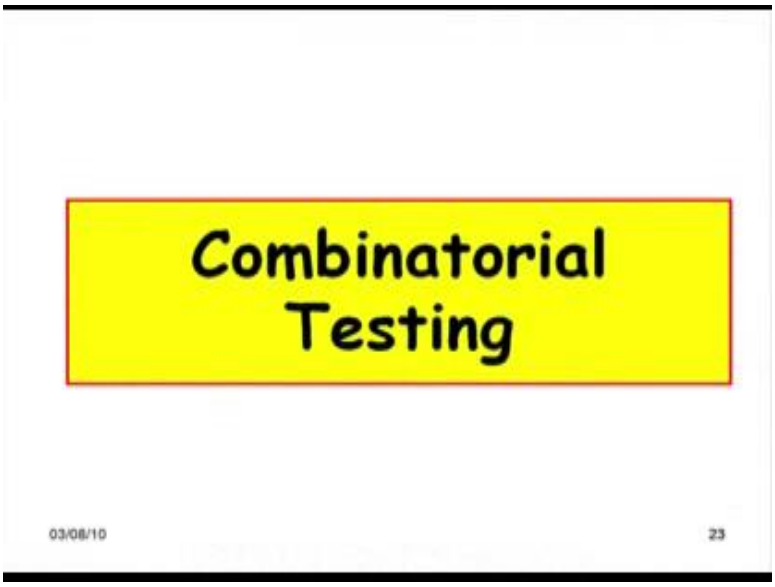
Software Testing
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 08
Combinatorial Testing

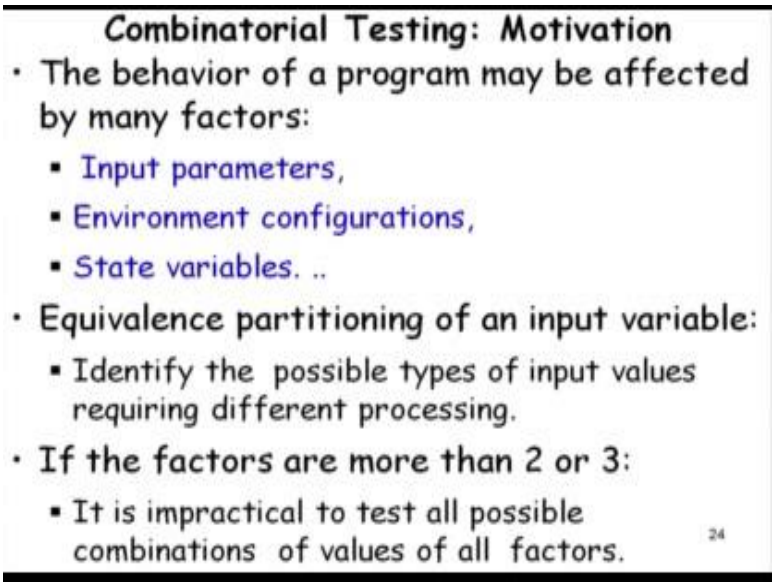
Welcome to this session. We were discussing in the previous sessions about Black-box testing techniques.

(Refer Slide Time: 00:43)



And we looked at two important black-box testing techniques which are equivalence class testing and special value testing. Now let us look at Combinatorial Testing, which is another black-box testing technique.

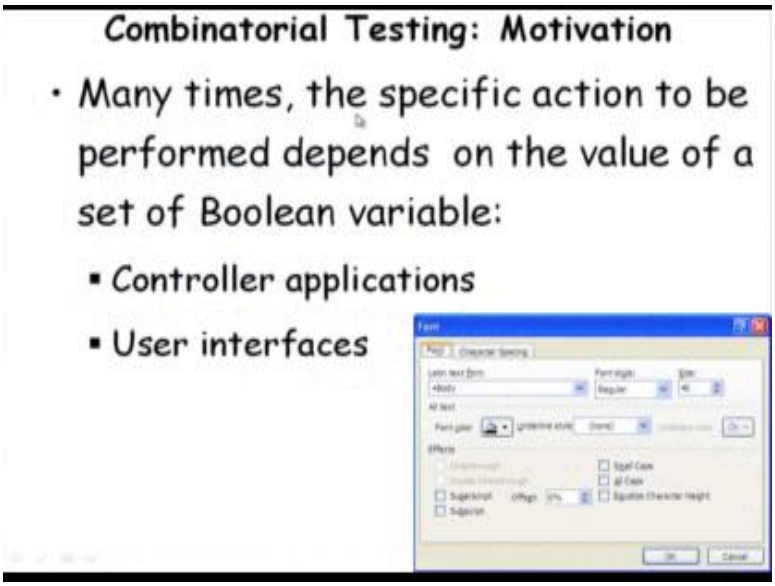
(Refer Slide Time: 00:48)



First, let us look at the motivation behind combinatorial testing. One thing is that in both equivalence class testing and in a special value testing, when the number of parameters arises, we will have difficulty in designing the test cases. Also sometimes, we have environmental configurations which also affect the result, for example, if we set a program in the expert mode, or in a novice mode, or moderate mode so these are environmental configurations the program behaves differently.

We might have other state variables, which might also affect different components of the program, their behaviour of the different components. For example, a book is issued out and then the behaviour of the book is issue procedure already issued out, and then we will have the book issue, behaviour of the book issue will be different. Just a very trivial case, but we might have more complex state representation for different components. So, in this case, it becomes difficult to design the equivalence classes, when the factors are the parameters are more than 3, 4.

