A weaker adequacy criterion based on the notion of modified condition decision coverage, also known as an *MC/DC coverage*, allows a thorough yet practical test of all conditions and decisions. As is implied in its name, there are two parts to this criteria: the MC part and the DC part. The DC part corresponds to the decision coverage already discussed earlier.

The next example illustrates the meaning of the *MC* part in the MC/DC-coverage criteria. Be forewarned that this example is merely illustrative of the meaning of *MC* and is not to be treated as an illustration of how one could obtain an *MC/DC* coverage. A practical method for obtaining the *MC/DC* coverage is given later in this section.

Example 6.22: To understand the MC portion of the MC/DC coverage, consider a compound condition $C = (C_1 \text{ and } C_2)$ or C_3, where C_1, C_2, and C_3 are simple conditions. To obtain MC adequacy, we must generate tests to show that each simple condition affects the outcome of C *independently*. To construct such tests, we fix two of the three simple conditions in C and vary the third. For example, we fix C_1 and C_2 and vary C_3 as shown in the first eight rows of Table 6.6. There are three ways of varying one condition while holding the remaining two constant. Thus, we have a total of 24 rows in Table 6.6.

Many of these 24 rows are identical as is indicated in the table. For each of the three simple conditions, we select one set of two tests that demonstrate the independent effect of that condition on C. Thus, we select tests (3, 4) for C_3, (11, 12) for C_2, and (19, 20) for C_1. These tests are shown in Table 6.7 that contains a total of only six tests. Note that we could have as well selected (5, 6) or (7, 8) for C_3.

Table 6.7 also has some redundancy as tests (2, 4) and (3, 5) are identical. Compacting this table further by selecting only one among several identical tests, we obtain a minimal MC-adequate test set for C shown in Table 6.8.

The key idea in Example 6.22 is that every compound condition in a program must be tested by demonstrating that each simple condition within the compound condition has an independent effect on its outcome. The example also reveals that such demonstration leads to fewer tests than required by the multiple-condition-coverage criteria. For example, a total of at most eight tests are required to satisfy the multiple condition criteria when condition $C = (C_1 \text{ and } C_2)$ or C_3 is tested. This is in contrast to only four tests required to satisfy the MC/DC creation.

6.2.9 MC/DC-Adequate Tests for Compound Conditions

It is easy to improve upon the brute-force method of Example 6.22 for generating MC/DC-adequate tests for a condition. First, we note that

Table 6.6 Test cases for $C = (C_1$ and $C_2)$ or C_3 to illustrate MC/DC coverage

	Inputs			Output
Test	C_1	C_2	C_3	C
Fix C_1 and C_2 to `true`, vary C_3				
1(9, 5)	`true`	`true`	`true`	`true`
2(11, 7)	`true`	`true`	`false`	`true`
Fix C_1 to `true`, C_2 to `false`, vary C_3[a]				
3(10, 9)	`true`	`false`	`true`	`true`
4(12, 11)	`true`	`false`	`false`	`false`
Fix C_1 to `false`, C_2 to `true`, vary C_3[a]				
5(1, 6)	`false`	`true`	`true`	`true`
6(3, 8)	`false`	`true`	`false`	`false`
Fix C_1 and C_2 to `false`, vary C_3[a]				
7(2, 10)	`false`	`false`	`true`	`true`
8(4, 12)	`false`	`false`	`false`	`false`
Fix C_1 and C_3 to `true`, vary C_2				
9(1, 5)	`true`	`true`	`true`	`true`
10(3, 9)	`true`	`false`	`true`	`true`
Fix C_1 to `true`, C_3 to `false`, vary C_2[a]				
11(2, 7)	`true`	`true`	`false`	`true`
12(4, 11)	`true`	`false`	`false`	`false`
Fix C_1 to `false`, C_3 to `true`, vary C_2				
13(5, 6)	`false`	`true`	`true`	`true`
14(7, 10)	`false`	`false`	`true`	`true`
Fix C_1 and C_3 to `false`, vary C_2				
15(6, 8)	`false`	`true`	`false`	`true`
16(8, 12)	`false`	`false`	`false`	`true`

(*continued*)

Table 6.6 *(Continued)*

Test	Inputs C_1	C_2	C_3	Output C
Fix C_2 and C_3 to `true`, vary C_1				
17(1, 9)	`true`	`true`	`true`	`true`
18(5, 1)	`false`	`true`	`true`	`true`
Fix C_2 to `true`, C_3 to `false`, vary C_1[a]				
19(2, 11)	`true`	`true`	`false`	`true`
20(6, 3)	`false`	`true`	`false`	`false`
Fix C_2 to `false`, C_3 to `true`, vary C_1				
21(3, 10)	`true`	`false`	`true`	`true`
22(7, 2)	`false`	`false`	`true`	`true`
Fix C_2 to `false`, C_3 to `false`, vary C_1				
23(4, 12)	`true`	`false`	`false`	`false`
24(8, 4)	`false`	`false`	`false`	`false`

[a]Corresponding tests affect C and may be included in the MC-DC adequate test set.
Identical rows are listed in parentheses.

only two tests are required for a simple condition. For example, to cover a simple condition $x < y$, where `x` and `y` are integers, one needs only two tests, one that causes the condition to be `true` and another that causes it to be `false`.

