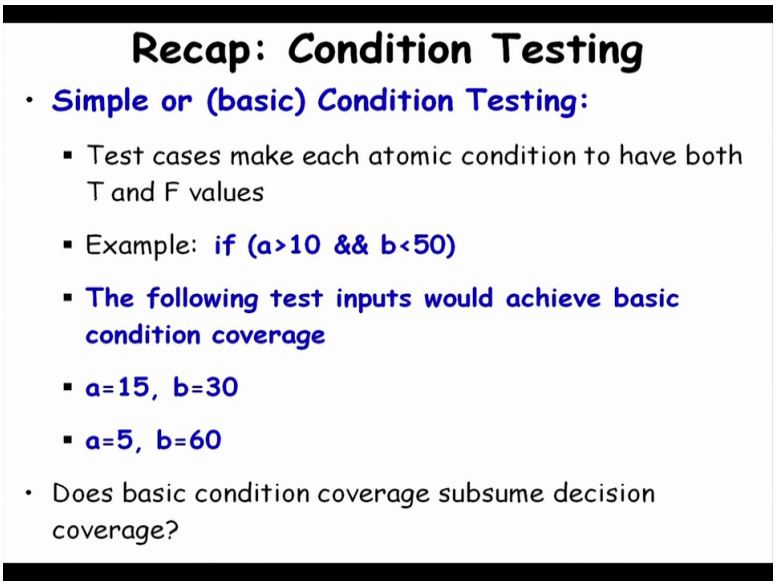
Software Testing
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 12
MC/DC Testing (Contd.)

Welcome to this session. So far, we have seen some white box testing strategies. We had seen the statement coverage, branch coverage and condition coverage. We had seen that the basic condition coverage is the one, where each of the component condition should assume true and false values. And then, we had seen the multiple condition coverage; where all possible combination sub truth values should be assumed by the basic conditions. And then, we had also looked at condition decision coverage. One of the main short coming of the condition based testing is that the multiple condition testing generates too many test cases, and therefore becomes impractical.

On the other hand, the basic condition coverage is very weak testing. Now, let us see can we achieve the thoroughness of testing of multiple condition coverage without having an exponential number of test cases. So, that is about MCDC testing. Let us look at what is involved in MCDC testing.

(Refer Slide Time: 01:44)



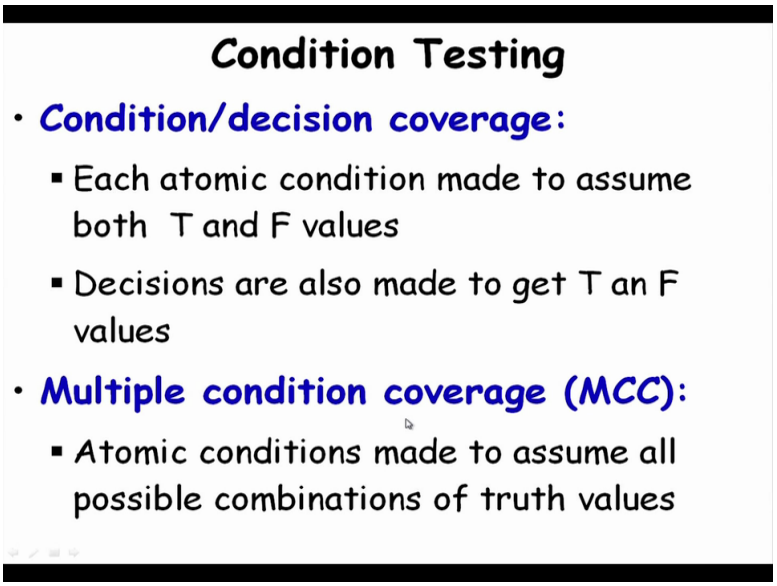
In the simple or basic condition testing, we had said that each of the basic condition or the component conditions must take true and false values. For example, in this case, in this statement there are two basic conditions, and each of this need to take true and false values. So, the following two input, test inputs will achieve basic condition coverage because the first one, a = 15, we will set it true and b = 30, we will also set it true, and a = 5, we will set it false and b = 60, we will also set it false.

But, the question that I want to ask is that does basic condition coverage subsume decision coverage? Is basic condition coverage, a stronger testing than decision coverage? To answer this, you do not have to think very hard. Actually, I can always choose both of these to be equal; this is true and false and false and true. And therefore, the basic condition coverage is achieved, whereas the decision coverage is not achieved.

Let me repeat. I can set the first condition true, second one false. In the second test case, I can have this as false and this as true. And therefore, both this component conditions have taken true and false values, but then each time the decision evaluates to false. So, decision coverage is not achieved. To answer this question that the basic condition coverage does not subsume decision coverage. In other words, the basic condition coverage even though a very simple testing, very intuitively simple testing technique, but is a very weak testing. And, even weaker than the statement coverage.

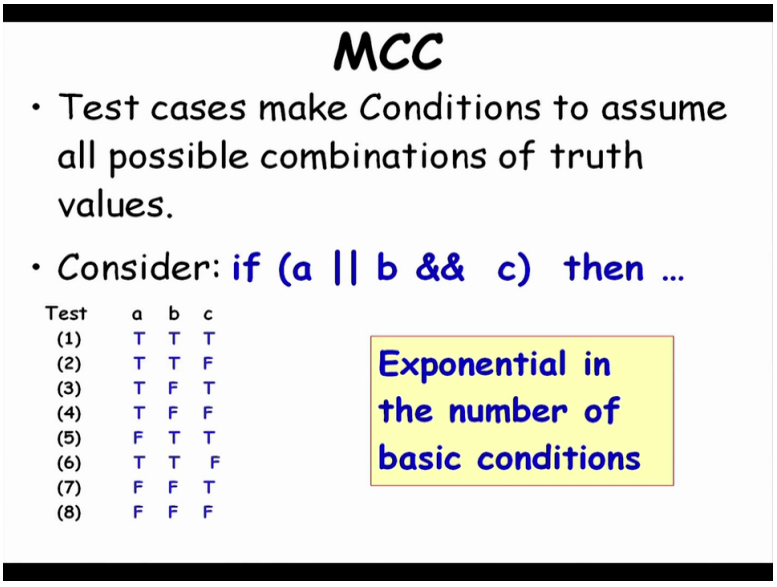
.

(Refer Slide Time: 04:14)



But, then we can insist that not only we need basic condition coverage, we also want decision coverage. So, that is known as the condition or decision coverage. So, here each atomic condition should assume true and false values. And in addition, we need to ensure that the decision itself gets true and false values. And then, we had also seen the multiple condition coverage, where the different atomic conditions should assume all possible combinations of truth values.