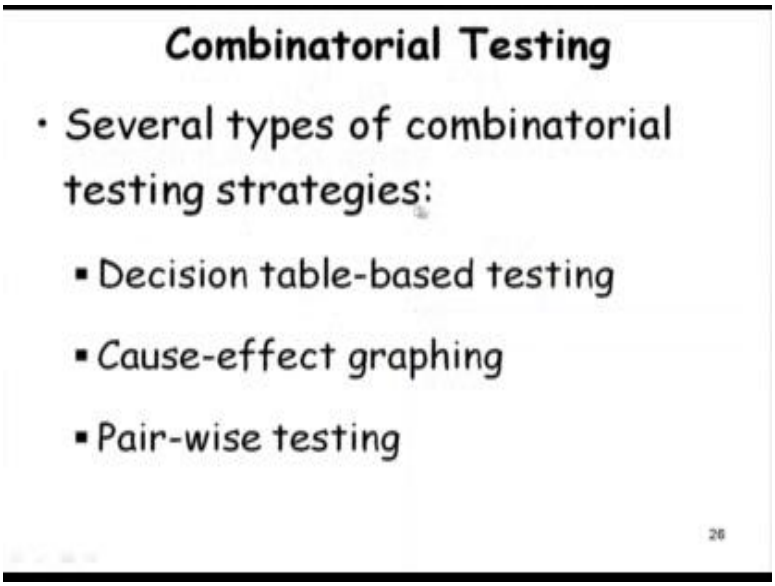
(Refer Slide Time: 02:29)



And also sometimes, we have too many Boolean conditions here. There are Boolean variables especially in user interfaces and controller applications, there are too many of these. For example, let us say this is a font setting for the power point software. And here just see depending on whether we select small cap, all cap, equalise, superscript, subscript etcetera. And also there are number of values here the size, the font style, the text font and so on, the colour etcetera. We have different action that would take place. So the font will look different.

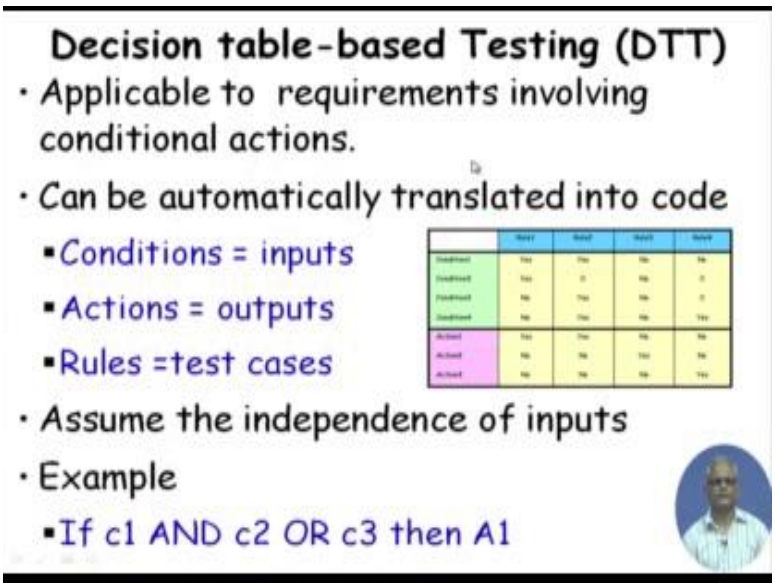
So we do not know whether for a specific combination of these settings, the font will get smudged or the font does not appear. For example, if the text font is body, and the font style is let say regular or bold, and then size is 38, and then font colour is red, and then superscript is switched on and all caps switched on, we have a problem. So, for these situations, how do we get the equivalence class partitions becomes difficult. Let us look at the combinatorial testing problem where we are trying to find out that there are many parameters, some parameters are directly input and some parameters are state variables or environment variables; and for specific combinations of those parameters, we have to test, because there might be problem.

(Refer Slide Time: 04:34)



We will look at three combinatorial testing techniques - the decision table-based testing, cause-effect graphing and pair-wise testing.

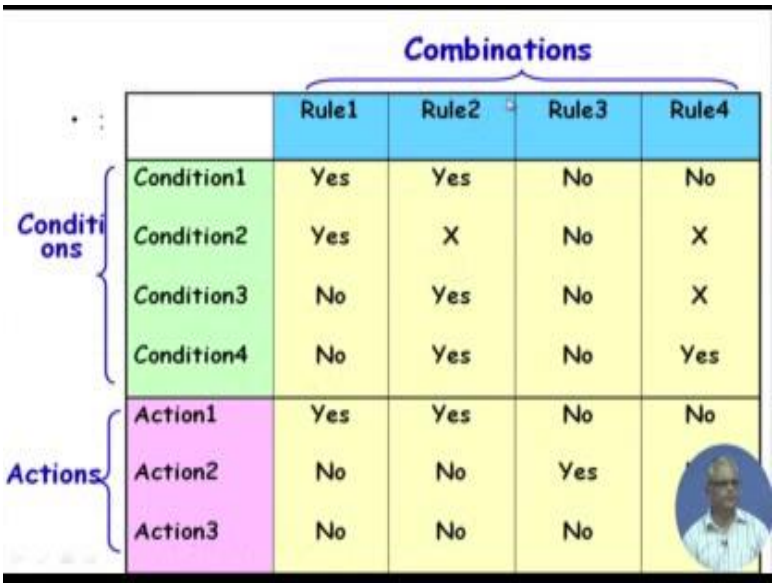
(Refer Slide Time: 04:52)



First, let us look at the decision table-based technique. In decision table-based testing we develop a decision table based on the problem description. So, this is the black-box specification for a problem, and then by looking at it, we develop this decision table. In the decision table, if you see here, we have the conditions appearing at the top of the table. So, these are the conditions on the input parameters.

