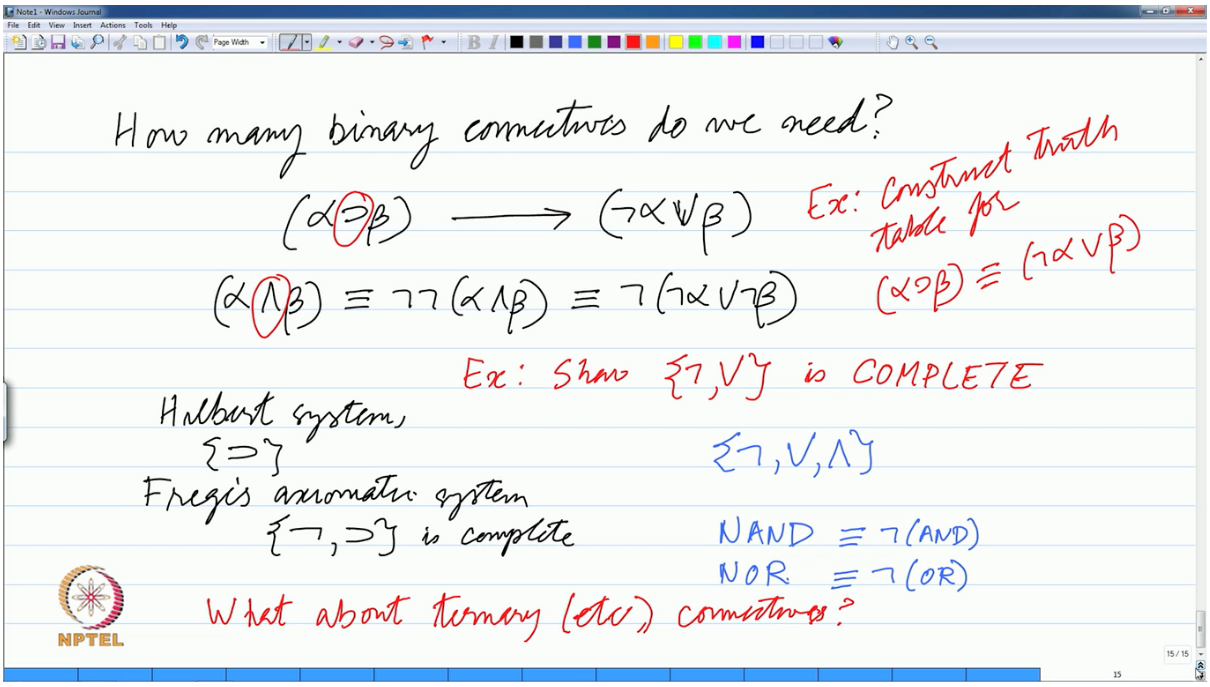
Likewise if we have alpha and beta then we can actually replace it by so let me use the standard notation is equal to not not alpha and beta which is equivalent to not not alpha not beta so the interesting thing is we can do away with these two symbols we can do away with implication and and whatever you wanted to say with implication, whenever we wanted to say alpha implies beta instead we could have have said not alpha or beta likewise whenever we wanted to say alpha and beta we could have said not of not alpha or not beta. So the exercise is show that the set of negation and or is complete. What do we mean by complete is every that can be said by other connectives can be said using these two connectives?

(Refer Slide Time: 18.08)



