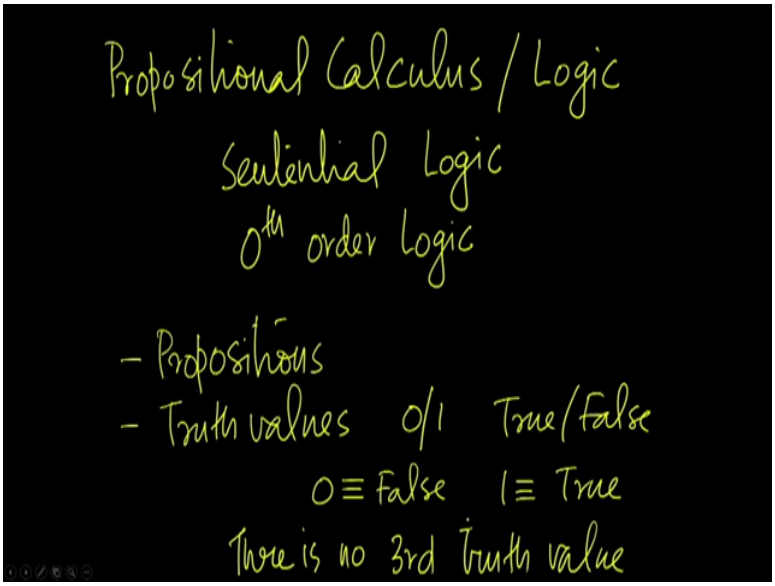
Discrete Mathematics
Professor Sajith Gopalan, Professor Benny George
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati
Mathematical Logic
Lecture 1

Welcome to the NPTEL MOOC on discrete mathematics.

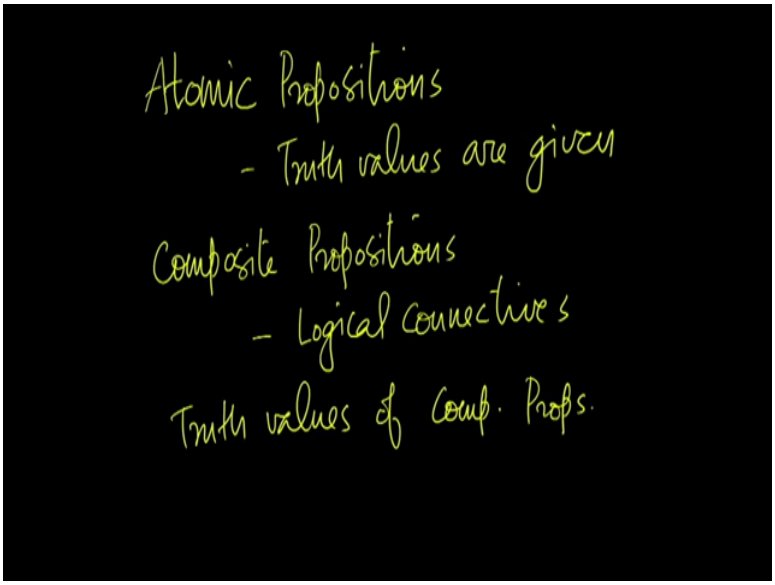
(Refer slide time: 00:43)



We begin our study of mathematical logic today and we begin with propositional calculus. Propositional calculus is also called propositional logic, sentential logic, 0th order logic. As supposed to first-order logic which is a richer logic and we shall study late. In propositional calculus, we deal with propositions, these propositions take on truth values, the only semantic entities that we deal in propositional calculus are truth values.

The truth values could be 0 or 1 or true or false, 0 corresponds to false and 1 corresponds to true, there are only two truths values, there is no third truth value.

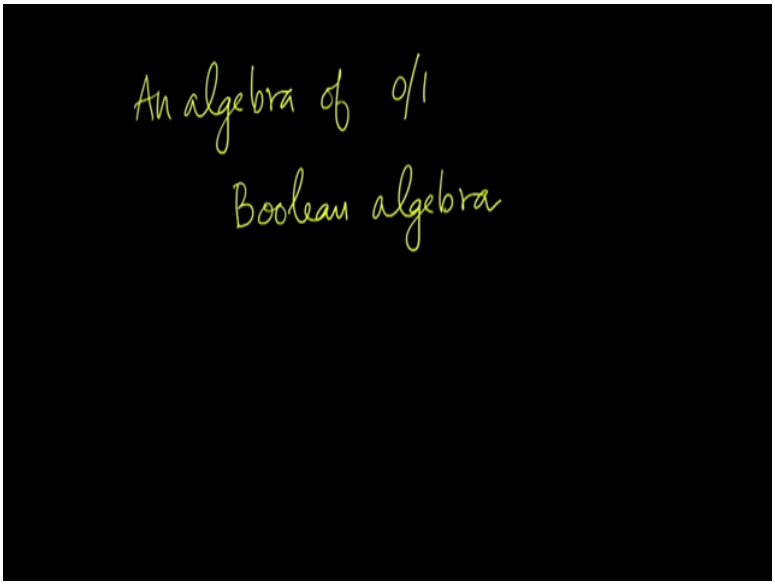
(Refer slide time: 02:12)



Propositions take on truth values as I mentioned, there are, what are called atomic propositions? For these atomic propositions the truth values are given, that is we do not analyse these atomic propositions to find out how they became true or false, we just know that they are true or false and then we can combine atomic propositions to form what are called composite propositions.

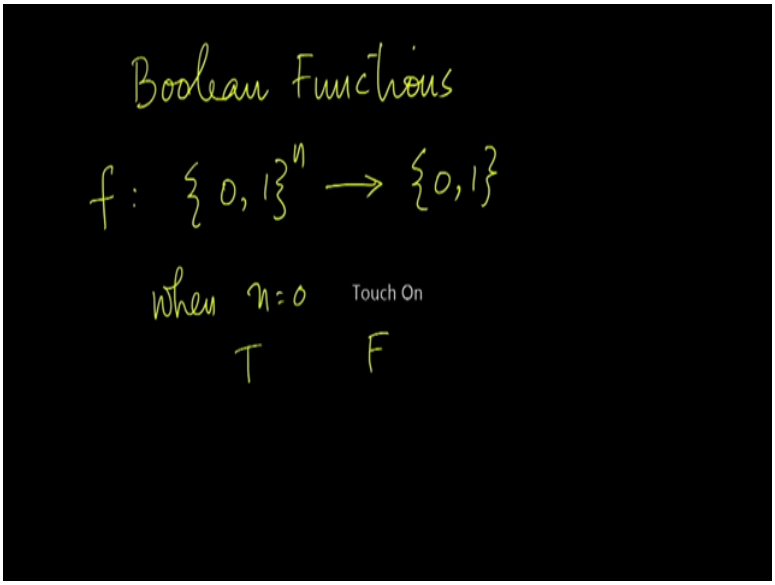
Composite propositions are made using atomic propositions and these atomic propositions are combined using logical connectives, that is composite propositions are synthesized from atomic propositions using logical connectives. We can compute the truth values of composite propositions using the truth values of its constituents and also the properties of logical connectives.

(Refer slide time: 03:36)



This essentially gives us an algebra of truth values, algebra of 0 and 1, this algebra is called Boolean algebra after mathematician George Boole. So, let us begin with a study of this algebra, Boolean algebra which is the underlying algebra of propositional calculus.