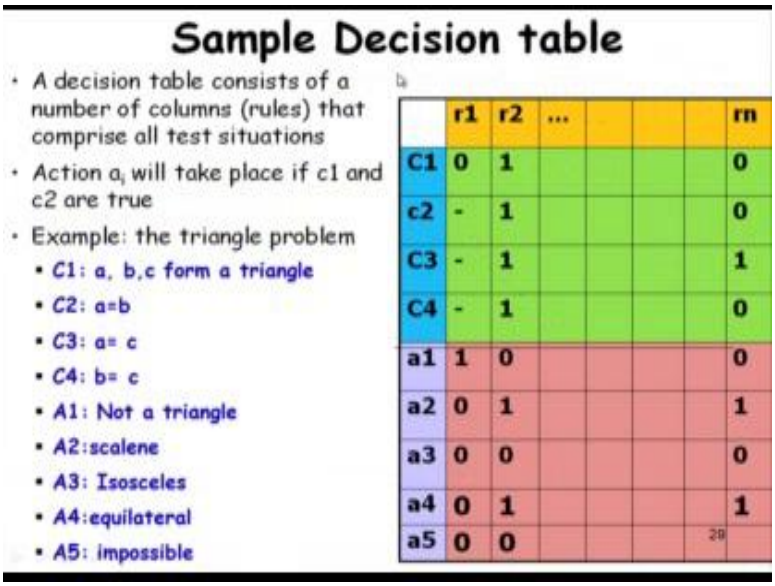
If something is switched on or something is switched off, combinations of those and so on; and then if specific conditions holds then we have some actions that take place. So, this we call as a rule. The conditions that hold so we have to consider all possible conditions and their combinations of values, and then we identify what actions would take place. There can be more than one action, which might be taking place. And each of these column here, we call it as a rule. For example, one of the condition here may be if c1 and c2 or c3, so c1, c2, c3 are the input parameters. And we have expression on the input parameter c1 is true and c2 is true or c3 is true then we have action A1.

(Refer Slide Time: 06:41)



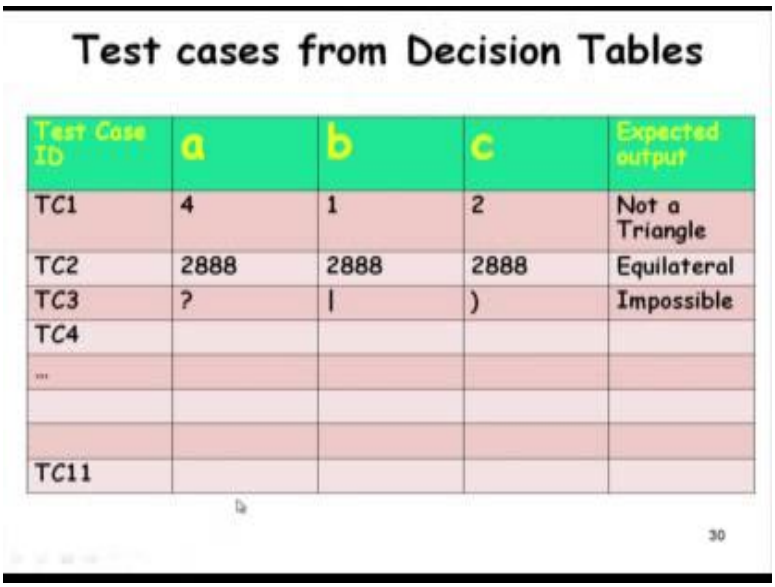
So, given a problem, where we have number of parameters, and based on some conditions on those parameters, we have some actions that take place. We can develop this decision table. And of course, we ( ) have to be careful that we have considered all possible combinations of conditions.

(Refer Slide Time: 07:12)



Now, let us look at an example. Let us say we have written a program, which reads three sides of a triangle. And then it displays whether the triangle, it is not a triangle based on the three sides you entered, it outputs not a triangle, it is a scalene, its isosceles, equilateral. So A5 is also as same as given; A 5 is actually invalid, we can consider. And we form the different conditions here, so c1 is a less than b plus c; b less than c plus a; c less than a plus b. And c2 is a is equal b; C 3, a is equal to c. And C 4, b is equal to c. So then depending on some specific combinations here, we would display these actions.

(Refer Slide Time: 08:21)



And based on this, the values of a, b, c, we can have the test cases. And we also have the expected output from the decision table, so each of these rows becomes a test case.

(Refer Slide Time: 08:44)

Decision Table for the Triangle Problem

Conditions										
C1: a < b+c?	F	T	T	T	T	T	T	T	T	T
C2: b < a+c?	-	F	T	T	T	T	T	T	T	T
C3: c < a+b?	-	-	F	T	T	T	T	T	T	T
C4: a=b?	-	-	-	T	T	T	T	F	F	F
C5: a=c?	-	-	-	T	T	F	F	T	T	F
C6: b=c?	-	-	-	T	F	T	F	T	F	T
Actions										
A1: Not a Triangle	X	X	X							
A2: Scalene										X
A3: Isosceles							X		X	X
A4: Equilateral				X						
A5: Impossible					X	X		X		

We can also write it in this form; a is less than b plus c; b is less than a plus c; and c is less than a plus b. And if any one of them is false, then we say that it is not a triangle. And if a is equal to b, b is equal to c, and a is equal to c, all these are true, then we write that it s a equilateral triangle and so on.

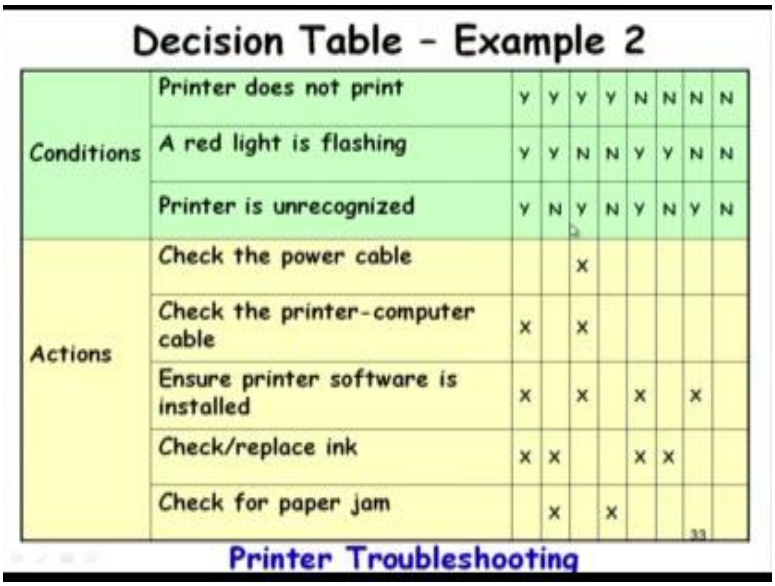
(Refer Slide Time: 09:22)

Test Cases for the Triangle Problem

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

So, we will have the test cases and their expected outputs.