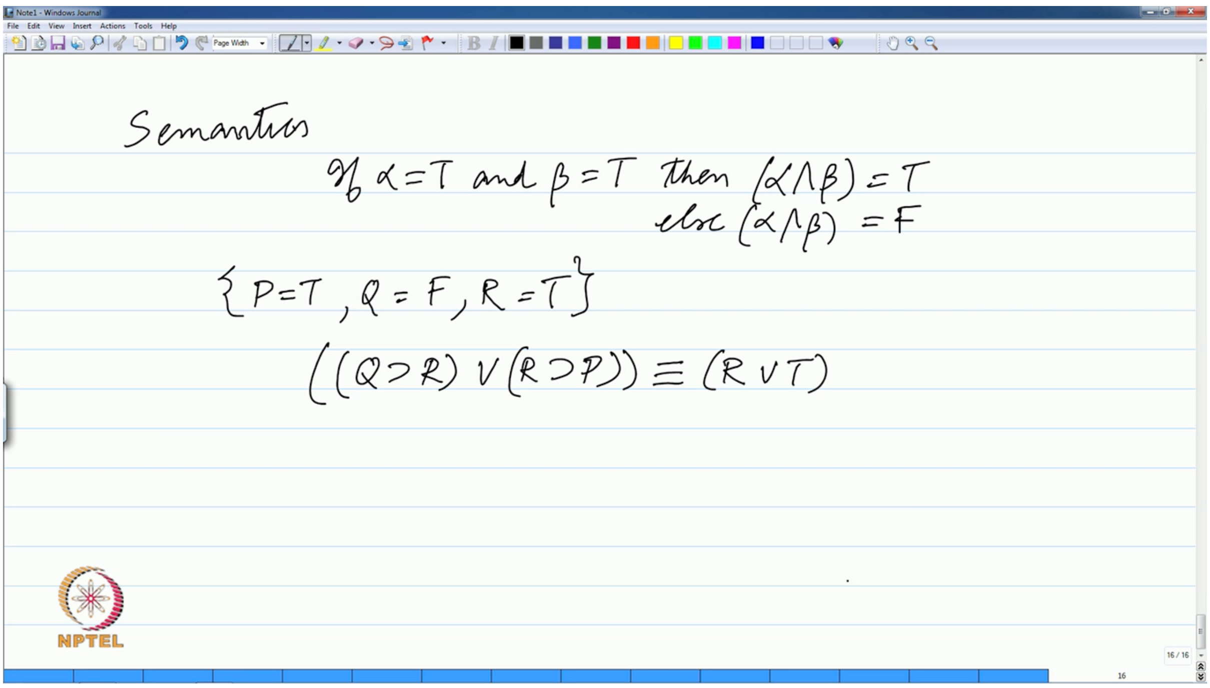
So there is a notion of completeness in the choice of logical connectives that we use and what people were trying out long time ago was trying out different combinations of connective and trying to say which is a good set to work with essentially. So we will see a little bit later when you look at Hilbert systems, they work only with implication essentially. Only one thing. But the thing is that implication is not complete so you can do certain things with Hilbert system but then we will look at also Fregis axiomatic system and Fregis actually showed that this set of negation and implication is complete which means that anything you want to say using any of these binary connectives and negation can be said only with negation and implication. So other sets which were of interests were which we will see later is this set of negation or and and which we will use in one of our proof procedures. Then you would have heard about NAND and NOR, these were known as sheffer stroke and Pierce arrow because they were shown by sheffer and pierce. These are single connectives. This NAND is equal to NOT of and, so if you take and truth table and just flip it you will get nand and nor is not of or and it has been shown and those of you who have studied Boolean circuits would know that the nand gate and nor gate are enough to construct any circuit that you want which also means that anything you want to say in propositional logic can be said just by the nand gate and the nor gate. Only one connective is enough essentially. One more question I want to ask is what about let's say ternary connectives and other higher order connectives? Do we need them? Do we need something which will say that I want to connect three formulas to form one compound formula? Because all the connectives we have seen are binary connectives, they take two formulas and give you one compound formulas, what about taking three at a time or four at a time or five at a time. Can we say something more than with those things? Again we will not go in the details but it has been shown that everything you can say with higher arity connectives can be said with binary connectives and everything that can be said with binary connectives can be said with any of these sets which is complete. So not and or is complete, not or and is complete, nand is complete, nand is complete nor is complete. So we can use any of them.