(Refer Slide Time: 04:51)

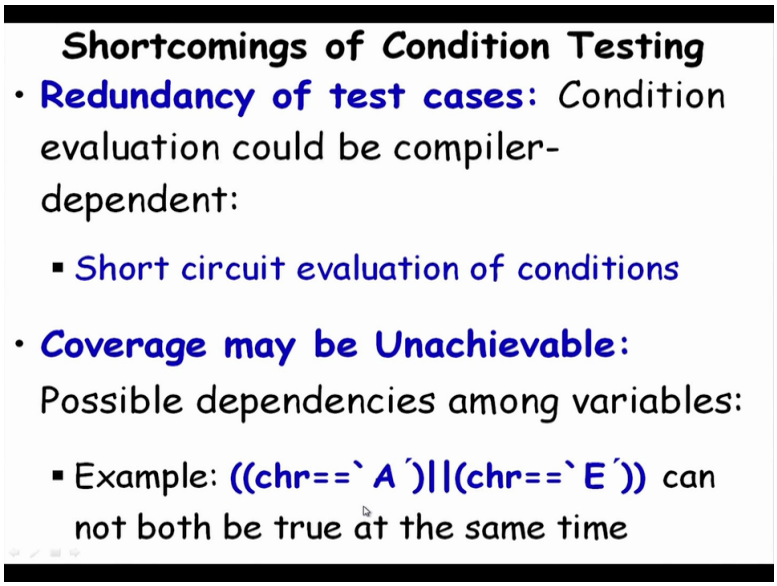


Just to elaborate what we mean by multiple condition coverage, if we have an example such as a or b and c is the compound condition, then there are three atomic conditions here; a, b, c. So, we need eight test cases for testing all possible combinations of truth values of the basic conditions; a b c set to all true; true, true, false and so on.

But, then as the number of component conditions increases, the number of test cases required to achieve the multiple condition coverage becomes exponentially large. And therefore, if we have ten basic conditions, we need thousand test cases to achieve multiple condition coverage. And, if we have 20 component conditions, then we need a million test cases. And if we have 30 billion, but then you might ask that do they really occur that many combinations, component conditions in conditional expressions, do they occur? Yes, in many applications they do occur. For example, control applications.

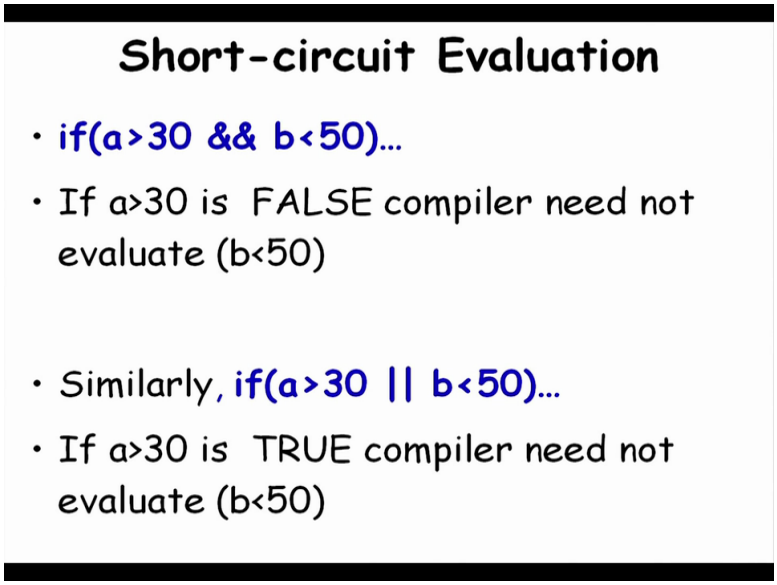
The control action may depend on several conditions. And 20, 30 component conditions is very common in controllers. And, there are also many other applications, where we can have a compound condition involving twenty or thirty atomic conditions. Therefore, the multiple condition coverage becomes impractical. We need to somehow achieve the thoroughness of multiple condition coverage testing without really blowing up the number of test cases.

(Refer Slide Time: 07:06)



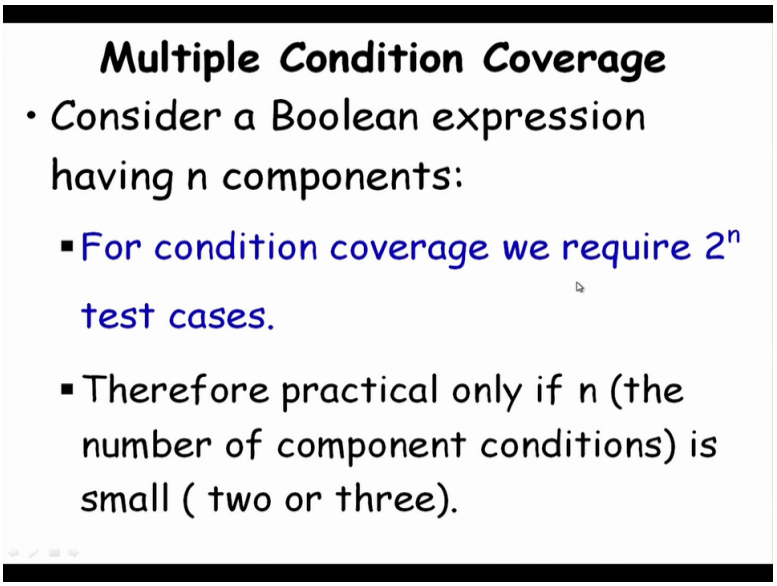
So, this condition testing; some of the shortcomings are redundancy of test cases. So, the test cases are redundant because due to the short circuit evaluation of the compiler, many of the test cases, they actually map to the same values or same expression evaluation, same conditional expression evaluation results. So, some of the test cases become redundant. And also, some of the coverage may be unachievable. Some of the condition, combinations may be unachievable. For example, just this example that character is equal to A or character equal to E. We cannot have both true achieved at the same time. So, multiple condition coverage would not be possible to achieve for this case.