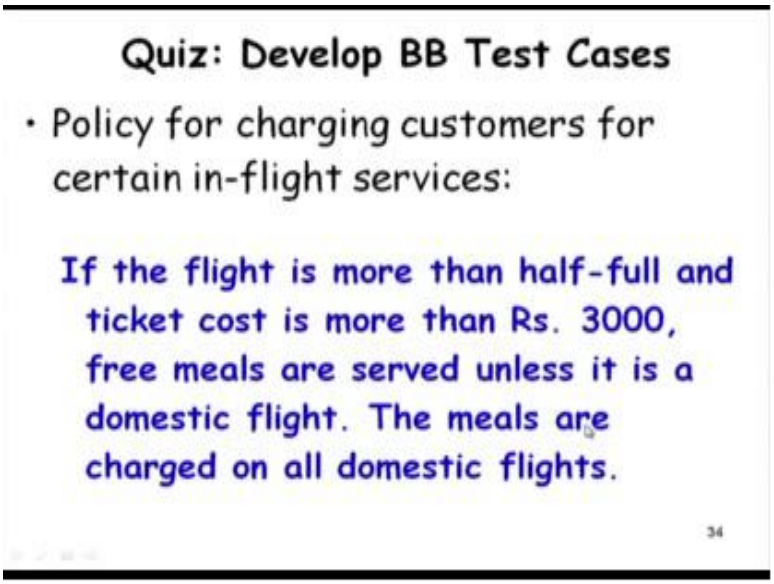
(Refer Slide Time: 09:29)



Now let us look at another example. Let us say we have printer diagnosis software where we enter the condition and it outputs the actions that need to be take place. For example, if the printer does not print and the red light is flashing; and printer is unrecognized then we have check and replace ink check paper jam.

If we have all three of them, yes, then it check the printer computer cable, hence your printer software is installed and check and replace ink. So, for specific conditions that we enter, the values of specific conditions that we enter, the program will display what action to take place, and then we have to consider all possible combinations of conditions because the user may enter any possible combinations; printer does not print and printer unrecognized or may be all three return and so on. So, we represent the decision table and then we consider each one as a rule and this forms our test case. So, we have the input values and we have also what actions should be expected, which is exactly what is required for a test case.

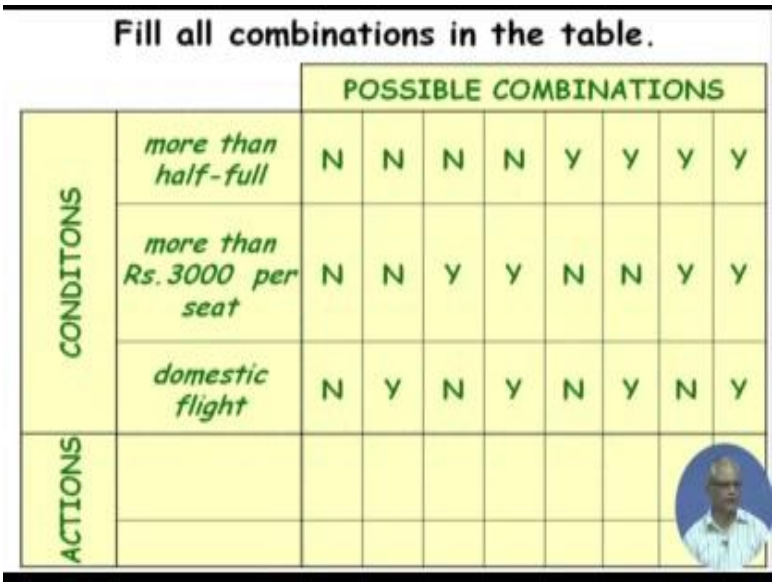
(Refer Slide Time: 11:08)



Now, let us have a small quiz. Suppose, we have policy on a flight that if the flight is more than half-full, the flight is more than half-full, and the ticket cost is more than 3000, then free meals are served unless it is a domestic flight. The meals are charged on all domestic flights.

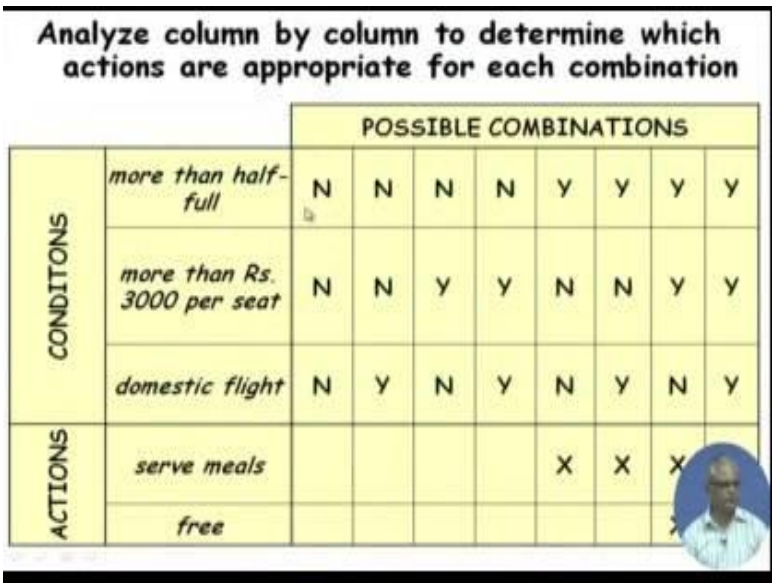
How do we develop the decision table testing? The decision table test cases have to be designed for such a situation. Now please try doing this. If the flight is more than half-full, so this becomes a parameter; whether the flight is half-full or not. If the ticket cost is 3000, or not more than 3000, or not and free meals are served is an action. And it is a domestic flight or not that is another input parameter. And for all domestic flights meals are charged. So, we need to develop the decision table for this. Please try this out, but let me just display the solution to this, so that you can check with our answer.