(Refer Slide Time: 21.58)



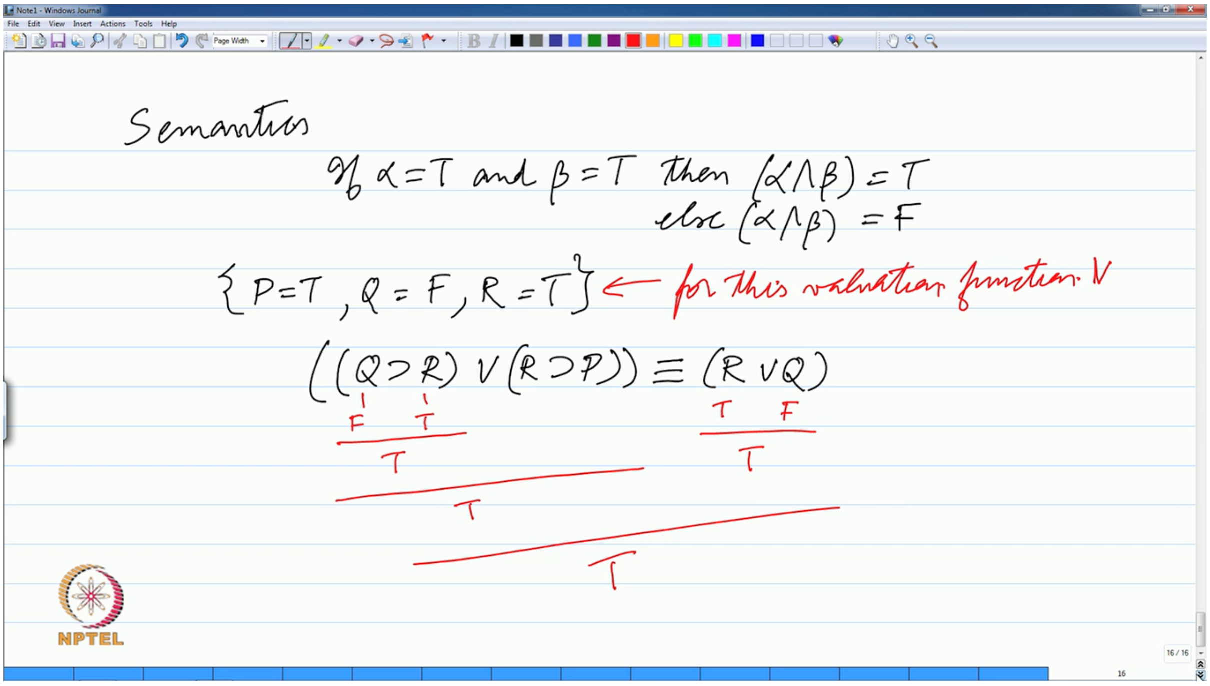
So what we have done now is to define a language and we have said we started with an alphabet, the alphabet consists of binary connectives and as we said we can choose a subset plus a set of propositional symbols and then through structural recursion we have determined the set of formulas that can be expressed in the language essentially. Then we looked at the semantics and we said as to how do you determine the truth value of a sentence essentially. We didn't write it explicitly but we will have statements like if alpha is equal to t and beta is equal to t then alpha and beta is equal to t else alpha and beta is equal to false, we could have said statements like this. So it means that for any given formula you can determine its truth value based on the truth value of the inputs. So let us say that we have p is equal to t and q is equal to false and r is equal to t essentially; and this is the set we are given, so we are given three propositional variables and we are giving a valuation function which says p is true and q is false and r is true, then we take an arbitrary formula q implies r and r implies p and the whole thing is equivalent to r or t, some random formula I have written. The question I would like to ask is and you know that we can construct an infinite set of formulas like this. So given any arbitrary formula like this the question we are interested is this formula true or is this formula false, based on the fact that we have been told that p is true an q is false and r is true essentially.

(Refer Slide Time: 24.35)