(Refer Slide Time: 08:11)



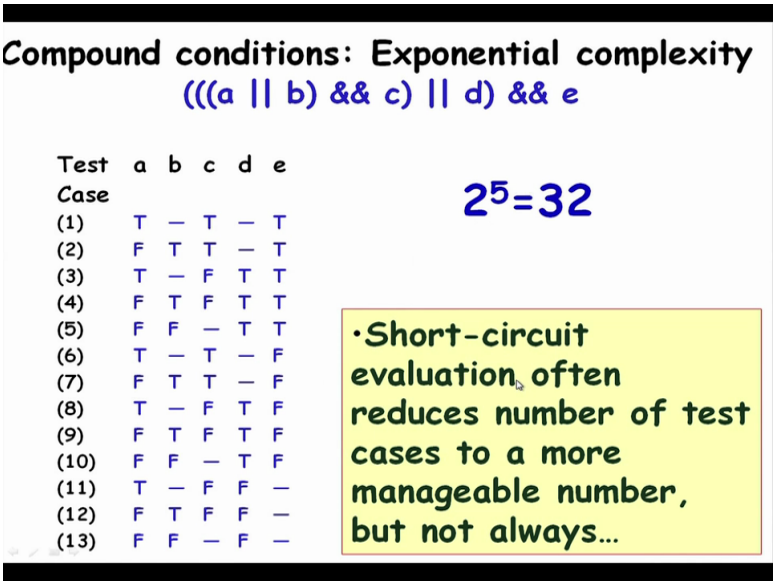
And, just to refresh what is short circuit evaluation, the compiler need not evaluate b < 50, once it knows that a > 30 is false. And, similarly if a > 30 is true, then the result of the decision is irrespective of what is there on the second condition. The second condition is not evaluated. The decision result are computed based on the first condition. So, these are the short circuit evaluation; some of the short circuit evaluation techniques adopted by compilers.

(Refer Slide Time: 08:49)



Now, let us see how the multiple condition coverage can help us achieve the thoroughness of; sorry, the MC, DC coverage can achieve, help to achieve the thoroughness of the multiple condition coverage without requiring an exponential number of test cases.

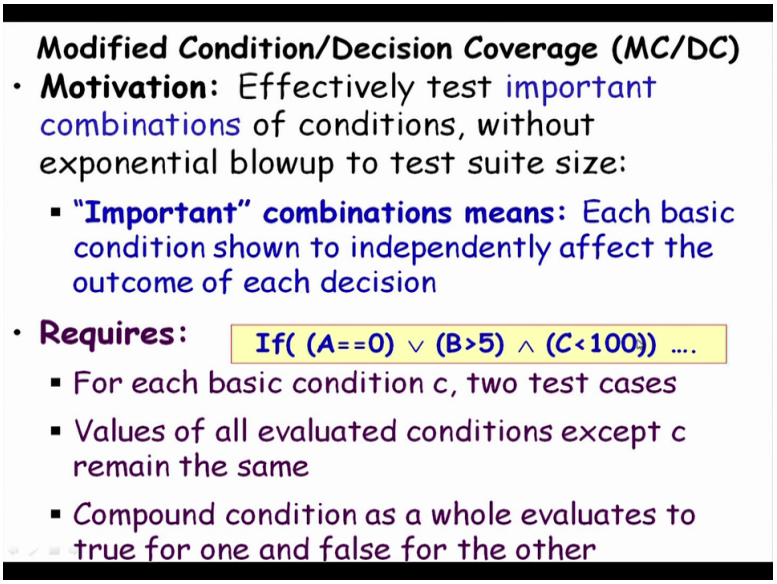
(Refer Slide Time: 09:11)



So, this ( ) example that shows that here we have 5 component conditions here, and therefore we need 32 test cases for multiple condition coverage. But, if we use the short circuit evaluation techniques adopted by compilers, actually the short circuit evaluation adopted by different

compilers differ. So, this is just one example; short circuit evaluation. In that case, we might just achieve condition coverage, multiple condition coverage with just 13. So, you need not use 32 because of the short circuit evaluation.

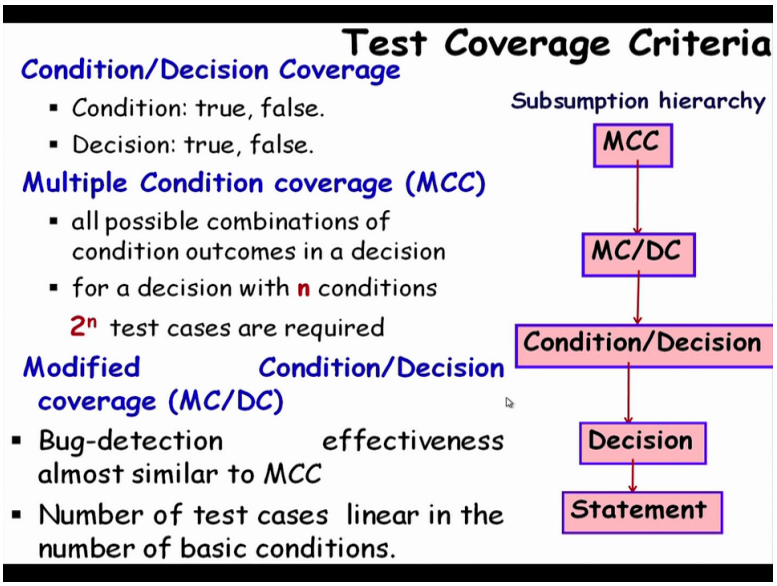
(Refer Slide Time: 09:59)



Now, let us see what is multiple condition decision coverage. So, here we test some important combinations of conditions. And therefore, the number of test cases is much less. Actually, linear in the number of basic conditions. We avoid the exponential blow of in the size of the test cases. But, at the same time we achieve as much, almost as much thorough testing as the multiple condition coverage testing. Here we have this compound condition has three atomic conditions A = 0 or B > 5, C < 100.

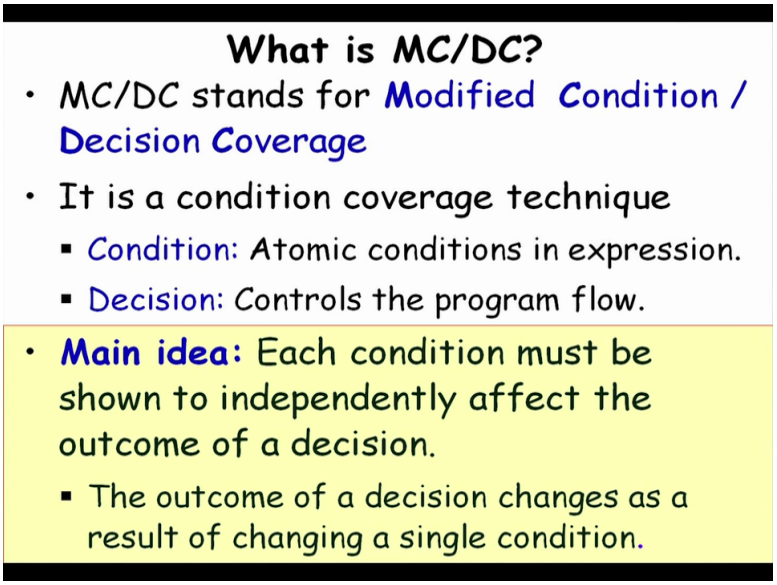
So, in the multiple condition decision coverage we let each of these basic conditions achieve true and false values. And when it achieves true and false values, the others are held constant truth values. And, these true and false value actually determine the branch outcome. That we have to ensure. So, let us see how it is done and how to design the test cases for this.