(Refer Slide Time: 12:44)



As we said that there are three input conditions, one is more than half-full, ticket cost is more than 3000, and whether it is a domestic flight or not.

(Refer Slide Time: 13:00)

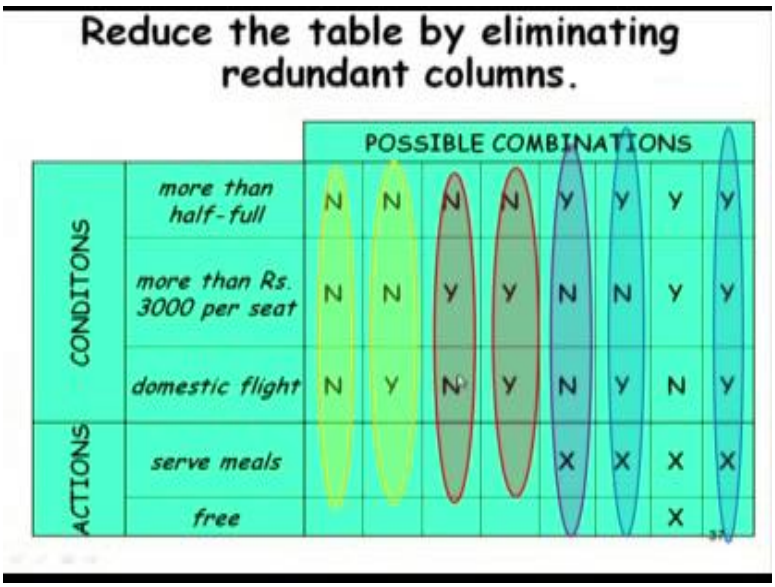


So, if all are no that it is not a domestic flight, more than 3000, not more than 3000, we do not serve free meals. If it is so as long as it is not more than 3000, we do not serve free meal. If it is a more than 3000, but it is a domestic flight, we do not serve free meal. And if it is not more than half-full, and it is domestic flight; obviously, we do not serve free meal, sorry, we do not serve meals.

A meal is served free only when it is more than half-full, more than 3000 per seat and it is not a domestic flight. But meals are served when it is more than half-full, and it is not a domestic flight, and ticket cost is not more than 3000. And if it is more than half-full, more than 3000 per seat, and it is domestic flight then meals are served, but that is not a free meal.

And then each of these becomes test case. So, we would require 1, 2, 3, 4, 5, 6, 7, 8, so for n parameters, if each one is a Boolean parameter, we need 2^n test cases. But then we have a scope to optimize this, because some of these are actually do not care terms; once it is a domestic flight, and its ticket cost is less than 3000, we do not have to really check whether it is more than half-full or not, becomes a do not care. Similarly, here, just check here that this is per both of this more than half-full, more than 3000, and if it is a domestic flight or not, we do not serve, so we can combine these two test cases into a single one.

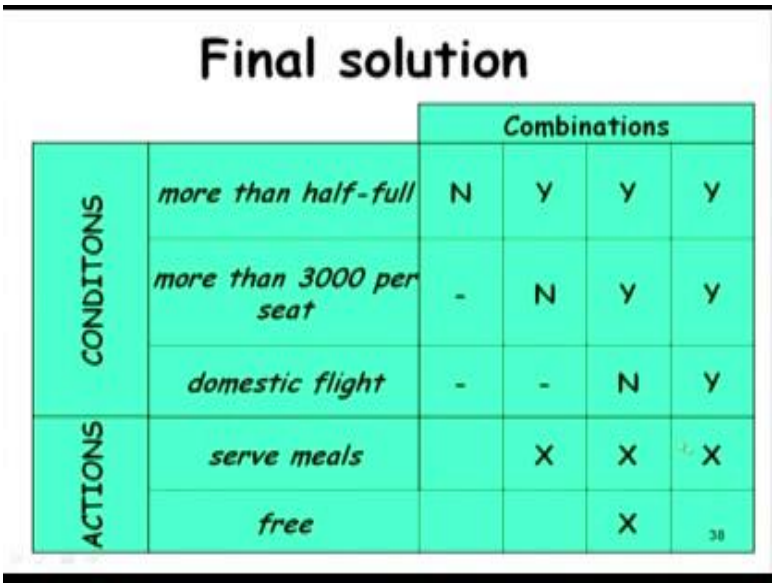
(Refer Slide Time: 15:23)



So, if we apply that we can remove the redundant test cases, we can combine these two, because these two produce same action; and irrespective as along as these two are no more than half-full and more than 3000 per seat. Irrespective, whether it is domestic or not, we do not serve meals neither free nor charged meals. Similarly, as long as this two are no and this is yes, we do not bother whether it is a domestic or not.

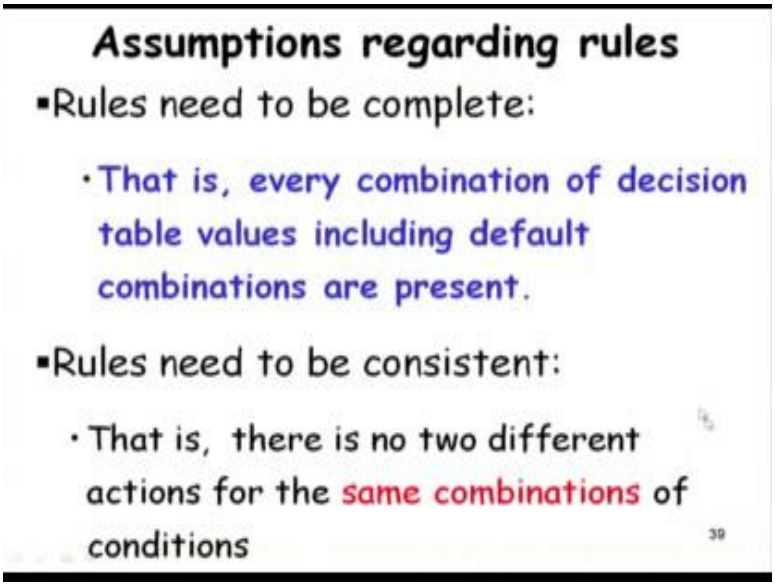
And similarly, between these two, as long as it is more than half-full, and it is a domestic flight, we do not bother whether it is a more than 3000 per seat or not, we just serve the meal (Refer Time: 16:26).