(Refer slide time: 04:05)

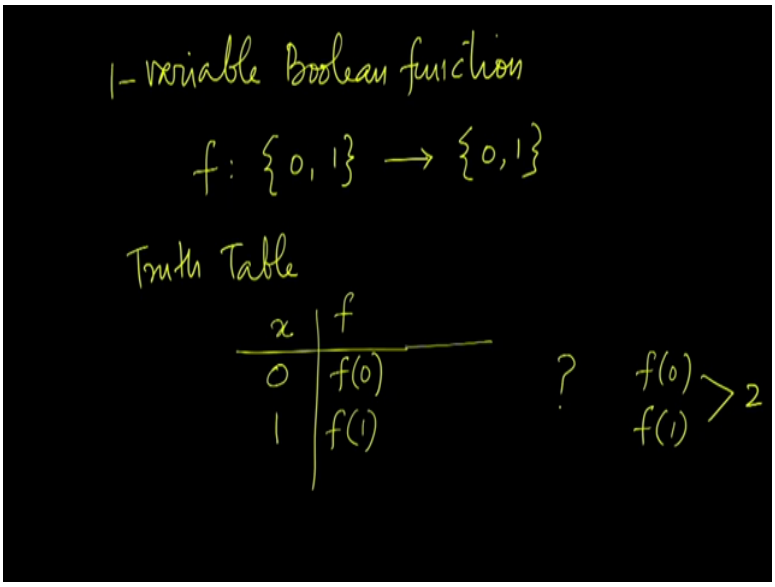


So, in Boolean algebra, we have what are called Boolean functions? A Boolean function has a signature of this form, a Boolean function is a mapping from truth values to truth values. So, a Boolean function with n inputs, an n variable Boolean function takes n truth values as inputs and produces one truth value as the output. so, let us consider truth values Boolean functions for various values of n when n equal to 0 that means when there is no input, there

are two Boolean functions, one is the constant Boolean function true which always produces one and then the constant Boolean function f which always produces 0.

So, these are the zero variable Boolean functions.

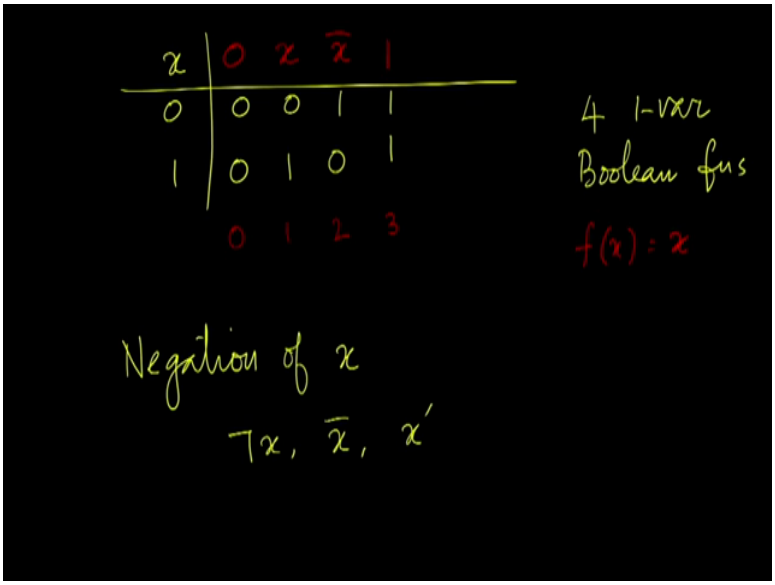
(Refer slide time: 05:19)



When we consider 1-variable Boolean functions, we are considering functions that take one Boolean value as input and produce one Boolean value as output, functions of this signature. Such a function can be specified using what is called a truth table. In a truth table, we list the inputs, so in this case there is only one input and specify the possible outputs. so, input here could be either 0 or 1, if the input is 0 what would the output be? If the input is 1, what is going to be the output? Once you specify these, we have the truth table for function f.

Now, how many such functions could be there, depending on how you choose f of 0 and f of 1, we will have different Boolean functions, which means there are two choices to be made, one choice for f of 0 and 1 choice for f of 1, each choice is going to be a Boolean value. So, there are going to be four such possible choices.

(Refer Slide Time: 06:38)



So, let us enumerate all such Boolean functions, when input x is 0 or 1, the output could be both zeros, we are choosing 0 for both f of 0 and f of 1, or we could choose 0 for f of 0 and 1 for f of 1, or we could choose 1 for f of 0 and 0 for f of 1, or we could choose 1 for both. So, these are the four possible choices for f of 0 and f of 1, each choice will define one Boolean function.

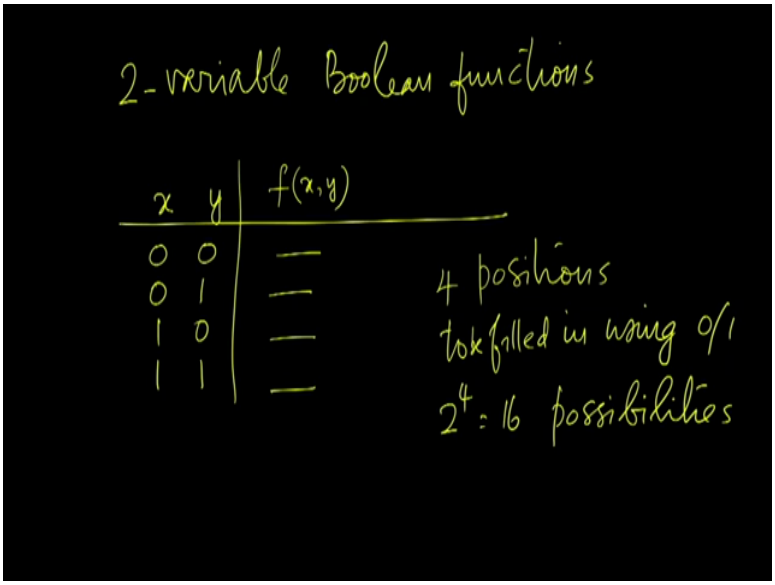
So, we have four 1-variable Boolean functions. Now, let us name these functions, in particular, let us look at the first column. In the first column both the function values are 0 therefore let me call this the zero function, the function which produces zero irrespective of the input value. Similarly, look at the last column here f of 0 is 1 and f of 1 is also 1, so this is a function which produces 1 irrespective of the input value, so let me call this function 1.

Now, the first column, these are the 0th and the third column respectively. So, let me consider this first column now which corresponds to output values 0 and 1. So, when you observe that these reproduce the input, we realize that this function is nothing but x, so this is the identity function, f of x equals x. The first column corresponds to the identity function, so let me call it x it reproduces x when you look at the second column you find that it complements x when x is 0 it produces 1 and when x is 1 it produces 0.

So, this is the complement of x, this we call x bar the negation of x, negation of x inwards the truth value when x is two it produces false and when x is false it produces two. Negation of x is denoted variously in these ways, so we will use all these notations interchangeably, so you

should remember these notations, negation of x could be represented in any of these ways. So, we have now seen four 1-variable Boolean functions.

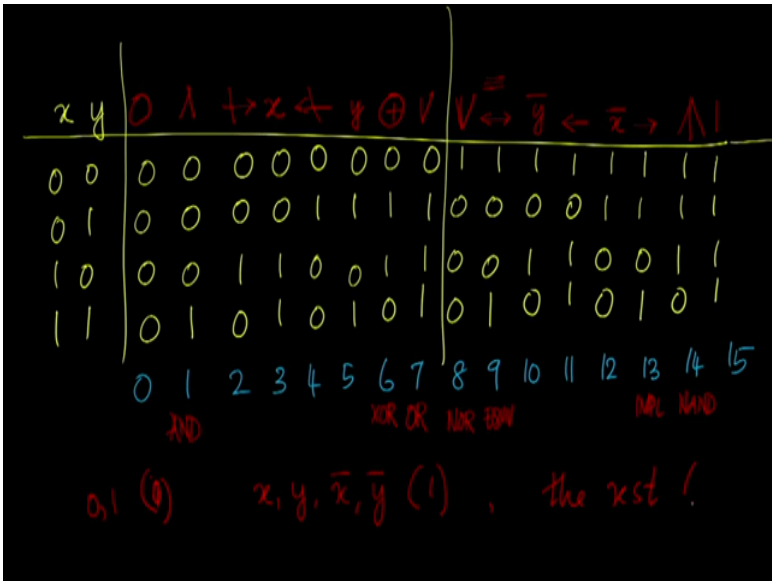
(Refer Slide Time: 09:24)



Now, let us go on to look at 2-variable Boolean functions, how many 2-variable Boolean functions could there be? Let us say x and y are the inputs to our 2-variable Boolean functions, since there are two Boolean inputs the possible inputs are 0, 0, 0, 1, 1, 0 and 1, 1, so these four are the form the set of all possible inputs. Now, what could the output be? f of x, y to specify f of x, y you have to specify a Boolean value at each of these four positions.

So, there are 4 positions to be filled in using 0 and 1, so there are 2 power 4 which is 16 possibilities. Let us look at all these 16 possibilities.