(Refer Slide Time: 11:17)



If we look at the hierarchy, the multiple condition decision coverage is stronger than both statement coverage, the decision or branch coverage, the condition decision coverage. But, it is weaker than the multiple condition coverage.

(Refer Slide Time: 11:44)

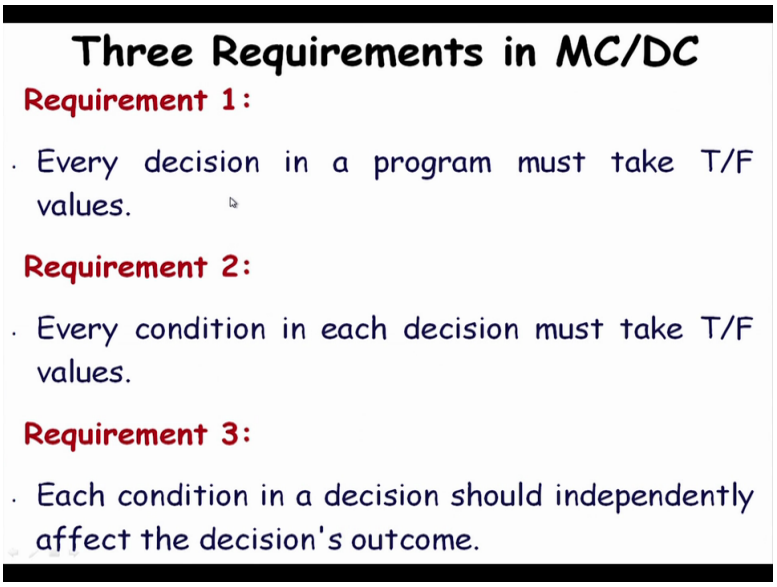


Now, let us see what is involved in MCDC testing, how we design the test cases for this, how many test cases and so on. So MCDC stands for modified condition decision coverage. And, here the terminology that we have been using. Let me repeat that again. Condition, we mean the atomic

conditions. And, the decision is the one that determines the program flow, the control flow it determines the outcome of the compound condition. So, this is a modified condition decision coverage.

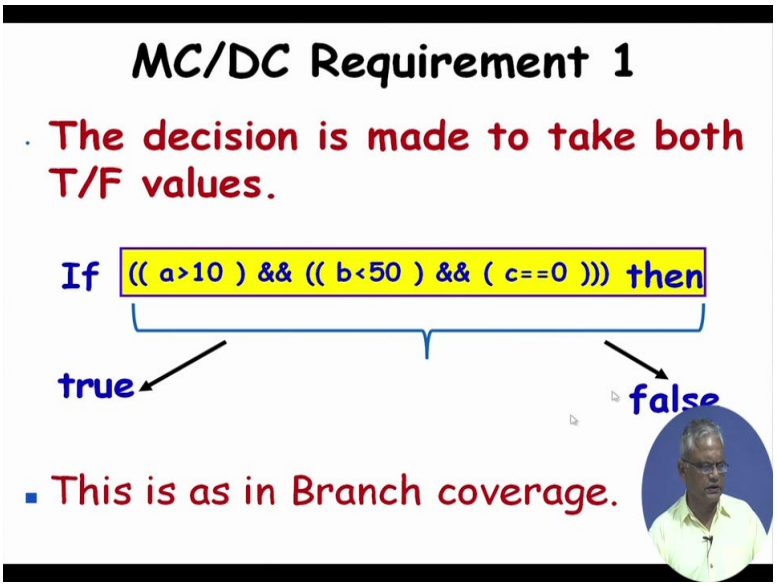
Here, the main idea is that every atomic condition will be set to true and false values. And, we have to ensure that when this is set to true and false values, the other basic conditions, the other atomic conditions are held at some constant value. And, this atomic condition will determine the result of the compound condition or the decision.

(Refer Slide Time: 13:03)



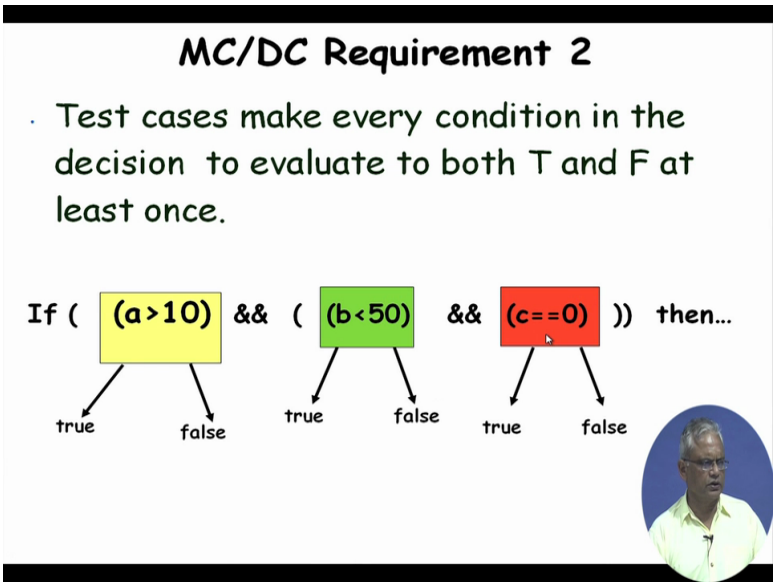
So, there are three requirements here. As I was saying that the first thing is that the decision in the program must take true and false value. So, the compound condition must take true and false value; every atomic condition in the decision must take true and false value. And, when an atomic condition takes true and false value, the other atomic condition truth values are to be held constant. And, this atomic condition as it takes true and false value. The decision outcome also toggles between true and false.