(Refer Slide Time: 16:27)



So, we can optimize that and we find four test cases.

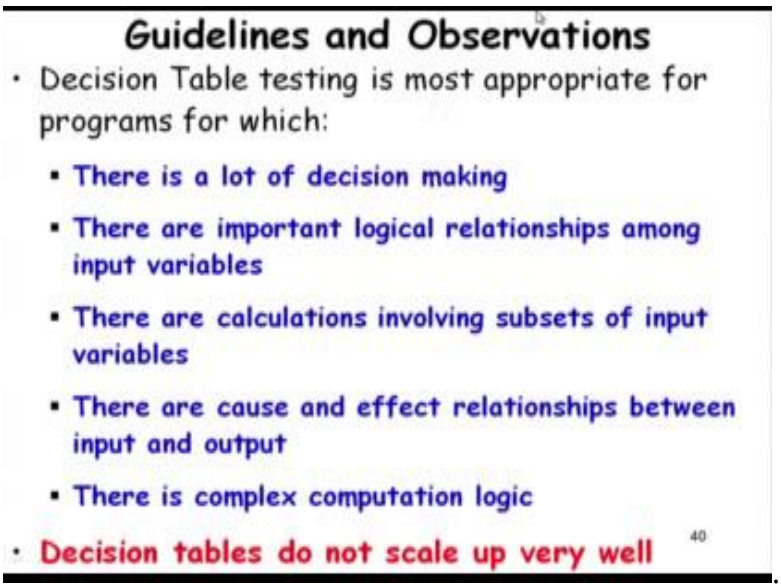
(Refer Slide Time: 16:34)



So, one thing we must remember I have told earlier that we have to consider all possible combinations of condition, because there is a chance that given many parameters we might miss out some combinations of conditions.

And also while forming the decision table, we have to ensure that there are two actions for the same combination, so it cannot be that the same combination will result in two different test cases, and then we have something wrong in our formulating the decision table.

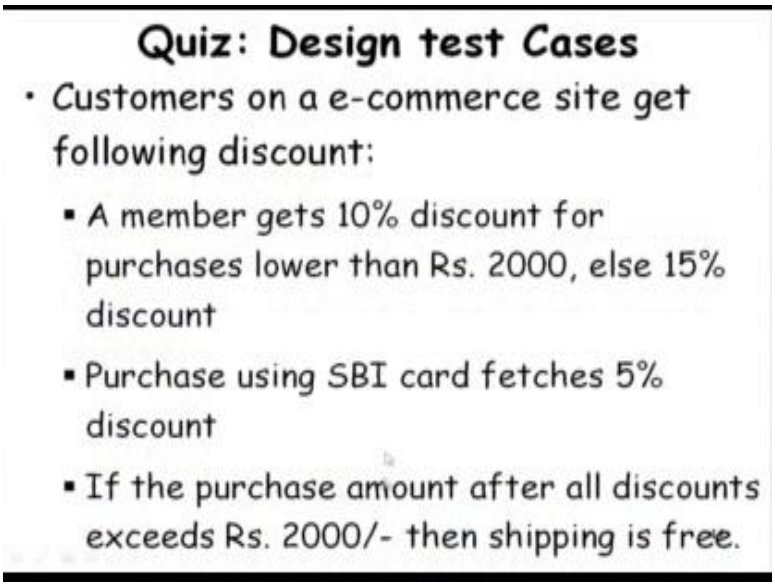
(Refer Slide Time: 17:15)



Now, let us see what are the problems where you can apply this decision table-based testing. When there is lot of decision making which is obvious in the problem statement, different actions take place depending on some combinations of the input conditions. Logical relationship between input variables, calculation involving subset of the input variables, and there is a cause effect relationship between input and output. So the output is caused by some values and the input.

But one difficulty with the decision tables is that if the number of parameters are 3, 4, 5, we can form the decision table. But let us say the number of parameter says 30, and each one takes let say 3, 4 values, so the number of columns will become 3^{20}. If we consider all possible combinations of 20 parameters, and each parameter takes three, values so the number of possible combinations of the conditions becomes 3^{20} which is just too many to handle manually and to form the test cases and then to optimize. We need to do something else so that we will look at it.

(Refer Slide Time: 18:49)



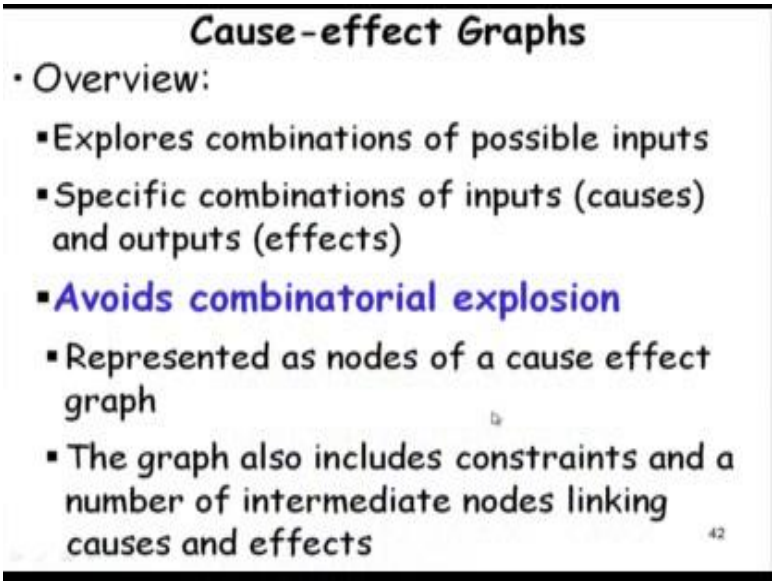
There is another quiz here. As I said and having telling many times that for designing test cases, we need lots of practise. Sometimes just knowing the theory, we really do not come up with the solutions unless we have practised these theories. So, let us try to design the decision table test for another problem. So, the problem is that an e-commerce site gives following discount. So, there is a program return at the e-commerce site, which displays the discount that is applicable to a customer.