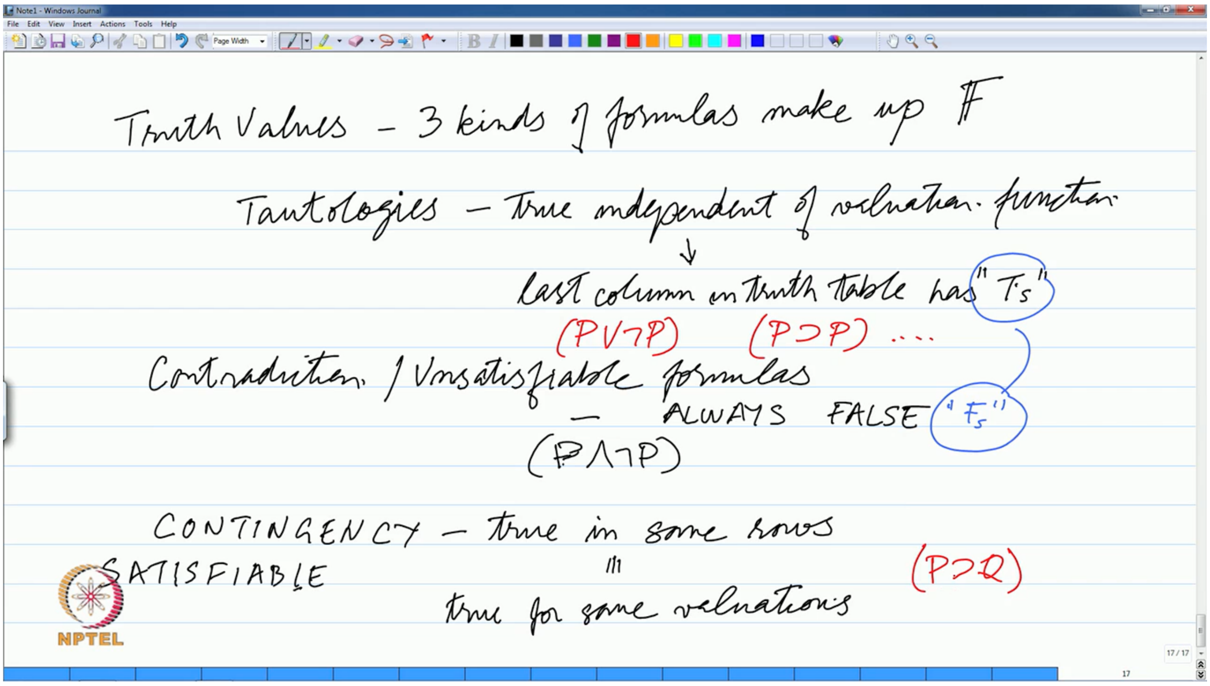
So we can now of course break it down. We can now see that r is true, sorry I don't have q, I could have used the tau symbol but let's just use something simpler q, so we said that this is false and r is true so this formula by the semantic of or becomes true because at least one should be true and q is false and if you look at the truth table of q you will see that whether r is true or not doesn't really matter so this becomes true. Since this is true the whole formula is true irrespective of what is the valuation of r and p because of the or connective here and because this is true and this is true, this whole formula becomes true. So we can take any arbitrary formula and evaluate its truth value by plugging in the value of that essentially. But this is dependent upon the fact that we have been given a valuation function for this valuation.

(Refer Slide Time: 26.37)



So in general there are three kinds of formulas. Remember that we have defined this set of formulas given a language essentially. This set is partitioned in to three kinds of formulas. One is called tautologies which are always true and this is determined by the last column in the truth table has all Ts essentially. If you construct the truth table for that particular formula, then the last column in the truth table will always be Ts and such formulas are called tautologies. The examples are, no doubt you are familiar with such things, p or not p, obviously since one of them have to be true or p implies p which you will recognize as the same formula said in the different way and so on. There can be many many tautologies. Then we have contradictions or unsatisfiable, these are formulas which are always false, when you look at this statement here for this one, it will be all fs. In unsatisfiable it will always false irrespective of what valuation you choose for the constituent atomic sentences, that formula will always be false and again you are familiar with formulas like this p and not p is a classical example of a contradiction essentially. The third kind of formulas are known as contingencies. True in some rows which is equivalent to saying true for some valuations because each row in truth table corresponds to a valuation, you are saying p is true or q is false or r is true or false. Each row corresponds to a valuation and contingencies are formulas which are true for some rows and not all rows essentially. We also call them as satisfiable formulas. Strictly speaking the set of satisfiable formulas also include tautology. But we can find a valuation that can satisfy the formula and example of that is p implies q a simple formula which will be true in some rows and false in some rows as indicated by the truth table essentially.

(Refer Slide Time: 31.25)



Ok so this defines the syntax and semantic of propositional logic. In the next class we will look at proof methods essentially. How do we arrive at these truth values through different process which is the proof process essentially?