(Refer Slide Time: 10:52)



So, enumerating all these 16 possibilities, let us consider all possible 4 inputs 0, 0; 0, 1; 1, 0; and 1, 1, so there are 4 vacancies to be filled f of x, y could be any of these 16. So, the first one is 0, 0, 0, 0 and the next one is 0, 0, 0, 1, third one is 0, 0, 1, 0 and the fourth one is 0, 0, 1, 1 and the next one is 0, 1, 0, 0 then 0, 1, 0, 1; 0, 1, 1, 0; 0, 1, 1, 1 these form one half of the possibilities, the remaining possibilities will complement these.

Then we have 1, 0, 0, 0 then we have 1, 0, 0, 1; 1, 0, 1, 0; 1, 0, 1, 1; 1, 1, 0, 0; 1, 1, 0, 1; 1, 1, 1, 0; 1, 1, 1, 1, so those are the 16 possibilities. So, let me number the columns in this fashion, starting from 0 and going up to 15, there are sixteen such columns. Now, let us analyse these columns, look at the 0th column, the 0th column produces a 0 irrespective of the input whatever be the values of x and y the output is 0 therefore we call this the 0 function, its mirror image is column 15, in column 15 the output is 1 irrespective of the input value therefore this function we call 1.

Now, consider the first column, in the first column the output is 1 precisely when both the inputs are 1, if either x is 0 or y is 0 then the output is 0, the output is 1 precisely when x and y are 1, so this is called the AND function, which we denote in this manner, the AND function, this is one way of denoting an AND function. Its complement is column 14, in column 14 the output is 1 if either x or y is 0, this is the negation of the AND function, therefore we call it NAND, this is called the NAND function, column 1 is the AND function.

Now, let us look at column 7, in column 7 the output is 0 precisely when both the inputs are 0 or in other words, the output is 1 if either x or y is 1, therefore this we called the OR function,

column 7 is the OR function. The complement of column 7 is column 8, there the output is 1 precisely when both the inputs are 0, this is called the NOR function, the negation of the OR function which is denoted like this.

Now, let us come to column 6, column 6 is 1, if either x is 0 or y is 0 but not both that is exactly 1 of x and y should be 1 for the output to be 1 this is called the XOR function denoted in this fashion. The complement of that, is 1 precisely when both the inputs are same, that is when both the inputs are 0 the output is 1 when both the inputs are 1 the output is 1 but when one is 0 and the other is 1 the output is 0, so this function is called the equivalence function or the IF and only IF function denoted either this way or as an equivalence, this is the equivalence function.

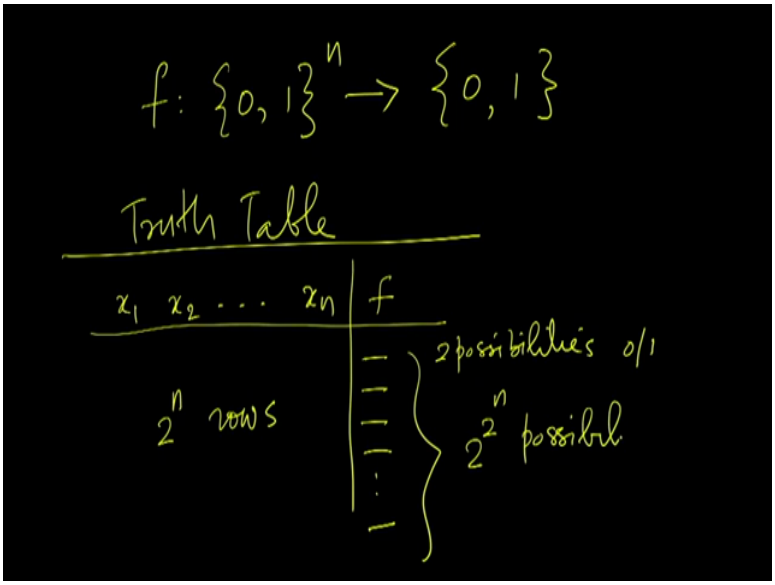
Now, consider column 13, column 13 is false precisely when x is true but y is false, this function is called the implication function, we say x implies y, we will have occasion to talk more about the implication function that is column 13. The complement of column 13 is column 2 which is the negation of implication which, we can denote in this fashion. Now, let us look at column 3, column 3 is 0, 0, 1, 1 column 3 reproduces the x column therefore this is the identity function in x.

Its complement is 1, 1, 0, 0 which is column 12, this is the negation of x. Now, let us look at column 5, which is 0, 1, 0, 1 which reproduces input y. Its complement is 1, 0, 1, 0 which is column 10 and therefore it is the negation of y, what remains now? Column 11, column 11 is 1, 0, 1, 1 therefore this is the reverse implication comparing it with column 13 you find that this is the reverse implication.

If column 13 is a forward implication x implies y then 11 has to be the reverse implication and then its complement 4 is the negation of the reverse implication. So, now we have named all the 16 functions. So, we find that some of these functions are not dependent on both the inputs for example, 0 and 1 are 1-variable functions, then x y negation of x negation of y are 1-variable function, sorry 0 and 1 are 0-variable functions.

x y negation of x negation of y are 1-variable functions the rest are 2-variable functions.

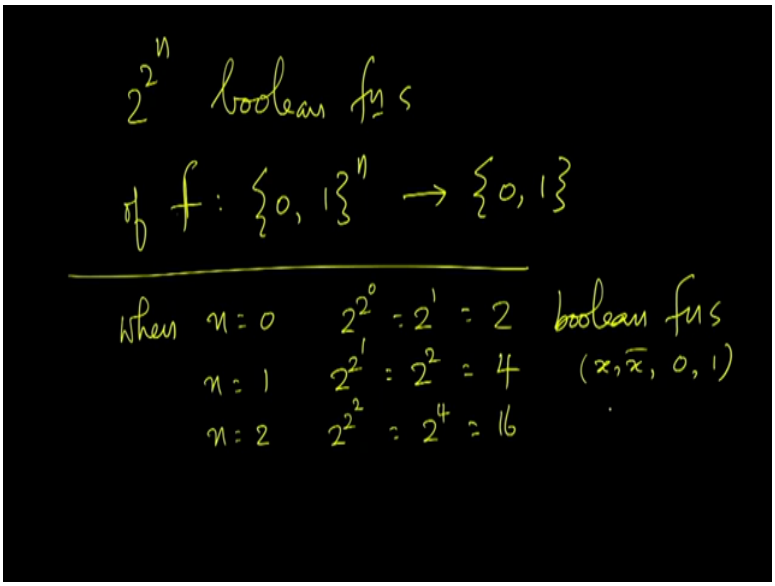
(Refer Slide Time: 18:34)



In general, when you have a function of this form to specify the function you can draw up, what is called the truth table. In the truth table, we list all the inputs to the function. So, here there are n inputs, we have one column for each of the inputs, each of the inputs can take on any of the truth values. So, there are 2 power n rows in this table corresponding to the various assignments of the truth values therefore when you specify the function you have to fill these 2 power n positions, at each position there are 2 possibilities, 0 or 1.

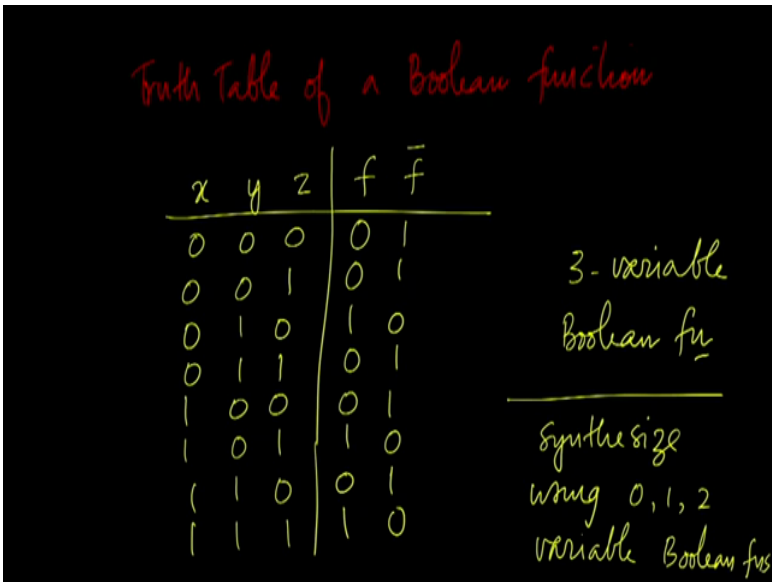
So, altogether there are 2 power 2 power n possibilities.

