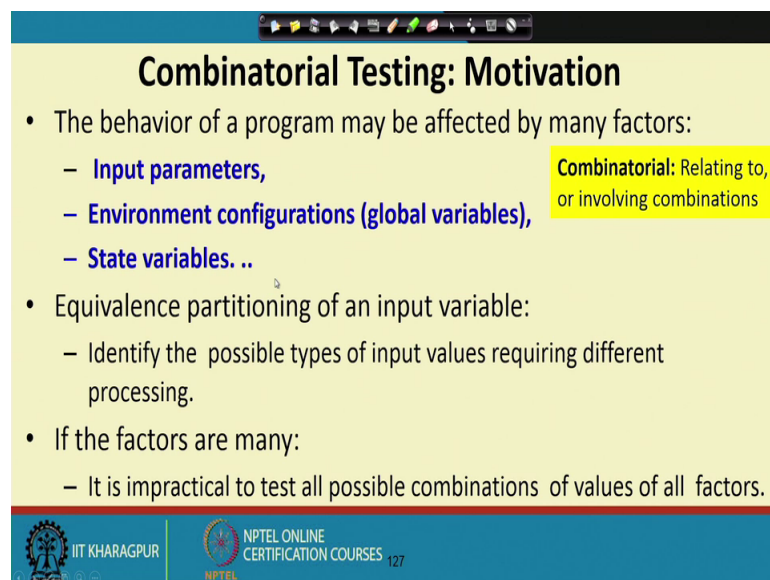


Lecture – 52
Decision Table Testing

Welcome to this lecture. In the last lecture we had started to discuss about combinatorial testing. We will discuss these testing techniques today, in further detail we will first revisit what we discussed in the last lecture and then we will discuss further.

(Refer Slide Time: 00:40)



Combinatorial Testing: Motivation

- The behavior of a program may be affected by many factors:
 - **Input parameters,**
 - **Environment configurations (global variables),**
 - **State variables. ..**
- Equivalence partitioning of an input variable:
 - Identify the possible types of input values requiring different processing.
- If the factors are many:
 - It is impractical to test all possible combinations of values of all factors.

Combinatorial: Relating to, or involving combinations

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 127

The word combinatorial, if you look in the dictionary, it means relating to or involving combinations.

The idea here is that in testing the software, there are multiple inputs and conditions. And, in the previous lectures, we had seen how to design black box test cases based on the input values, you looked at equivalence class boundary values and so on. But, then when there are multiple inputs, we need to design test cases based on various possible combinations among these input. And, also there can be global variables and so on. And, therefore, the combinations must be considered and of course, the combinations can become too many and then it becomes difficult to test. And, we will today see how to reduce the number of combinatorial test cases.

First, we will discuss the combinatorial test case design and, later we will see how to reduce the number of test cases that are designed. The behavior of a program is affected by many factors. For example, the specific parameters we give as input and, there can be data stored as global variables for example, arrays in a library example may be books, members, these are large arrays and their state changes depending on whether a book is issued returned and so on. And, there can be state variables, which define the state of the software.

We had seen that in the previous lecture that for every input variable, using the black box testing technique of equivalence class boundary values and so on, we can design test cases. And, then we have to consider various possible ways of combining this inputs, but the problem is that often the input maybe too many; there may be many parameters environmental variable state variable and so on.

And the number of test cases required for all possible combination may become trillions and the test should become extremely large. And, then we will see how to address this problem? And, reduce the number of test cases. But, first let see the basic idea behind combinatorial generation of test cases and then we will see how to reduce the number of tests.

(Refer Slide Time: 04:07)

Combinatorial Testing: Motivation

- Many times, the specific action to be performed depends on the value of a set of Boolean variable:
 - Controller applications
 - User interfaces

The slide also features a screenshot of a Windows 'Font' dialog box. The 'Font' tab is active, showing 'Latin text font' set to 'Arial', 'Font style' set to 'Regular', and 'Size' set to '10'. The 'Effects' section includes checkboxes for 'Strikethrough', 'Double Strikethrough', 'Superscript', 'Subscript', 'Small Caps', and 'All Caps', along with 'Underline style' (None) and 'Underline color' (Default). The 'OK' and 'Cancel' buttons are at the bottom.