Now there are different conditions based on which different discounts are given. A member gets 10 percent discount for purchase less than 2000. If the purchase is more than 2000 rupees then the customer gets 15 percent discount. And if the purchase is made by SBI card then gets 5 percent additional discount. And if the purchase amount after all discounts exceeds 2000, then shipping is free. So there are actions here; one is that what is the discount that is applicable, depending on the input value and the payment method, so these two are the parameters, what is the purchase amount and payment method.

So, we can form the decision table here based on the input conditions that whether the purchase amount is less than 2000, or it is 2000 and higher. And also whether it is a SBI card based payment or not. And then what is the total amount that is based on after the discount is applied. So, based on this, we have the actions that what is the discount rate,

is it 10 percent, 15 percent, 5 percent or 0 percent; and also is the shipping free. So, please form the decision table for this, and please form the test cases.

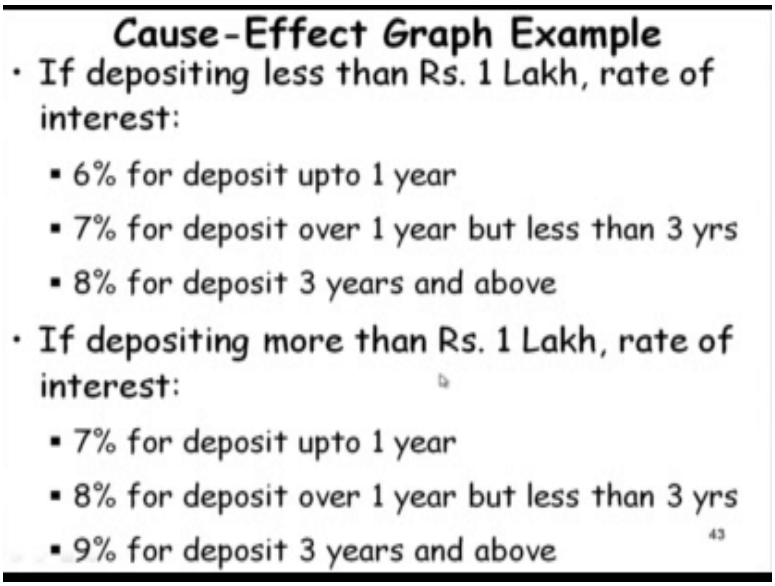
(Refer Slide Time: 21:37)



Another related combinatorial testing is the cause-effect graphing. The cause-effect graph to tell in brief is that it is a decision table testing actually, but it has specific notations by which we can by reading through a problem, we can then represent it in the form of a cause-effect graph, and we have a straight average method a straightforward method for coming up with the decision table. So to think of it, the cause-effect graphing method actually helps us to develop the decision table, so that may be the most succint statement to tell about cause-effect graph it has a set of notations by which given a problem we can represent the problem in the form of a cause-effect graph.

And once we have the cause-effect graph ready, we have a straightforward way to come up with the decision table, and then we can apply the decision table testing that is each column becomes a test case. So, here we look at the input and we call them as the causes, and the output as the effect. And then we look at various combinations of the causes and conditions. And here we look at only the possible ways in which the causes and combine to produce effect. So, we do not have to really consider all possible combinations as you are doing in the decision table development earlier. So in a sense, it avoids the combinatorial explosion problem.

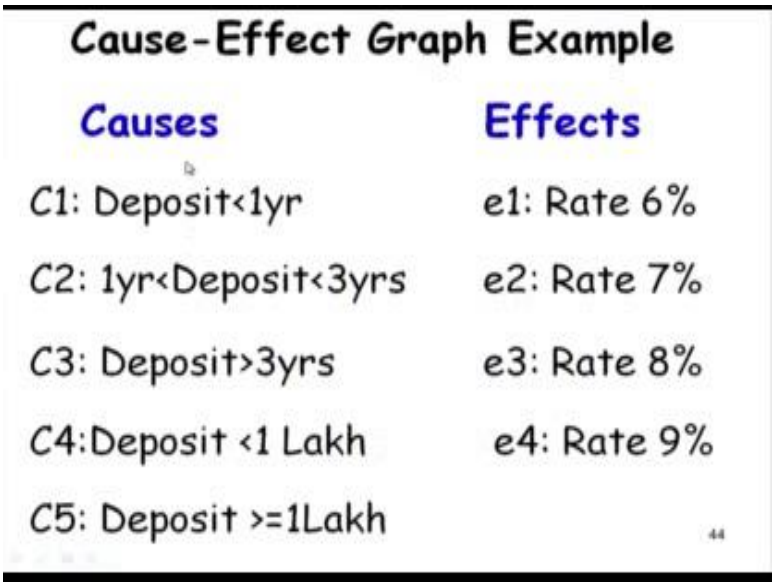
(Refer Slide Time: 23:38)



Let us look at the cause-effect graphing with an example, because the crux here lies in developing the cause-effect graph. And once we have the cause-effect graph, it is basically decision table based testing. Let us look at this problem and then try to represent it in the form of a cause-effect graph. It is very simple without really discussing much about cause effect graph, we can actually try to solve this example, and then you would be aware about what is cause-effect graph.