(Refer Slide Time: 19:50)



So, there are 2 power 2 power n Boolean functions of this signature. Special cases of this we have already seen when n equal to 0, there are 2 power 2 power 0 which is 2 power 1 equal to 2 Boolean functions, this are the zero function and the 1 function, when n equal to 1 we have 2 power 2 power 1 functions which is 2 power 2 which is 4, these are x the negation of x 0 and 1 and when n equal to 2 when there are 2 inputs we have 2 power 2 power 2 which is 2 power 4 equal to 16 Boolean functions, we have seen the list of them.

(Refer Slide Time: 20:52)

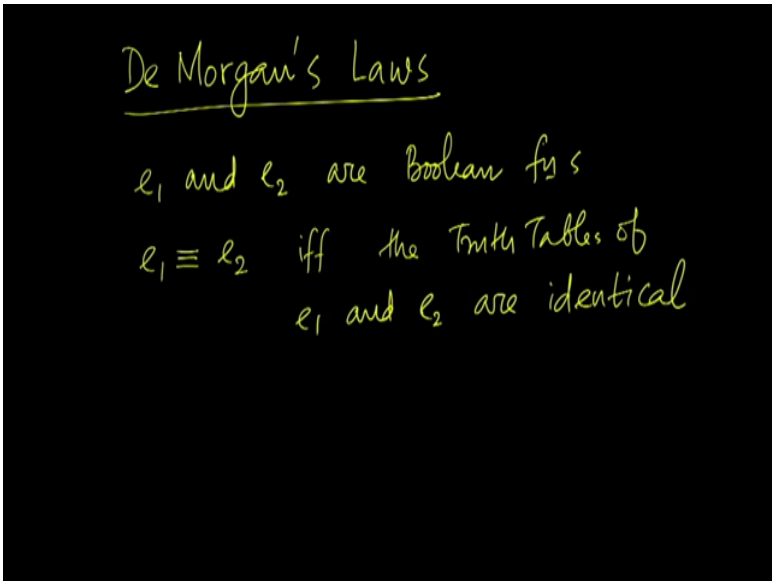


Now, consider the truth table of a Boolean function. Let us say, it is a 3-variable Boolean function, then there are eight rows in this truth table, these are the eight rows. Let us say, the function is specified in this fashion, the complement of f also we will specify. So, a truth table of a Boolean function is specified in this fashion, so this is a 3-variable Boolean function.

Now, let us post this question, can this 3-variable Boolean function be synthesized using 0, 1, 2 variable Boolean functions? So, this is the problem we want to address, we are given the truth table of a function. A Boolean function of Boolean function can be completely specified by specifying its truth table because this truth table now maps all the possible inputs to the corresponding outputs, so the function is completely specified by a truth table.

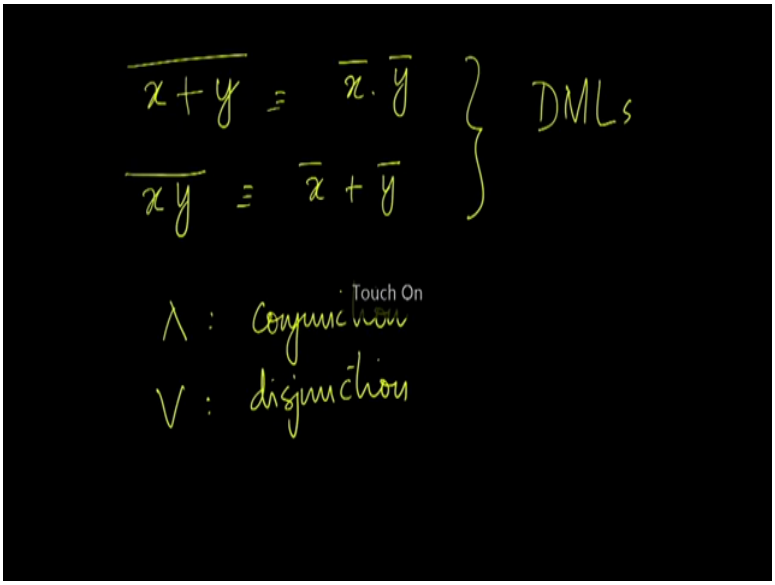
So, the function is specified using a truth table and we want to see, if the function can be synthesized using 0, 1 or 2 variable Boolean functions.

(Refer Slide Time: 23:00)



So, what will come in handy to do this, are what are called De Morgan's laws. For two Boolean functions e 1 and e 2, we say that e 1 is equivalent to e 2 if and only if the truth tables of e 1 and e 2 are identical, there is they should have exactly the same behaviour in terms of the output, that is when we say that two expressions e 1 and e 2 are equivalent.

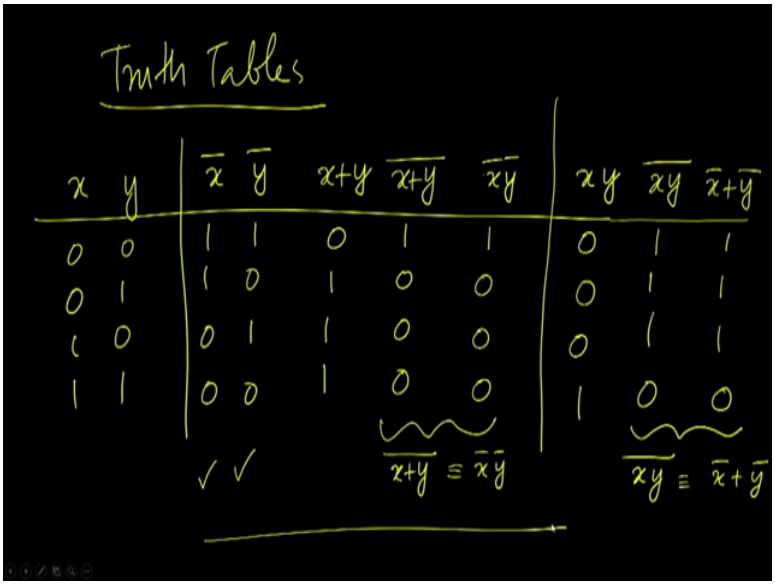
(Refer Slide Time: 24:03)



Now, De Morgan's laws specify certain equivalences, it says that for two Boolean variables x and y; x or y by the way or can be denoted in either of these two ways an AND can be denoted either as this or a concatenation in particular when I write xy I mean x and y. So, De Morgan's law, the first De Morgan's law says that the complement of x or y is the AND of x complement and y complement.

The second De Morgan's law says that, the complement of the AND of x and y is the OR of x complement and y complement by the way an AND is also called a conjunction and an OR is also called a disjunction, you should be familiar with these words. So, these are the two De Morgan's laws, we say that these two are equivalent, the left hand side is equivalent to the right-hand side but how do we show this? We can show this using truth tables.

(Refer Slide Time: 25:38)

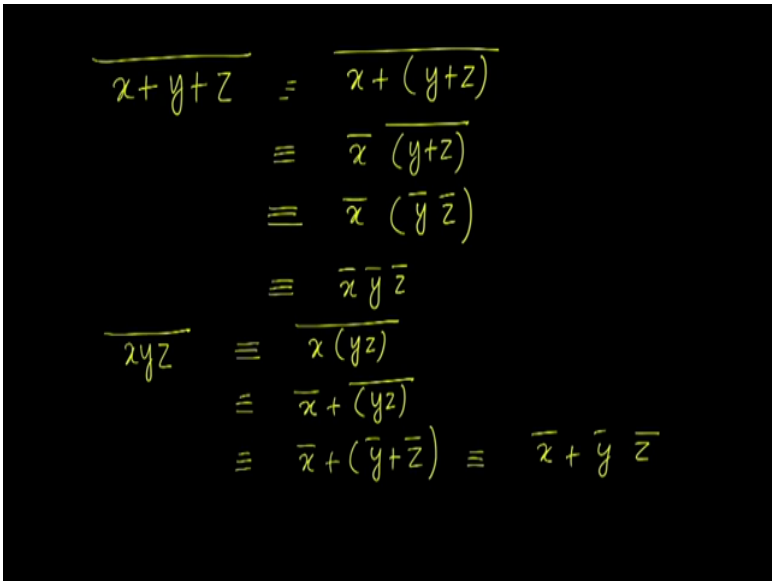


So, the two variable versions of De Morgan's laws can be shown like this, take the two inputs x and y and consider all possible inputs 0, 0, 0, 1; 1, 0 and 1, 1 then the complements of x and y would be these. Now, let us consider x or y, x or y would be these, what then would be the complement of x or y? That would be 1, 0, 0, 0. Now, De Morgan's law says that this is the same as the AND of x bar and y bar, 1 and 1 is 1, 1 and 0 is 0, 1 and 0 is 0 and 0 and 0 is 0.