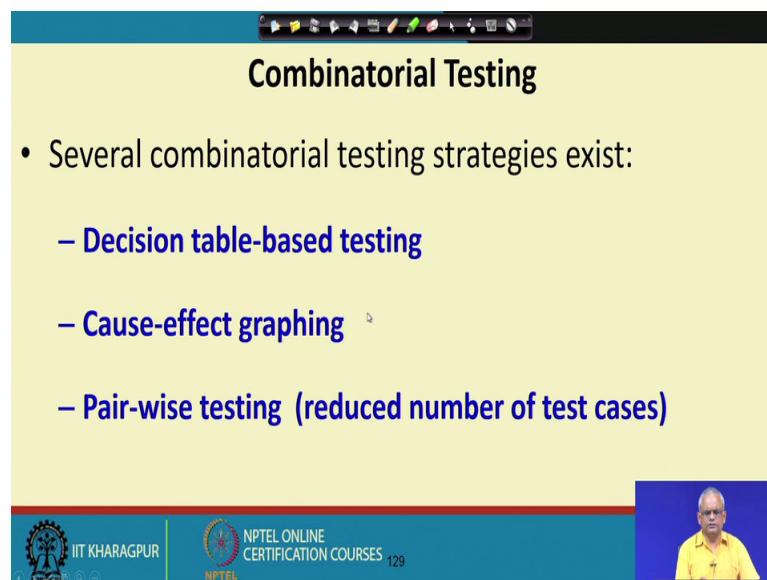
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 128

Let us look at one specific test situation; this is a user interface for a word processing software. You might have used similar software, where we can define the font

characteristics. And, see here we have many things to choose here. For example, whether the entered font is a superscript, subscript, small cap, all cap, then the font color, then the specific font type and whether it is a regular italic and so on and the size of the font and so on. This kind of interface is very common in user interfaces, but then if we want to test based on all possible combinations of the input, then it may become too many.

For example, we would like to check, whether it works well in superscript and when text font is something and text color is something. And, whether it works as subscript when for the same font, font color size and so on. This kind of situation is also there in controller applications, where the controller behavior is defined by many factors. For example, temperature pressure user input and other factors.

(Refer Slide Time: 05:51)



Combinatorial Testing

- Several combinatorial testing strategies exist:
 - **Decision table-based testing**
 - **Cause-effect graphing**
 - **Pair-wise testing (reduced number of test cases)**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 129

With this motivation, we will discuss 3 main types of combinatorial testing One is called as a decision table based testing cause effect graphing and pair wise testing. The pair wise testing actually reduces the number of test cases, which may be obtained through a decision table or cause effect graphing.

(Refer Slide Time: 06:21)

The slide features a yellow background with a list of bullet points on the left and a table on the right. A yellow box with the text 'Decision table-based Testing (DTT)' is positioned above the table. At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with the number 130.

- Applicable to requirements involving conditional actions.
- This is represented as a decision table:
 - Conditions = inputs
 - Actions = outputs
 - Rules =test cases
- Assume independence of inputs
- Example
 - If c1 AND c2 OR c3 then A1

	Rule1	Rule2	Rule3	Rule4
Condition1	Yes	Yes	No	No
Condition2	Yes	X	No	X
Condition3	No	Yes	No	X
Condition4	No	Yes	No	Yes
Action1	Yes	Yes	No	No
Action2	No	No	Yes	No
Action3	No	No	No	Yes

Let us revisit the decision table based testing. In decision table the upper rows of the table are the decision are the conditions, which are basically the inputs. And, the lower rows are the actions and these are the output and this table allows us to consider various possible combinations of the conditions.