(Refer Slide Time: 13:44)



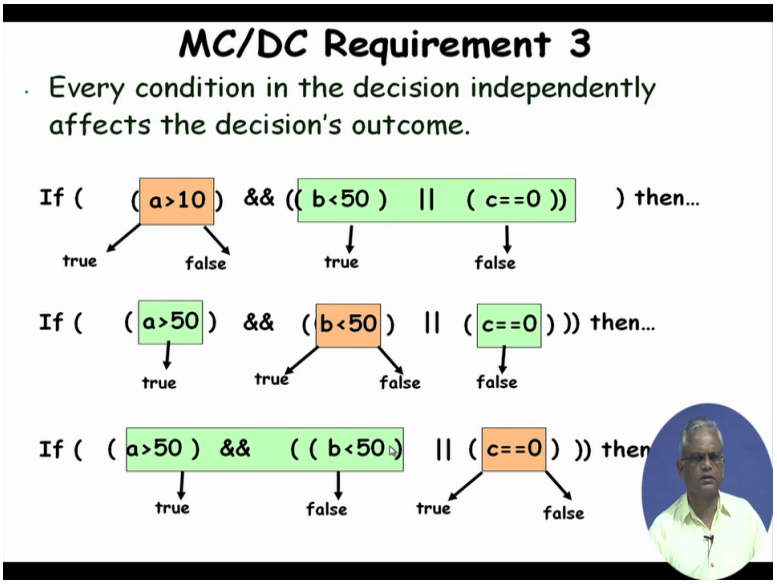
As we look at examples, it should become clear. Now, let us look at the first requirement. So, the first requirement is that the decision or the compound condition must take true and false value. This is the entire compound condition and the test cases should ensure that this branch condition evaluates to true and false value, which is of course the same as the branch coverage.

(Refer Slide Time: 14:17)



Now, the second requirement is that every atomic condition here should take true and false value. So, the first one must take true and false value; a > 10 and then a < 10. The second one also should take true and false value, the third one should also take true and false value.

(Refer Slide Time: 14:42)



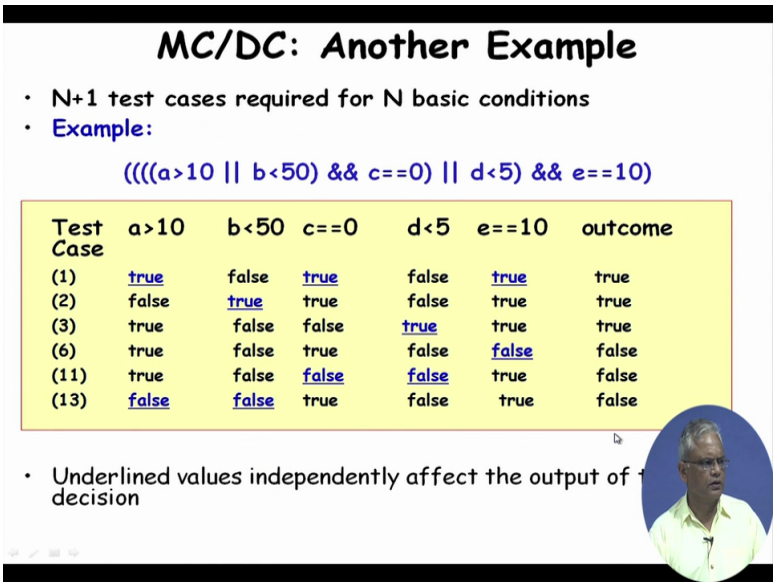
Now, the third requirement is that when we set the basic condition, the test case sets the basic conditions to true and false value. The other basic condition should be held at some constant value. And, the basic condition should affect the decision outcome.

Let us see the example. So, let us see the first basic condition. We set the basic condition to true. And then, the rest two are, let us say, held to true and false. And therefore, the atomic, the expression; the evaluation of this condition will become true. This is false, this is true. So, true or false is true. And when this is true, the outcome will be true. Now, set keeping these two same, true and false, will set a greater than 10 to false. And therefore, this will become true and this is false. So, false and true will be false. So, the outcome will become false, as this basic condition takes true and false value, so does the branch outcome keeping the other basic conditions at some constant truth value.

Similarly, we take up the second one. We set the test cases; set it true and false, while holding these; a true and false value. And, as you can evaluate the outcome of the branch toggles between true and false. Similarly, the third one. We set the third basic condition to true and false and keeping these

two at true and false, and therefore the branch outcome toggles.

(Refer Slide Time: 16:51)

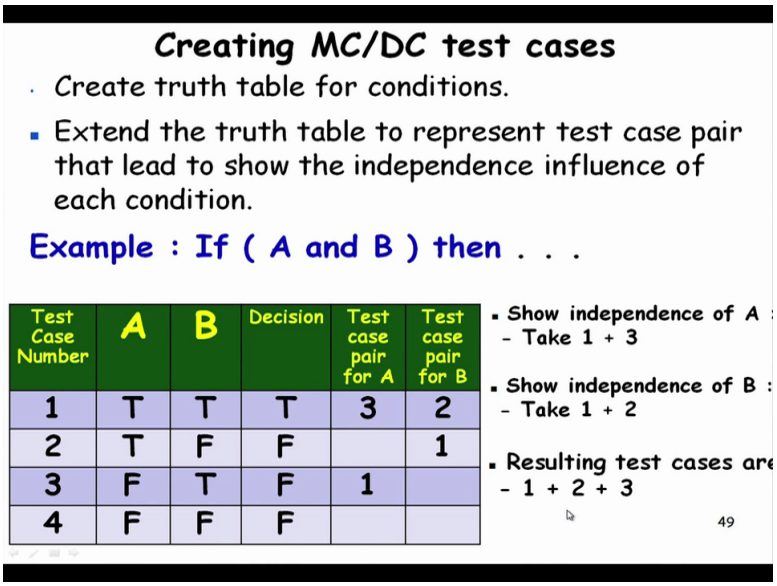


Now, let us look at another example just to understand this technique better. Let us look at another example, where we have 1, 2, 3, 4, 5; 5 test cases, sorry, 5 basic conditions. And then, we will see that we can achieve MCDC coverage with 6 test cases. And therefore, without giving any proof, we will speculate that we need n + 1 test cases, when we have n basic conditions in a compound conditional expression.

In other words the number of test cases required to achieve MCDC coverage is linear in the number of basic conditions. So, let us see here. So, we have these five conditions listed here. And then, we have the truth values listed here. And, when we; so, when a > 10 is true, and these are set at false true false true. And then, when a > 10 is false and the other ones are held as false true false true, just like the previous one. Then, the output toggles.

Similarly, for the second basic condition as it assumes true and false values, and the rest are held at some constant value, the output toggles. So these are the test cases which set to this truth values will achieve MCDC coverage. But, then you might ask that how do you know which test cases or how do we design test cases that will achieve MC, DC coverage.