So, you find that these two columns correspond exactly to each other therefore we have x or y the whole bar is equivalent to x bar y bar. Similarly, let us consider x, y; x, y is 0 here 0 and 0, 0 and 1 is 0, 1 and 0 is 0 and 1 and 1 is 1, so x, y is 0, 0, 0, 1. Then what would be x, y complement? That would be 1, 1, 1 and 0 and what would be x bar or y bar? We have x bar here and y bar here, if you take the OR of them you get 1 here, 1 here, 1 here and 0 here.

So, you find that they again correspond, these two columns correspond. So, the complement of x and y is equivalent to the OR of x bar and y bar. So, this truth table establishes the two De Morgan's laws for two variables.

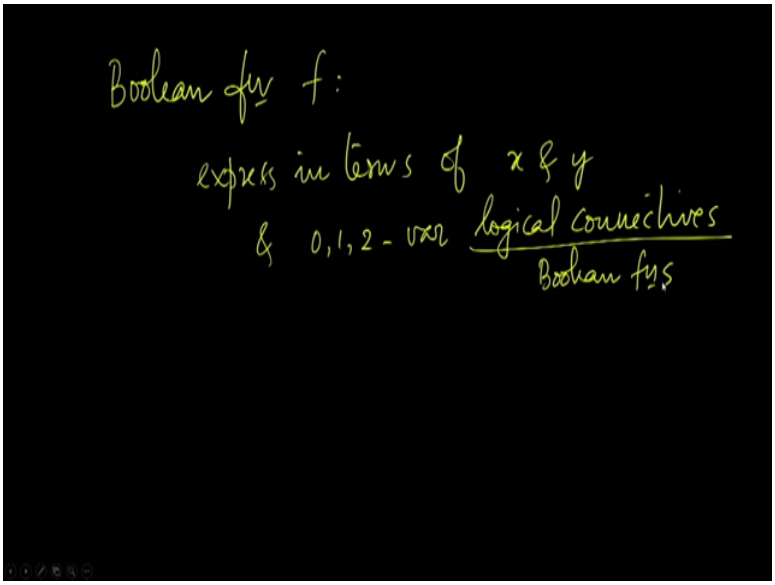
(Refer Slide Time: 28:09)

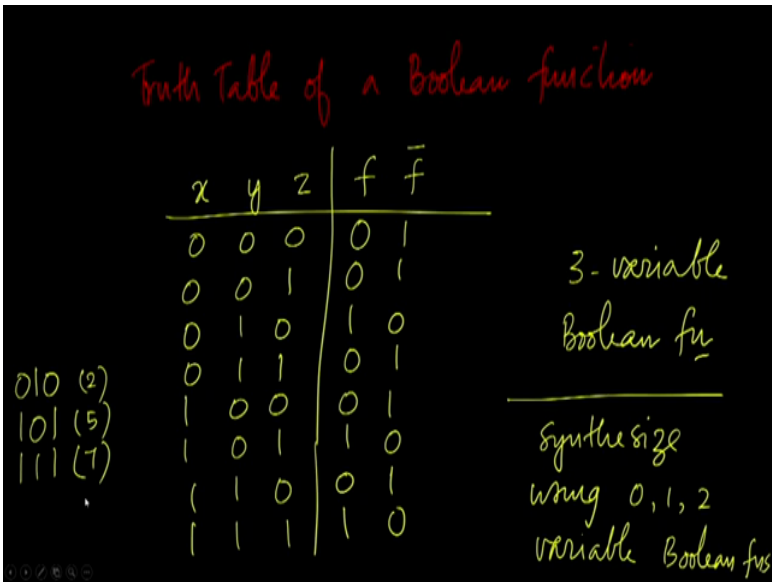


In fact, you can extend this to any number of variables for example, if you have three variables, let us say we want to compute the complement of x or y or z but since OR is an associative operator you can write this in this manner, then using De Morgan's law, we can write this as, the conjunction of x bar and y plus at the whole bar but then we know that y plus at the whole bar is y bar z bar by applying De Morgan's law for two variables, but since conjunction is associative, we can write this as x bar y bar z bar.

In other words, the complement of x or y or z is the conjunction of x bar y bar and z bar. Similarly, as a dual, we can also find that the complement of x and y and z is this by associativity of conjunction but that is x bar or y z bar but y z bar is y bar or z bar which by associativity can be written in this fashion, therefore De Morgan's laws can be extended from two variables to three variables and thus it can also be extended to four variables and so on.

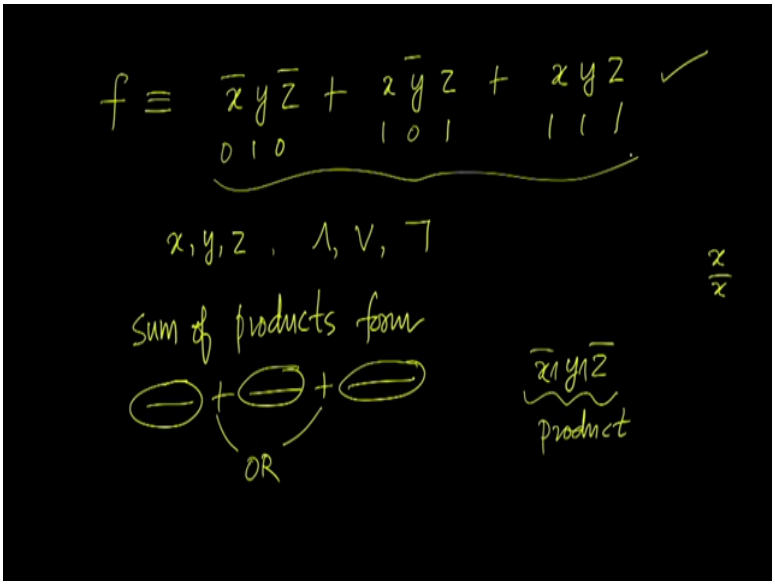
(Refer Slide Time: 30:06)

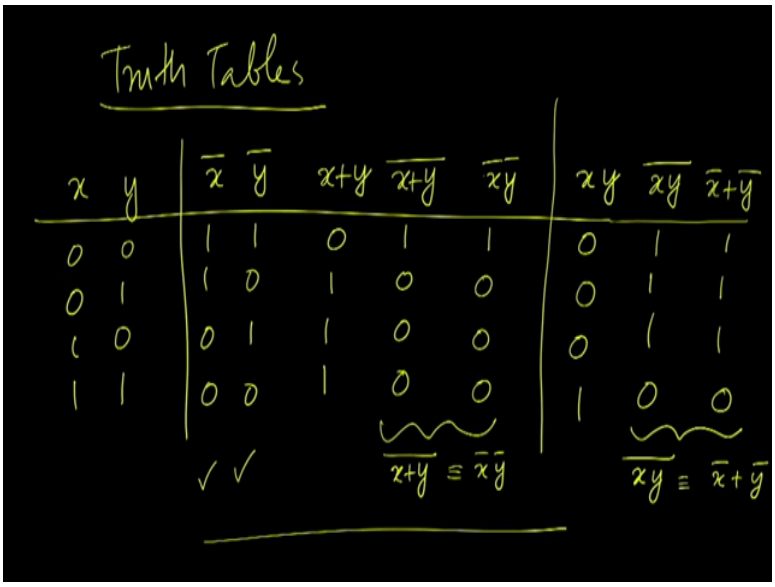




Now, let us go back to our Boolean function f and let us say we want to express f in terms of the inputs x and y and 0, 1, 2-variable logical connectives, logical connectives are the same thing as Boolean functions. So, let us see how f can be expressed in this form, for that let us take another look at the Boolean the truth table corresponding to f. So, here we find that f is 1 precisely when x, y, z are 0, 1, 0 or 1, 0, 1 or 1, 1, 1 that is 2, 5 and 7.