If, there are 4 conditions here and all conditions are Boolean, then if we consider all possible combinations we will get 16 rows here, but the conditions may not be Boolean, it can be let say an integer taking value 1 to 10 for a choice and so on. So, you can know once we list down the conditions and actions, we know that what are the possible values of the input here and we can define the rules based on the specific values of the inputs distinct values of the input. And, then identify which of the actions take place for that specific combination of conditions.

And, each of the column here becomes a test case and this is called as a rule, because this is a specific combination of conditions and each of these become a test case, just to get a feel of it let us try to do the decision table for this expression, that is given here. Here, let us assume that c 1 c 2 c 3 are Boolean expressions and if this is true then A 1 is the action.

But, if the outcome of this specific combination is false, then there is no action, but this is the simplest as I was saying, we might have much more complex where we might have several actions possible depending on specific outcome of the expression.

Just to explain the concept, how to draw the decision table for a expression like this, we need to write the conditions here C 1 C 2 and C 3. These are the upper rows of the decision table and the lower row are the actions just put a line here and we will write conditions or inputs and will write here the action.

(Refer Slide Time: 09:11)

• Applicable to requirements involving conditional actions.

• This is represented as a decision table:

–Conditions = inputs

–Actions = outputs

–Rules =test cases

• Assume independence of inputs

• Example

–If c1 AND c2 OR c3 then A1

Decision table-based Testing (DTT)

C1	T	T	T			
C2	T	T	F			
C3	T	F	F			
Action	A1	Y	N			

The slide also features the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and a small video inset of a presenter in the bottom right corner.

And, we will consider various combinations of conditions of the truth value of these conditions. And, each of these we will call as a rule or a test case let say C 1 is true, C 2 is true, C 3 is true and therefore, the action A 1 is yes it will be taken.

If C 1 is true C 2 is true and C 3 is false then still it is yes, but if C 1 is true C 2 is false and C 3 is false, then it will be no and so on. So, this is the simple way to construct a decision table given a simple expression, we should be able to write the decision table for that and these become the test cases in addition table based testing.

(Refer Slide Time: 10:57)

Combinations

		Rule1	Rule2	Rule3	Rule4
Conditions	Condition1	Yes	Yes	No	No
	Condition2	Yes	X	No	X
	Condition3	No	Yes	No	X
	Condition4	No	Yes	No	Yes
Actions	Action1	Yes	Yes	No	No
	Action2	No	No	Yes	No
	Action3	No	No	No	Yes

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the idea here is that we write the conditions, the top rows actions in the bottom rows and consider various combinations of conditions and each of these become a test case and we call them as a rules.

(Refer Slide Time: 11:20)

- A decision table consists of a number of columns (rules) that comprise all test situations
- Example: the triangle problem
 - C1: a, b,c form a triangle
 - C2: a=b
 - C3: a= c
 - C4: b= c
 - A1: Not a triangle
 - A2:scalene
 - A3: Isosceles
 - A4:equilateral
 - A5: Right angled

Sample Decision table

Check triangle (a, b, c)

	r1	r2	rn
C1	0	1				0
C2	-	1				0
C3	-	1				1
C4	-	1				0
a1	1	0				0
a2	0	0				1
a3	0	0				0
a4	0	1				0
a5	0	0				0

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 132

Let us take a very simple example. Let us assume that we are trying to design the decision table based testing for a function, which is check triangle, check triangle the name of the function and takes 3 integers a b c as it is input. And, the function will display that the specific triangle is a scalene isosceles equilateral not a triangle and so on.

Now, here the conditions based on which the action had taken can write here the top rows, C 1 is that it forms a triangle, C 2 a equal to b, C 3 is a equal to c, C 4 is equal is b equal to c.

So, if it is a triangle if it is a equal to b, b equal to c, c equal to a, then it is a A 4 is a equilateral triangle, S if it is not a triangle then it will display not a triangle that is A 1 and so on. We can easily develop this table, but the problem is that here we have written here the condition 1 is a b c form a triangle, but that is not a very practical condition to check, we will see that we can refine the table.