(Refer Slide Time: 18:42)

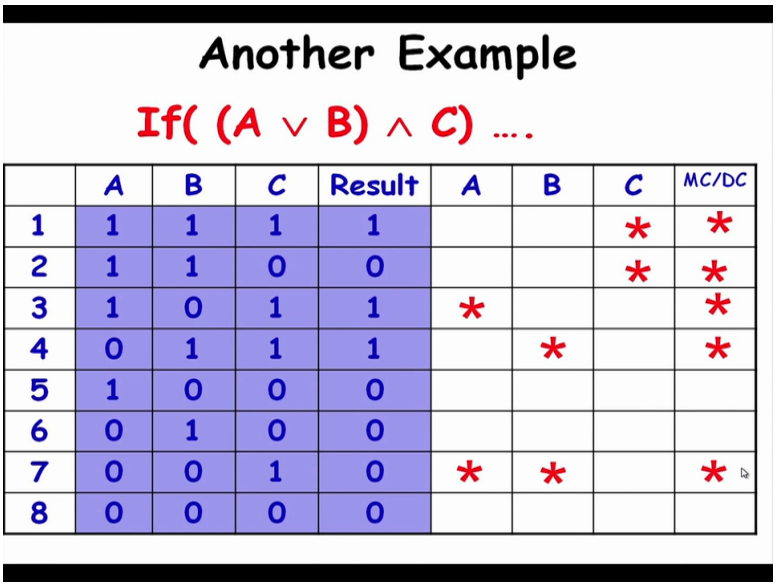


Let us see. First, let us look at a very simple example; how do we design MCDC test cases. Let us look at this examples A and B. So, we first develop the truth table and the decision outcome. So when A B true, decision outcome is true and so on. And then, we check that if we hold A at true and B at true, the output is true. Now, when we set A is false holding B at true, the outcome is false.

In other words the test case one along with test case three will achieve the independent evaluation of the first basic condition. Now, what about B? So, if the test case pair for A is 1 and 3. And, we will check that for two, we cannot achieve the independent evaluation for the first atomic condition A and with three, of course with 1. We will achieve the independent evaluation of A. Now, what about B? So, if you set B is equal to true and A equal to true, the outcome is true. Now, if we keep A as true and B as false, the outcome is false.

So, one along with two achieves the independent evaluation of the basic condition two; that is B. So, 1 3; 1 plus 3 and 1 plus 2, they achieve the independent evaluation of a and b. Or in other words, our MCDC test case will be 1 + 2 + 3.

(Refer Slide Time: 20:44)

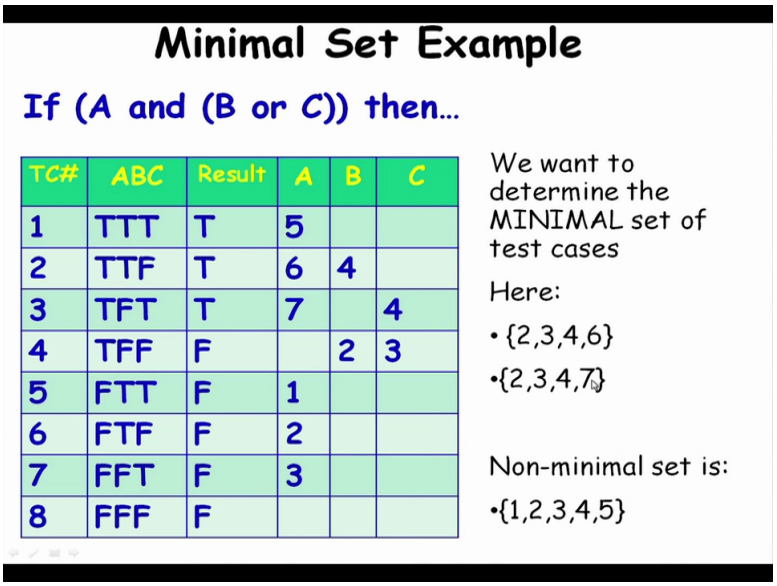


	A	B	C	Result	A	B	C	MC/DC
1	1	1	1	1			*	*
2	1	1	0	0			*	*
3	1	0	1	1	*			*
4	0	1	1	1		*		*
5	1	0	0	0				
6	0	1	0	0				
7	0	0	1	0	*	*		*
8	0	0	0	0				

Now, let us look at another example. So, here it is slightly more complex example involving three basic conditions in this conditional expression. Now, we develop the truth table for ABC and we write the result here. Now, we have to first check which two test cases achieve the independent evaluation of A. Can we achieve? With respect to one, here A is 1 and BC are 1 1 . Now BC are 1 1 and A is 0. But, then the result is 1 here. So, the result does not toggle. And therefore, one and four do not achieve independent evaluation of A. What about 1 0? Now, 1 0 is here. Here also the outcome does not toggle. But, look at here. The third one, A is 1 and BC are 0 1. And the seventh one, A is 0, BC is held at 0 1. And, 7 outcome toggles.

In other words 3 and seven will achieve independent evaluation of the basic condition A. Now, what about B? Similarly, we will look for the; when we toggle B keeping A and C constant, the outcome should toggle. So, we find this. Four along with seven; so in four, B set 1 and A and C are set at 0 1 and B is set at 0 and this is 0 1. And, outcome toggles. And, similarly C is 1 2. And therefore, our MCDC test case will be 1, 2, 3, 4 and 7. I hope you would have given a compound condition; will be able to design the MCDC test cases for this.

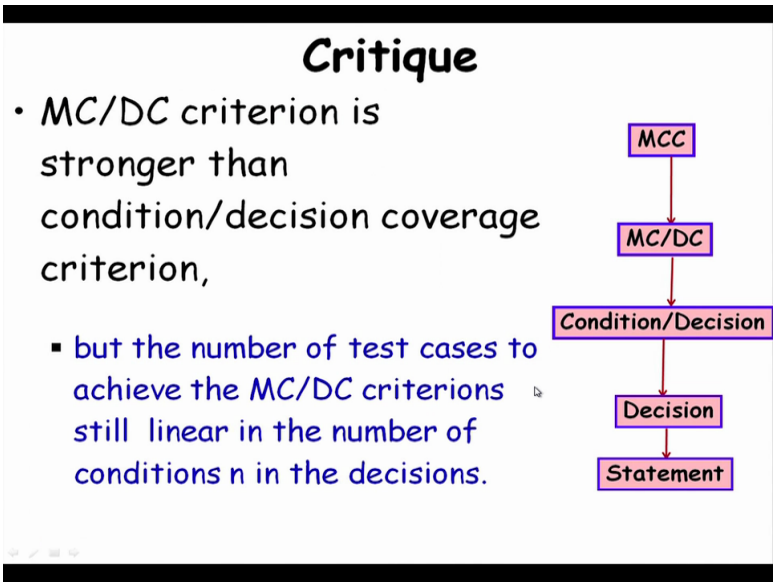
(Refer Slide Time: 22:55)



But then, there can be several sets of test cases which can achieve MCDC coverage. And, we need to check the minimal number of test cases. And, for this we do an evaluation. Just like the previous examples and find out which pairs achieve independent evaluation. We will find that multiple pairs achieve independent evaluation. For example, 1 and 5 achieve independent evaluation of the first basic condition. 2 and 6, 3 and 7, 5 and 1, 6 and 2, 7 and 3; this achieve independent evaluation of A.