(Refer Slide Time: 31:16)





Which means, f is logically equivalent to x bar y z bar which corresponds to 2 0, 1, 0 or x y bar z which corresponds to 5 1, 0, 1 or x y z which corresponds to 1, 1, 1 when one of these combinations happen then f becomes 1 that is what the truth table tells us. So, f can be expressed as a composite expression of this form, this uses the input variables x, y and z and logical connectives AND, OR and NOT and this form is called the sum of products form, that is because it is expressed as an OR of several terms or corresponds to the addition symbol that is why we call it a sum and then each of the terms here is a conjunction.

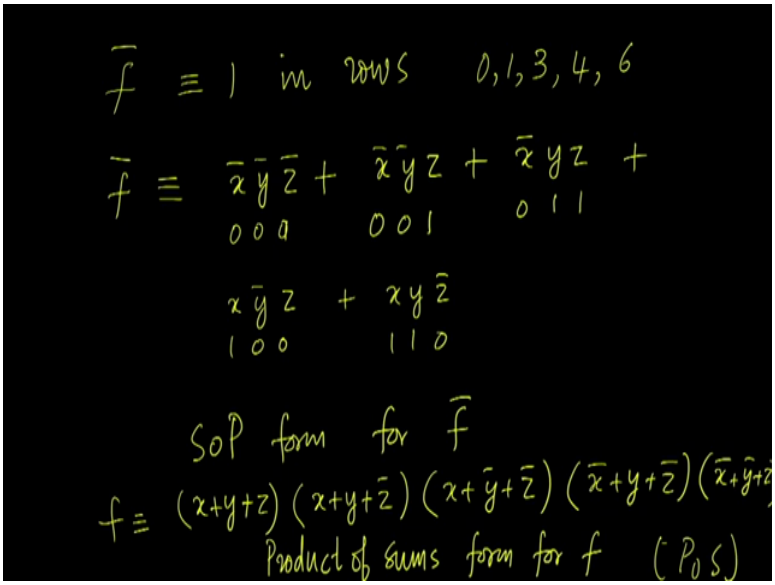
For example, x bar y z bar is a conjunction, it is an AND of three literals, a literal is either a variable or its complement. So, this is a conjunction of several literals therefore this is called the product form because in arithmetic expressions extra position corresponds to

multiplication when several entities are juxtapose we call it a product. Similarly, here also we call this a product, so therefore this is a sum of products form.

So, looking at the truth table, we can write the sum of products form of the Boolean function in this manner. So, this we can extend to any truth table given an n variable Boolean functions truth table. We can identify the rows in which the Boolean function becomes 1 then corresponding to these rows we can form conjunctions like this. So, in this case there are three rows corresponding to 0, 1, 0; 1, 0, 1 and 1, 1, 1 so they can be translated into product forms like this x bar y z bar; x y bar z; x y z, if either if any one of them is true then the function evaluates to true.

So, what we want to say is that the function is true if and only if at least one of these three terms is true. So, for any Boolean assignment at most one of them will be true. So, we want to say that f is true if and only if exactly one of them is true that can be expressed using an OR of all these terms that is why we call this a sum of products form. Therefore given the truth table of a Boolean function we can write the sum of products form of the function.

(Refer Slide Time: 34:41)



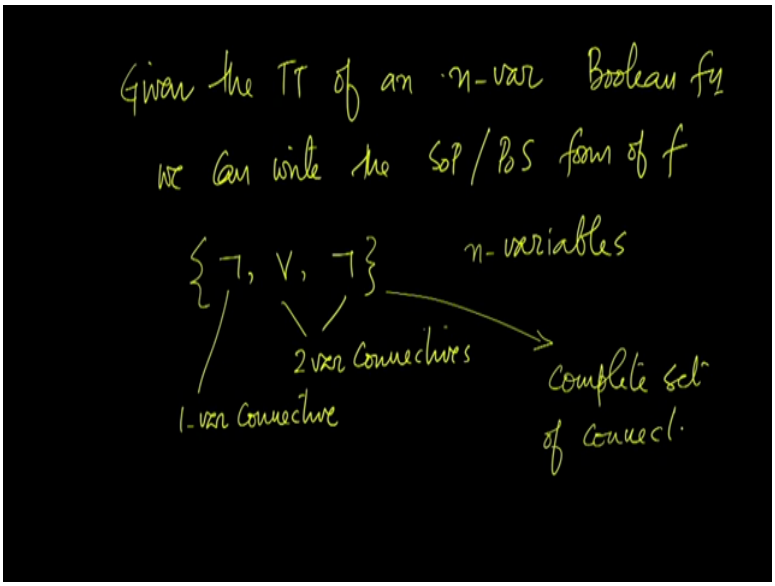
Now, let us consider the complement of this function. This complement of the function is 1 in rows which are not 2, 5 or 7 which means 0, 1, 3, 4, 6, these five rows will make the complement of f 1. Therefore, I can write f bar as x bar y bar z bar which corresponds to 0, or x bar y bar z which corresponds to 1, or x bar y z which corresponds to 3, or x y bar z which corresponds to 4, or x y z bar which corresponds to 6.

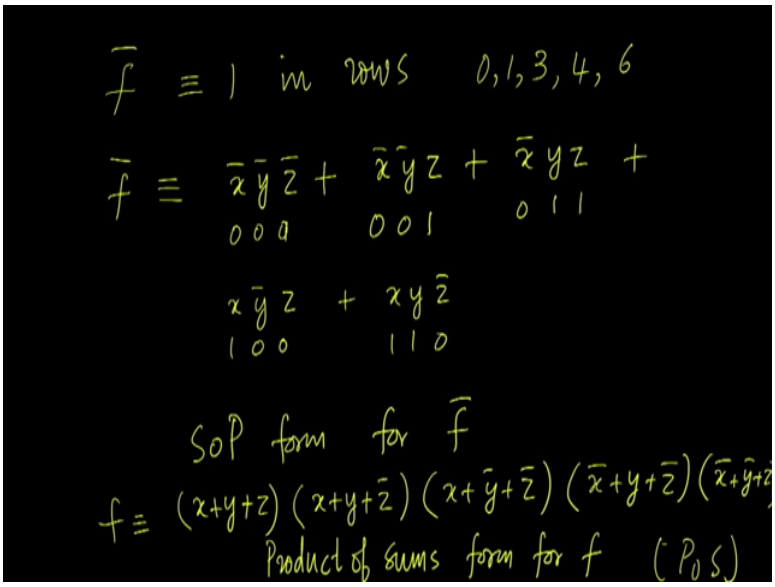
So, this is the sum of products form for f bar. So, once you have the sum of products form for f bar we can construct an expression for f. So, an expression for f can be obtained using De Morgan's law from this sum of products form. Now, we want an expression for f which is the complement of f bar, f bar is the complement of f and the complement of that is f. So, we want to complement the left hand side so we should complement the right hand side also.

So, on the right hand side we have a disjunction, so when you compliment a disjunction what you need to do is to complement each of its terms and then take the conjunction of these complements. So, we should take the complement of x bar y bar z, which is by De Morgan's law x plus y plus z plus stands for OR that has to be ended with the other complements, the complement of x bar y bar z would be x plus y plus z bar and the complement of x bar y z would be x plus y bar plus z bar and the complement of x y bar z would be x bar plus y plus z bar and the complement of x y z bar would be x bar plus y bar plus z, this is the product of sums form for f.

So, what this shows is that we can find the product of sums form for f when we are given the truth table for f from the truth table for f we construct a truth table for f bar, f bar is true in certain rows, you identify all the rows in which f bar is true, these are precisely the rows in which f is false then from these rows we construct the product sum of products form for f bar then if you apply De Morgan's law on this sum of products form for f bar we get the product of sums form for f.