(Refer Slide Time: 13:13)

Test cases from Decision Tables

Test Case ID	a	b	c	Expected output
TC1	4	1	2	Not a Triangle
TC2	2888	2888	2888	Equilateral
TC3	?)	Impossible
TC4				
...				
TC11				

C1: a, b, c form a triangle
 C2: a=b
 C3: a= c
 C4: b= c

But, once we have drawn the decision table, we can design the test cases which are specific combinations of the conditions and the corresponding actions are the output to be checked. And, if C 1 is that a b c is not a triangle is true then the display will be not a triangle irrespective of the condition C 2 C 3 C 4.

We, can do that write the specific value which will satisfy condition 1, which is 4 1 2. And, then we say the expected output is not a triangle. Similarly, if it is a triangle and also a equal to b, b equal to c, c equal to a, then we write some specific values which satisfies that condition and the expected output is equilateral. And, each of these is a test case we give the test case number here test case id and that is how we design the equivalent the decision table based test cases?

(Refer Slide Time: 14:32)

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

This is a more complete decision table for the example we discussed, because in the top row we have replaced, whether it forms a triangle or not with specific conditions, which are easier to check a is less than b plus c b is less than a plus c c equal to a plus b . And, if all of this is true then it forms a triangle. As long as it is false as long as it is false, then the output is not a triangle, but if all the 3 conditions where $C_1 C_2 C_3$ is true, then it forms a triangle.

And therefore, we check the other conditions $C_4 C_5 C_6$ and based on that we write whether it is a scalene isosceles equilateral or it is impossible, impossible is that it is not integer values and so on.

(Refer Slide Time: 15:58)

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

And, once we have develop the table, we can write specific values for the input and we write the expected output and the all of these form a test case.

(Refer Slide Time: 16:16)

Decision Table – Example 2	Conditions	Printer does not print	Y	Y	Y	N	N	N	N	
		A red light is flashing	Y	Y	N	N	Y	Y	N	N
		Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Printer Troubleshooting	Actions	Check the power cable			X					
		Check the printer-computer cable	X		X					
		Ensure printer software is installed	X		X		X		X	
		Check/replace ink	X	X			X	X		
		Check for paper jam		X		X				

Let us take one more example, which is the printer troubleshooting example. Here depending on some specific condition the program recommend some action to take. For example, if the printer does not print red light is flashing and printer is unrecognized. It recommends check printer cable ensure printer software is installed and check replace check or replace ink. If, it is printer is not printing and red light is flashing only there is

no printer unrecognized message. And, then need to check and replace ink and check for paper jam.

Similarly, if there is printer does not print and printer is unrecognized, then we need to check the power cable for the printer. And we need to check the printer cable and ensure that the printer software is installed and so on, we develop the table and each of the row columns here become a test case.

(Refer Slide Time: 17:33)


Quiz: Develop BB Test Cases

- Policy for charging customers for certain in-flight services:

If the flight is more than half-full and ticket cost is more than Rs. 3000, free meals are served unless it is a domestic flight. Otherwise, no meals are served. Meals are charged on all domestic flights.

		C1		C2		
						C3
		C1	Y	Y	-	-
		C2	Y	Y	-	-
		C3	Y	N	-	-
		A1	Y	Y	-	-
		A2	N	Y	-	-

Meals Served Free



Let us develop the test case decision table test case for a specific situation. If a flight is more than half-full and ticket cost is more than 3000 free meals are served, unless it is a domestic flight. In domestic flight meals are served, but these are not free, but if the flight is not half full is less than half-full and ticket cost is less than 3000 or ticket cost is less than 3000 then no meals are served.

To develop the decision table for this we need to identify the conditions. The specific conditions here is that whether the flight is more than half-full, we will write that as condition 1, ticket cost is more than 3000 we will write that as condition 2, the actions is whether free meal is served or meal is served and whether it is a domestic flight.