Let us look at the problem. In a bank, if we deposit less than 1 lakh rupees as the principle, then the rate of interest is applicable is 6 percent, 7 percent, or 8 percent depending on the deposit is for 1 year, 3 year, or 5 years. And if the deposit is more than 1 lakh then the rate of interest is 7 percent, 8 percent, 9 percent depending on whether it is less than 1 year, between 1 to 3 year, or 3 years and above. So let us try to develop the cause-effect graph for this problem.

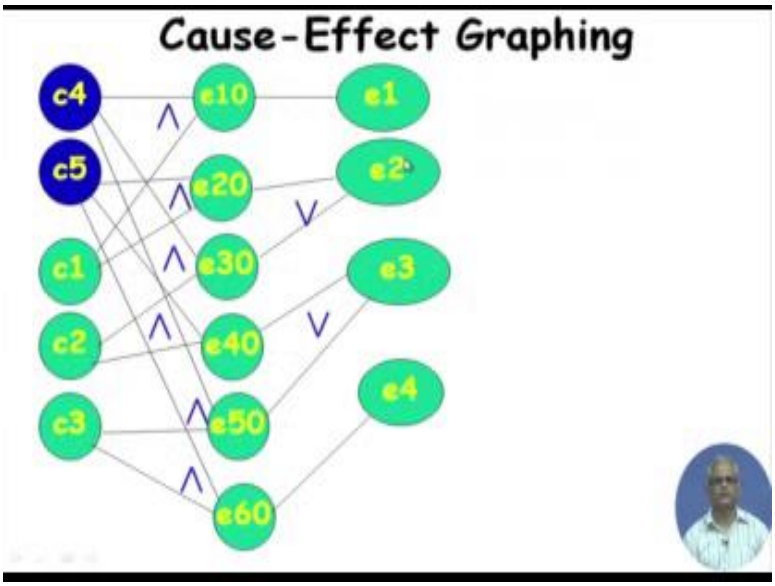
(Refer Slide Time: 24:57)



Let us identify the causes and the effects first. If deposit is for less than 1 year, these are the different inputs; deposit is less than 1 year; deposit is between 1 year and 3 year; deposit is greater than 3 year; deposit is less than 1 lakh; and deposit is greater than 1 lakh, so these are all are input causes. And then the effects are 6 percent, 7 percent, 8 percent, 9 percent rates.

And specific combinations of these causes produce some effect for example, the deposit is less than 1 year and deposit is less than 1 lakh, it will produce 6 percent. If the deposit is 1 year to 3 year and deposit is less than 1 lakh, it will produce 7 percent. And if deposit is less than 1 year and deposit amount is more than 1 lakh then also it will produce 7 percent.

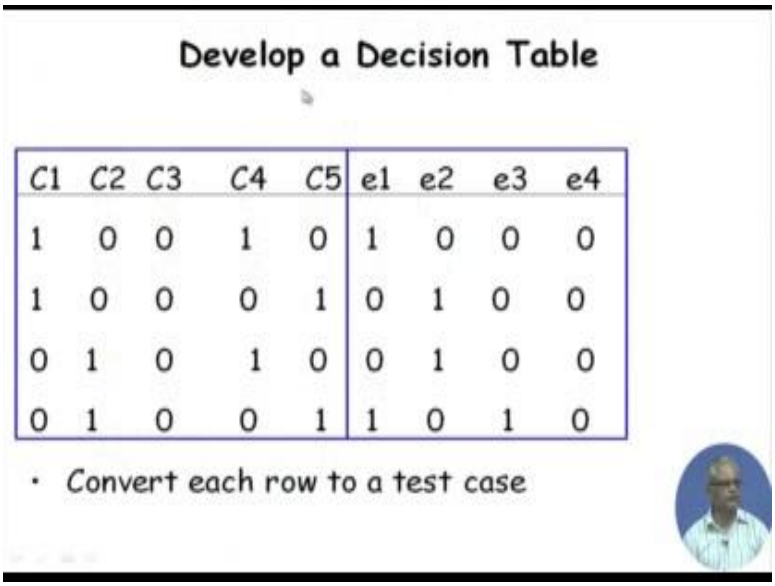
(Refer Slide Time: 26:06)



Now, let us represent these aspects. So, the causes and the effects, we denote in the form of circles, nodes and we have written all the 5 causes, and then whenever there are combinations of causes which produce some effect then we have this intermediate nodes appearing here.

We might have multiple levels of intermediate nodes, if for more complex problems. So, here if it is less than 1 year and if it is less than 1 lakh, so this is and condition here; so if both of these are two then it produces 6 percent interest rate. And if the deposit is greater than 1 lakh and it is for less than 1 year, it produces 7 percent; and if it is less than; so c4 need to just look up, c4 is I think, it is less than 1 lakh, and it is between 1 to 3 year, it produces 7 percent. And similarly, if deposit is more than 1 lakh and it is less than 1 year, it produces 7 percent and so on.

(Refer Slide Time: 27:54)



So, once we have it, then it is straightforward to develop the decision table. So, we just write the conditions and then the corresponding actions, the straightforward translation of the cause-effect graph. And once we have the decision table ready, we can apply the decision table testing that each column becomes a test case. So cause-effect graphing is very simple technique, to come up with the decision table, avoids the exponential combination of test cases that need to be considered in the case of a decision table-based testing.

(Refer Slide Time: 28:32)

But then for very complicated problems, the number of test cases can become very huge. For example, let us say we have a situation where the number of conditions, input conditions are 50 and each one takes either true or false, or may be that take multiple values. And there are several situations where such problems arise, specially the context of user interfaces and embedded controller software, so in these cases really designing 2^{50}, so for let us say 50 input parameters and each one takes two values.

Number of test cases required for a just if testing is 2 to the power 50. So, we try to test all possible combinations of conditions becomes 2 to the power 50, and which is huge number, cannot really test in any reasonable period of time. So, we need to have some effective ways of doing adequate testing with less number of test cases, and that is all about the pair-wise testing, where we show that we do not have to really consider all possible combinations of input values in both decision table-based testing and cause-effect graphing, we were considering all possible input combinations. And here we will not consider them.

And we will discuss about the pair-wise testing in the next lecture.

Thank you