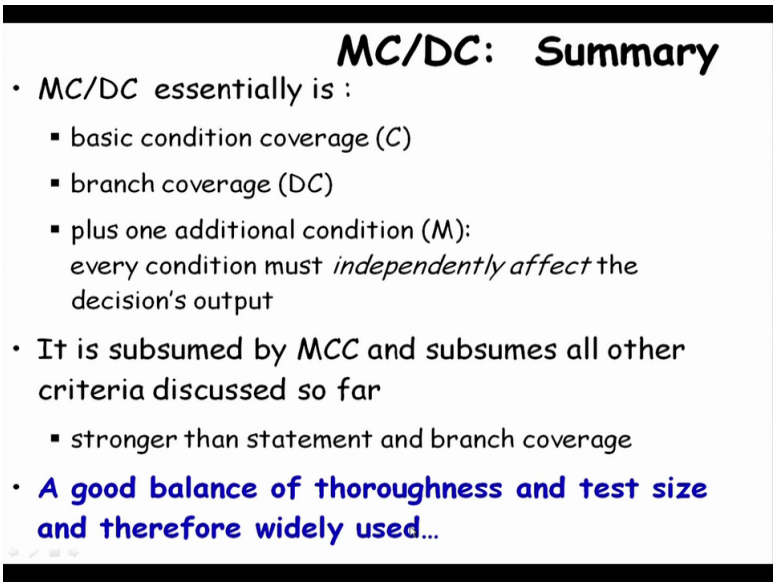
Similarly, 2 and 4 and 4 and 2 achieve independent evaluation of B. That is the second condition. And, the third one is only 3 and 4 or 4 and 3. Now, if we find out which are the test cases that achieve the MCDC coverage, it is 2, 3, 4, 6 or 2, 3, 4, 7 or 1, 2, 3, 4, 5. And therefore, if we want to choose the minimal set, we can choose either of these two.

(Refer Slide Time: 24:13)



So, we just saw that MCDC testing is a very effective testing technique with a small number of test cases. The number of test cases is linear in the basic conditions.

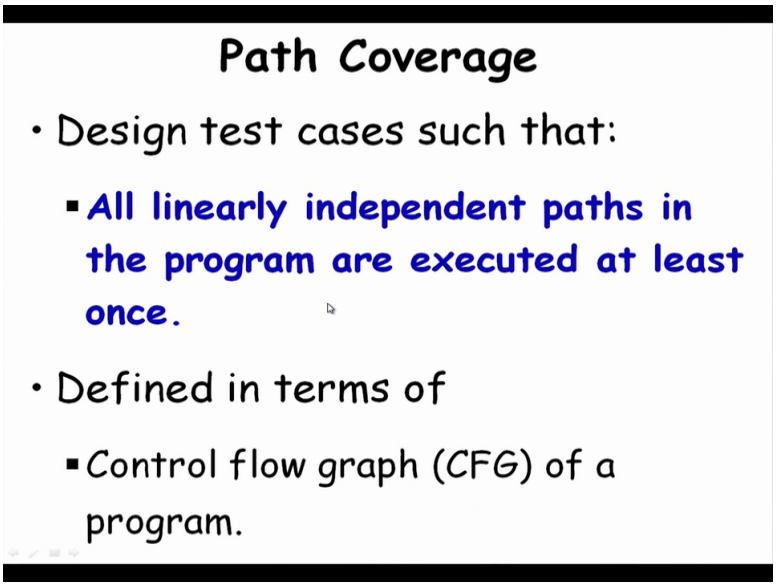
(Refer Slide Time: 24:35)



To summarize MCDC, we said that there are three things that are required. Each basic condition should achieve true and false values, the branch coverage should be achieved and the third condition is that every condition must independently affect the decision's output. And, as we have already been saying that it provides a good balance of the thoroughness of testing and the test size.

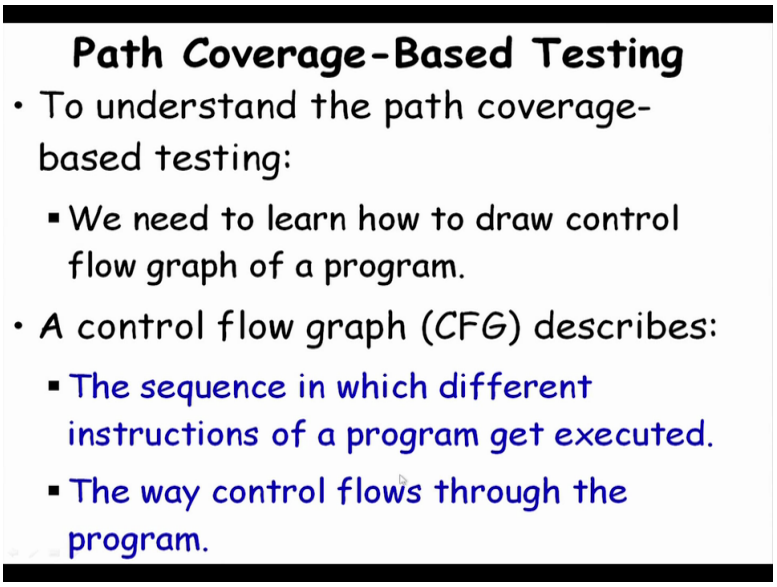
And therefore, it is widely used and is mandated by many testing standards.