So, this is the third condition here the flight is more than half-full ticket cost is more than 3000 and it is a domestic flight we can identify that these are the conditions, and the actions are that whether meal is served and whether it is free.

So, we can write here C 1 C 2 C 3 and action 1 action 2. Action 1 is whether meal is served and Action 2 is whether it is free. And, we will check whether C 1 is that flight is more than half-full ticket cost is more than 3000. And, it is a domestic flight, then meal is served A 1 let me just write here meal is served and A 2 is that it is free meal. So, will write here, yes and A 2 is no. Similarly, if it is more than half-full. And, it is ticket cost is more than 3000, but it is not a domestic flight then meal is served and also it is free and so on. Then find all possible combinations of the conditions and each of the columns here will become a test case.

(Refer Slide Time: 20:58)

Fill all combinations in the table.		POSSIBLE COMBINATIONS							
		C1	C2	C3	A1	A2	A3	A4	
CONDITONS	<i>more than half-full</i>	N	N	N	N	Y	Y	Y	Y
	<i>more than Rs.3000 per seat</i>	N	N	Y	Y	N	N	Y	Y
	<i>domestic flight</i>	N	Y	N	Y	N	Y	N	Y
ACTIONS									

So, I just drawn here for your reference that these are the 3 conditions and then the specific actions write all possible combinations of the conditions. If there are 3 here and each is a binary then we will have 2 to the power 3 8 that is the number of columns here, it will give all possible combinations of the conditions of these 3 input Boolean variables.

(Refer Slide Time: 21:28)

Analyze column by column to determine which actions are appropriate for each combination		POSSIBLE COMBINATIONS							
		1	2	3	4	5	6	7	8
CONDITIONS	<i>more than half-full</i>	N	N	N	N	Y	Y	Y	Y
	<i>more than Rs. 3000 per seat</i>	N	N	Y	Y	N	N	Y	Y
	<i>domestic flight</i>	N	Y	N	Y	N	Y	N	Y
ACTIONS	<i>serve meals</i>					Y	Y	Y	Y
	<i>free</i>							Y	

In this we have filled up the action part and these each of these become a test case, and then the corresponding actions are checked when the test case is executed.

(Refer Slide Time: 21:46)

Reduce the table by eliminating redundant columns.		POSSIBLE COMBINATIONS							
		1	2	3	4	5	6	7	8
CONDITIONS	<i>more than half-full</i>	N	N	N	N	Y	Y	Y	Y
	<i>more than Rs. 3000 per seat</i>	N	N	Y	Y	N	N	Y	Y
	<i>domestic flight</i>	N	Y	N	Y	N	Y	N	Y
ACTIONS	<i>serve meals</i>					X	X	X	X
	<i>free</i>							X	

But, then even using a decision table we can optimize the number of test cases, we can reduce little bit. For example, this is the decision table that we developed, but then we can see here that the action part is the same here. The action part is the same here there is no action for these 2 test cases or a rules, but then the specific input values are same for both of these conditions and this is just no and yes. And therefore, we can combine this

into a single rule and we will write do not care here, it is possible to combine these 2 because the action part is the same and irrespective of the domestic flight, nothing is done if it is less than 3000 and less than half-full.

Similarly, we can combine these 2. The action part is the same and then they differ with respect to this. So, this becomes a do not care, but what about this we cannot combine with any other, because action we do not have ok. We have this here, the same action here, but then ok. We can combine either this with this that these 2 are these 2 are having same action or we can combine this rule with this rule. So, they have the same action part. And, then they differ with respect to only more than 3000. So, you can combine this into single rule and then we cannot combine this further with this, because your already do not care here.

So, I have drawn that reduced the number of test cases and I have written here do not care.

(Refer Slide Time: 24:12)

Final solution		Combinations			
CONDITONS	<i>more than half-full</i>	N	Y	Y	Y
	<i>more than 3000 per seat</i>	-	N	Y	Y
	<i>domestic flight</i>	-	-	N	Y
ACTIONS	<i>serve meals</i>		X	X	X
	<i>free</i>			X	

So, the number of test cases required here is 4.