(Refer Slide Time: 38:13)

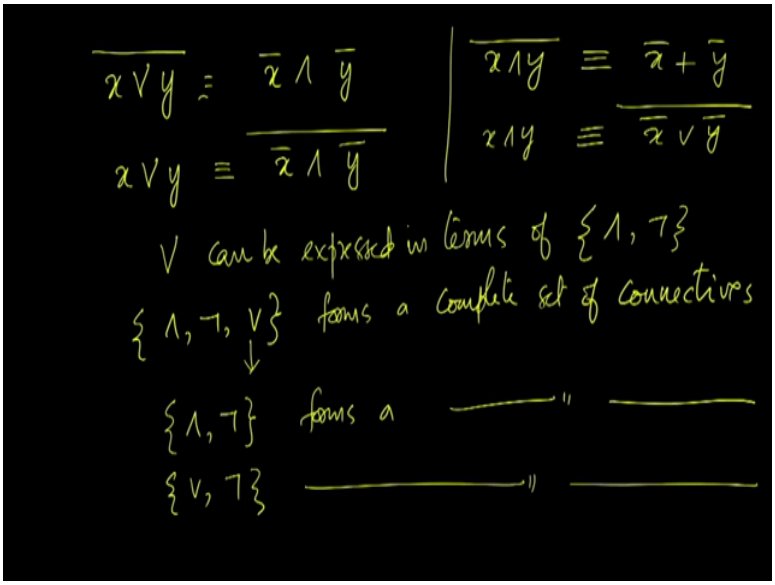




So, what that establishes is this, given the truth table of an n variable Boolean function, we can write the sum of products or product of sum form of f. So, we have an algorithm for doing this and this sum of product and product of sums form use only these Boolean connectives negation or an AND, in addition to the variables of the expression. So, the sum of products form of the product of sums form will use these n variables along with these connectives here AND and OR are 2-variable Boolean connectives whereas negation is a 1-variable Boolean connective.

Since every Boolean expression, irrespective of the number of variables in it can be expressed using these three Boolean connectives we say that this set is a complete set of connectives, that is these three connectives are enough to express any Boolean expression but is this the only complete set of connectives? By no means.

(Refer Slide Time: 40:02)

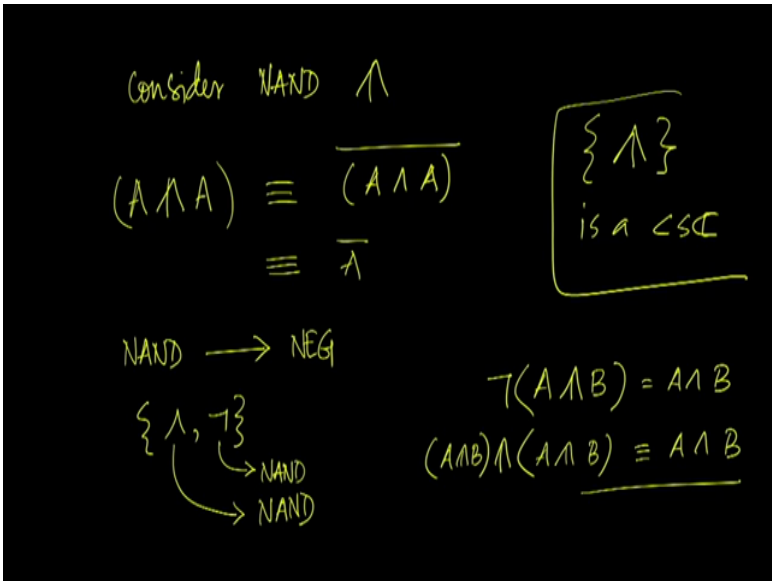


That is because, again from De Morgan's law we know that the complement of x or y is the complement of x and the complement of y therefore x or y is equivalent to the complement of x and the complement of y, the whole complement. Now, what this establishes is that OR can be expressed in terms of AND, and negation. Now, we know that AND negation and OR together form a complete set of connective, and here we find that OR can be synthesized in using AND and negation, so OR is not indispensable here.

Therefore, we find that AND and negation together forms a complete set of connectives. So, this is a complete set of connectives as well, and then we have the dual of this x and y the whole complement this equivalent to x complement or y complement, which means x and y is equivalent to x complement or y complement, the whole component. In other words, AND can be expressed in terms of OR and negation.

Therefore OR and negation also form a complete set of connectors. So, these are also complete sets of connectives, is there a smaller set of connectives that is complete? There is indeed.

(Refer Slide Time: 42:08)

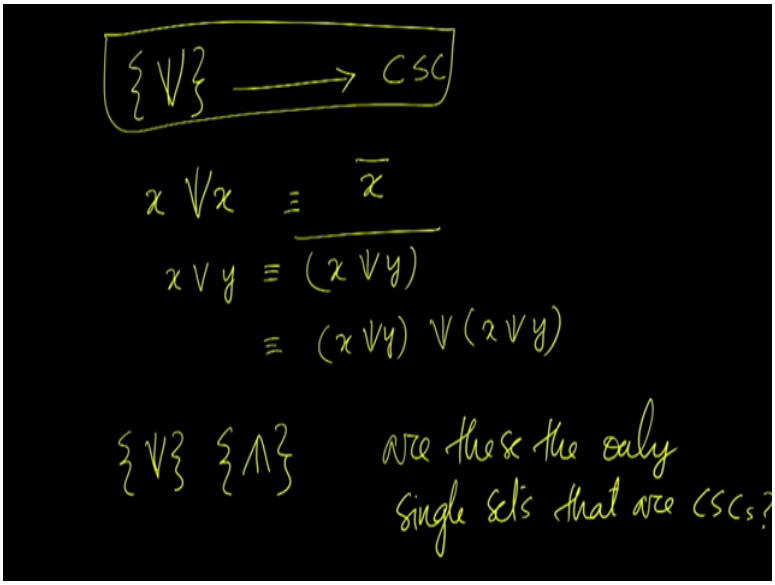


For example, consider the NAND function, denoted like this as an up arrow, what is A NAND A? A NAND A is equivalent to A AND A the whole complement but then what is A AND A when A is 0 it is 0 AND 0 which is 0 and when A is 1 it is 1 AND 1 which is 1, which means A AND A is nothing but A, so this is a complement which means when a variable is NANDed with itself, we get the complement of it or in other words we can use NAND to generate negation.

Now, look at this set AND an negation in this negation can be synthesized using NAND and then what about AND? AND is nothing but the negation of NAND, so if you take the negation of NAND of A and B and then negate it, what we get is, the AND of A AND B, how do we get this negation? This negation also can be synthesized using NAND. So, if you take the NAND of A, (A AND) A NAND B with it itself, what we get is, A AND B.

So, we find that both AND and not can be synthesized using NAND connectives. In other words, NAND alone is a complete set of connectives. Then, of course you would expect NOR also to behave the same way and it indeed does.

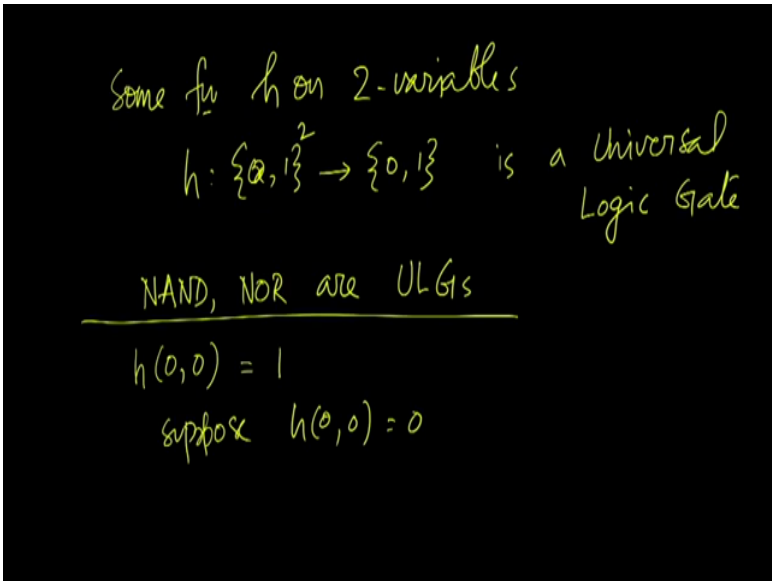
(Refer Slide Time: 44:15)



If you take x NOR x, you find that this is nothing but x complement when x is 0 we have 0 OR 0 which is 0 and the negation of that is 1 and when x is 1 we have 1 OR 1 with the negation of which is 0, therefore, x NOR x is the negation of x. So, negation can be synthesized to using NOR. Similarly, x or y is nothing but x NOR y complement and we have just seen that complement can be synthesized using NOR, therefore this also can be expressed in terms of NOR x NOR y NOR x NOR y this what x or y is.