(Refer Slide Time: 25:20)



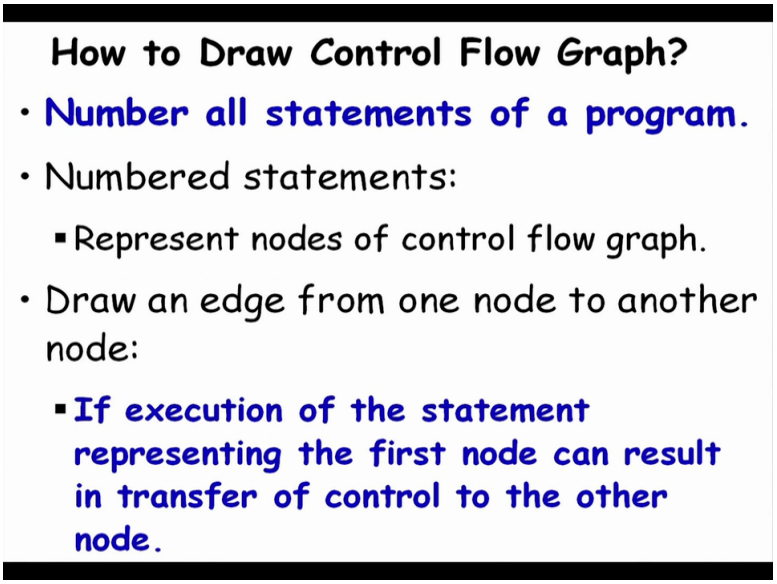
Now, let us look at path testing. In path testing, we want all the paths in a program to be covered. But then, we will see that there are too many paths in a program. And therefore, our requirement will be to cover the linearly independent paths in a program. The paths in a program are defined with respect to the control flow graph of a program.

(Refer Slide Time: 25:47)



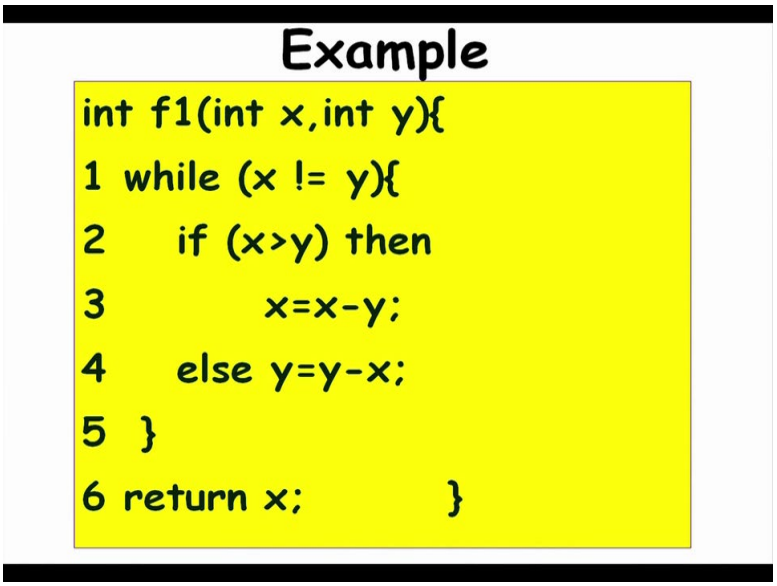
Here, as we know the control flow graph, it represents the sequence in which the different instructions of a program get executed. Or in other words, the way control flows through the program. So this, we represent in the form of a graph.

(Refer Slide Time: 26:13)



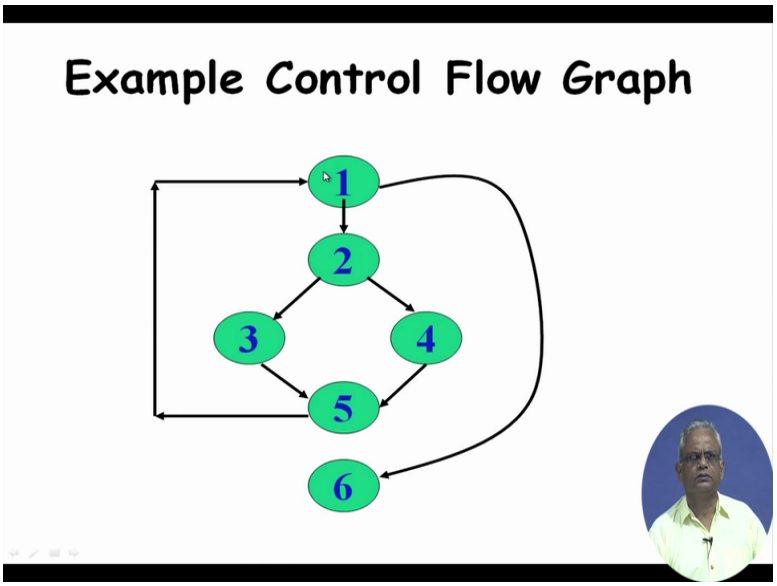
But, then how do you draw the control flow graph? Drawing control flow graph is very easy, if we understand it well. The first thing is we number all the statements of the program. And each of those numbers, we make nodes out of that. We draw nodes for each of those numbered edges. And then, we draw an edge from one node to other, if execution of a statement can transfer control to the other node. If execution of a statement can transfer control to the other node.

(Refer Slide Time: 26:55)



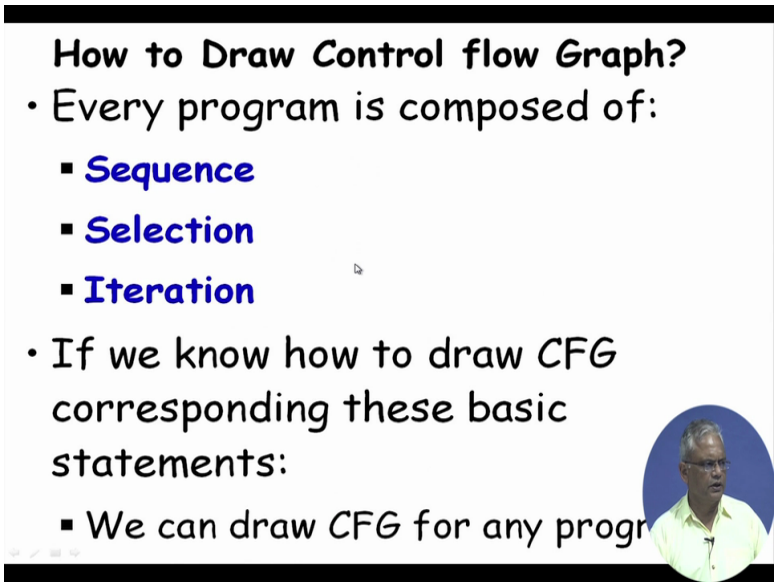
So, let us look at this program. We number the statements here.

(Refer Slide Time: 27:03)



Then, we draw the nodes here with the numbered statements. And then, we draw edges depending on whether control can transfer from a node to another node.

(Refer Slide Time: 27:19)



But, then we must have some systematic way of drawing this. And, that is based on realizing that every program consists of three types of statements; the sequence, the selection and the iteration type of statements. So, if we know how to draw the control flow corresponding to these three statements, we can very easily draw the control flow for any program; because they are after all composed of these three basic types of statements

And therefore, we need to just examine how we draw the control flow edges for these three types of statements. And then, we given even a very large program, we can develop its control flow graph. So, that we will look in the next session. Right now we will conclude with this.

Thank you.