(Refer Slide Time: 24:36)

Assumptions regarding rules

- Rules need to be complete:
 - That is, every combination of decision table values including default combinations are present.
- Rules need to be consistent:
 - That is, there is no two different actions for the same combinations of conditions

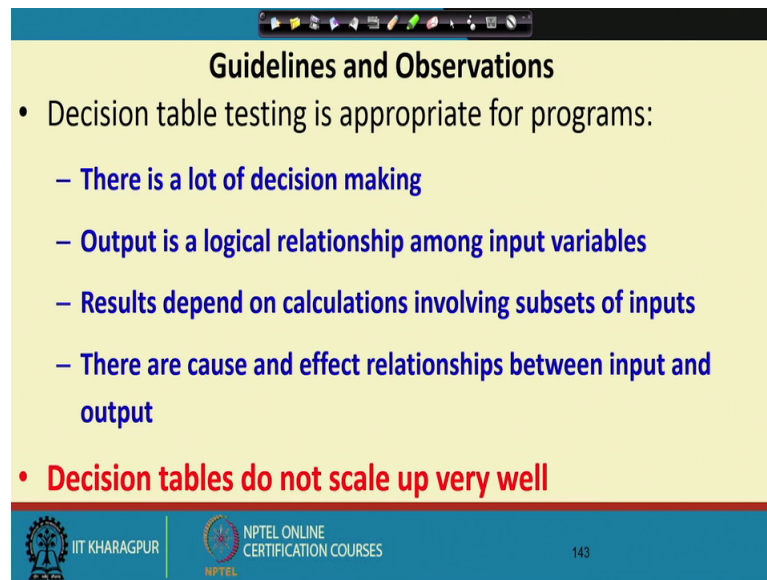
	Y	Y	Y
	Y	Y	Y
A	Y	Y	N
A	Y	Y	Y

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 142

Now, in the decision table based testing, we must ensure that all possible combinations of the decisions of the conditions are taken care. And, also we must ensure that the rules are consistent, that is it should not be the case, that for the same combination of condition we are writing 2 different actions that is inconsistency, basically basic if we write let say 3 conditions yes and yes and we write that action A 1 is yes, and A 2 is no. And, for another rule we write same combination of condition and the actions are different, it cannot happen because this is inconsistency.

We must take care that the table is consistent, and all possible combinations of conditions are present and that will give us our decision based test cases.

(Refer Slide Time: 25:53)



Guidelines and Observations

- Decision table testing is appropriate for programs:
 - There is a lot of decision making
 - Output is a logical relationship among input variables
 - Results depend on calculations involving subsets of inputs
 - There are cause and effect relationships between input and output
- **Decision tables do not scale up very well**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 143

The decision based test cases decision table based test cases are relevant for many problems. If, we look at the code and see that there is lot of decision not code. If we look at the description of the soft program and see that the specification involves lot of decision making. The output is a logical relation among input variables.

The results depend on calculation involving subsets of input or there are cause and effect relationship between input and output, these are the cases where we have to design the decision table based test cases. But, the only problem with the decision table based test cases is that, as the number of input conditions become large, the table becomes the table size grows exponentially if A 3 conditions, Boolean conditions we need 8 test cases.

If, we have 4 we need 16, 5 32 and so on if we have 10 then 1000. So, becomes difficult to develop the table and the number of test cases grows very rapidly, it becomes difficult to optimize manually reduce the number of test cases and so on. Also, the decision table based test cases even though it is very simple if we understand the problem very well, but then if we do not understand the problem very well; it becomes difficult to develop the decision table.

And, we will look at another technique which is cause effect graphing, which gives us a graphical way to represent the input conditions output conditions and from there we can develop the decision table. We are already at the end of this lecture, we will stop here continue in the next lecture.

Thank you.