Therefore, this is also a complete set of connectives. So, we find that there are these two single term sets that are complete sets of connectives, the NAND connective as well as the NOR connective but are these the only single term sets of, single term sets that are complete sets of connective, it transpires that they are the only ones, but how do we show this?

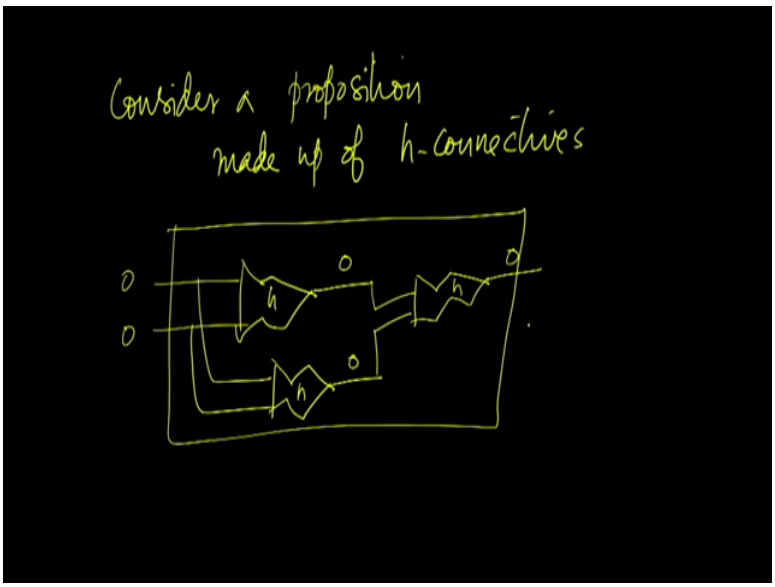
(Refer Slide Time: 45:58)



Suppose some function h on 2-variables is a universal logic gate. A universal logic gate is precisely a 2-variable Boolean function that forms a complete set of connectives by itself. So, we know that NAND and NOR are universal logic gates, but are these the only universal logic gates? We want to claim that they are the only universal logic gates. Now, suppose h is some 2-variable Boolean function that is a universal logic gate, then I claim that h of 0, 0 will have to be 1, why is this? Suppose h of 0, 0 is 0 that is when both the inputs to h are 0 then the output is 0.

Then consider a logical expression, a Boolean expression that is synthesized using h.

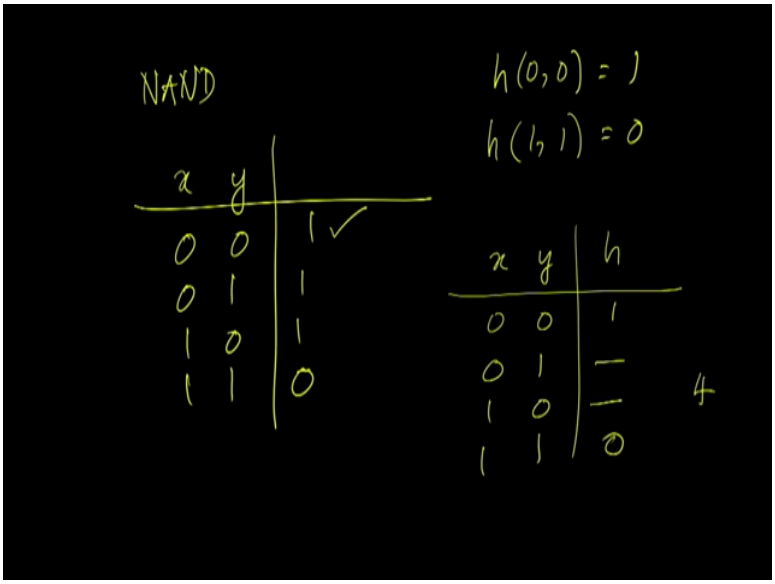
(Refer Slide Time: 47:38)



Consider a composite proposition made up of h connectives. So, in this the only connective that has been used is h and here all the inputs are, let us say both the inputs are let us say 0, in which case we have these inputs both 0 on which we apply this h connective and it produces a 0, let us say, and then on the same inputs we might have other h connective supply which will also produce 0 and then we might combine these 0s to reduce further signals using more h connectives, they will also keep producing 0.

In other words, if you have a circuit made up of h gates in this manner then the every signal which is inside this will be a 0 if both the inputs are 0. In other words, if both the inputs are 0 we will not be able to produce a 1 at the output of this, but every Boolean function is not of this form.

(Refer Slide Time: 49:00)

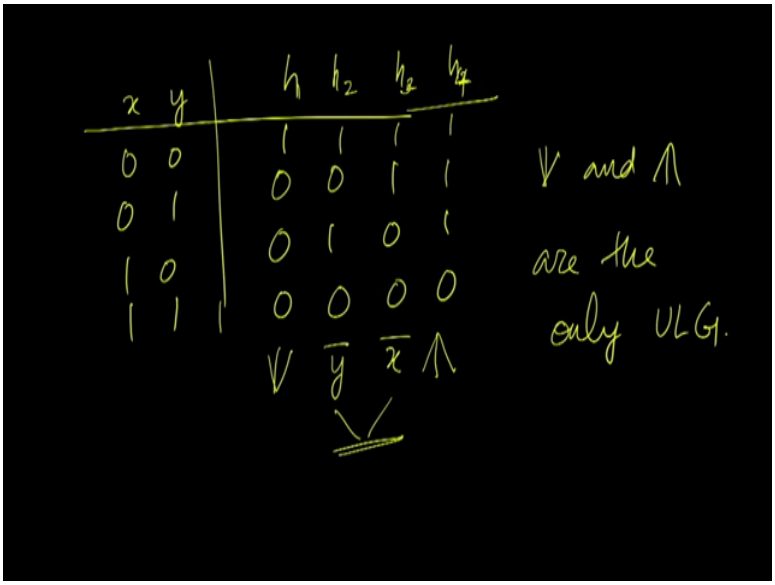


In particular, the NAND function, the truth table of NAND is like this, in particular this entry is 1 that is when both the inputs are 0 the output is 1. So, how would you synthesize NAND using h gates alone? That is not possible because if both the inputs are 0 h gates will keep producing only 0s, it will never produce a 1. So, NAND cannot be synthesized using the h case.

Therefore, we know that h of 0, 0 has to be 1. Similarly, we can also argue that h of 1, 1 has to be 0, that is when both the inputs are 1 the output has to be 0 otherwise we will not be able to produce for example, the OR function which produces a 1 when both the inputs are 1. Therefore, if you visualize the truth table of h, you find that it should be of this form, the first entry is 1 and the last entry is 0.

Now, there are two possibilities, the two vacancies these vacancies can be filled in 4 different ways. So, let us consider all those 4 possibilities.

(Refer Slide Time: 50:37)



So, here we have 1 and here we have 0 then let us fill these vacancies using 0, the other alternatives are 0, 1 here or 1, 0 here or 1, 1 here. So, these are the 4 possible functions that could be universal logic gates. Now, what is 1, 0, 1, 0? 1, 0, 1, 0 is the complement of y and what is 1, 1, 0, 0? It is a complement of x. These two cannot be universal because they depend only on 1-variable.

So, using these two gates that is the negation of y and the negation of x, we will not be able to synthesize any Boolean function which depends on both the inputs. So, these two are anywhere ruled out, what are the remaining? This is nothing but the NOR function and this is nothing but the NAND function. So, we find that NOR and NAND are the only universal logic gates. That is it from this lecture, hope to see you in the next, thank you.