Next, we determine the MC/DC-adequate tests for compound conditions that contain two simple conditions. Table 6.9 lists adequate tests for

Table 6.7 MC-adequate tests for $C = (C_1 \text{ and } C_2) \text{ or } C_3$

Test	C_1	C_2	C_3	C	Effect demonstrated for
1[3]	`true`	`false`	`true`	`true`	
2[4]	`true`	`false`	`false`	`false`	C_3
3[11]	`true`	`true`	`false`	`true`	
4[12]	`true`	`false`	`false`	`false`	C_2
5[7]	`true`	`true`	`false`	`true`	
6[8]	`false`	`true`	`false`	`false`	C_1

[x], x is the corresponding test in Table 6.6.

Table 6.8 Minimal MC-adequate tests for $C = (C_1 \text{ and } C_2) \text{ or } C_3$

Test	C_1	C_2	C_3	C	Comments
t_1	`true`	`false`	`true`	`true`	Tests t_1 and t_2 cover C_3
t_2	`true`	`false`	`false`	`false`	Tests t_2 and t_3 cover C_2
t_3	`true`	`true`	`false`	`true`	Tests t_3 and t_4 cover C_3
t_4	`false`	`true`	`false`	`false`	

such compound conditions. Note that three tests are required to cover each condition using the MC/DC requirement. This number would be four if multiple condition coverage is required. It is instructive to carefully go through each of the three conditions listed in Table 6.9 and verify that indeed the tests given are independent (also try Exercise 6.13).

We now build Table 6.10, which is analogous to Table 6.9, for compound conditions that contain three simple conditions. Notice that only four tests are required to cover each compound condition listed in Table 6.10. One can generate the entries in Table 6.10 from Table 6.9 by using

Table 6.9 MC/DC-adequate tests for compound conditions that contain two simple conditions

Test	C_1	C_2	C	Comments
Condition: $C_a = (C_1$ `and` $C_2)$				
t_1	`true`	`true`	`true`	Tests t_1 and t_2 cover C_2
t_2	`true`	`false`	`false`	
t_3	`false`	`true`	`false`	Tests t_1 and t_3 cover C_1
MC/DC-adequate test set for $C_a = \{t_1, t_2, t_3\}$				
Condition: $C_b = (C_1$ `or` $C_2)$				
t_4	`false`	`true`	`true`	Tests t_4 and t_5 cover C_2
t_5	`false`	`false`	`false`	
t_6	`true`	`false`	`true`	Tests t_5 and t_6 cover C_1
MC/DC-adequate test set for $C_b = \{t_4, t_5, t_6\}$				
Condition: $C_c = (C_1$ `xor` $C_2)$				
t_7	`true`	`true`	`false`	Tests t_7 and t_8 cover C_2
t_8	`true`	`false`	`true`	
t_9	`false`	`false`	`false`	Tests t_8 and t_9 cover C_1
MC/DC-adequate test set for $C_c = \{t_7, t_8, t_9\}$				

Table 6.10 MC/DC-adequate tests for compound conditions that contain three simple conditions

Test	C_1	C_2	C_3	C	Comments
Condition: $C_a = (C_1$ `and` C_2 `and` $C_3)$					
t_1	`true`	`true`	`true`	`true`	Tests t_1 and t_2 cover C_3.
t_2	`true`	`true`	`false`	`false`	
t_3	`true`	`false`	`true`	`false`	Tests t_1 and t_3 cover C_2.
t_4	`false`	`true`	`true`	`false`	Tests t_1 and t_4 cover C_1.
MC/DC-adequate test set for $C_a = \{t_1, t_2, t_3, t_4\}$					
Condition: $C_b = (C_1$ `or` C_2 `or` $C_3)$					
t_5	`false`	`false`	`false`	`false`	Tests t_5 and t_6 cover C_3.
t_6	`false`	`false`	`true`	`true`	
t_7	`false`	`true`	`false`	`true`	Tests t_5 and t_7 cover C_2.
t_8	`true`	`false`	`false`	`true`	Tests t_5 and t_8 cover C_1.
MC/DC-adequate test set for $C_b = \{t_5, t_6, t_7, t_8\}$					
Condition: $C_c = (C_1$ `x or` C_2 `x or` $C_3)$					
t_9	`true`	`true`	`true`	`true`	Tests t_9 and t_{10} cover C_3.
t_{10}	`true`	`true`	`false`	`false`	
t_{11}	`true`	`false`	`true`	`false`	Tests t_9 and t_{11} cover C_2.
t_{12}	`false`	`true`	`true`	`false`	Tests t_3 and t_{12} cover C_1.
MC/DC-adequate test set for $C_c = \{t_9, t_{10}, t_{11}, t_{12}\}$					

the following procedure that works for a compound condition C that is a conjunct of three simple conditions, that is $C = (C_1$ `and` C_2 `and` $C_3)$.

1. Create a table with four columns and four rows. Label the columns as Test C_1, C_2, C_3, and C from left to right. The column labeled Test contains rows labeled by test case numbers t_1 through t_4. The remaining entries are empty. The last column labeled Comments is optional.

Test	C_1	C_2	C_3	C	Comments
t_1					
t_2					
t_3					
t_4					

2. Copy all entries in columns C_1, C_2, and C from Table 6.9 into columns C_2, C_3, and C of the empty table.

Test	C_1	C_2	C_3	C	Comments
t_1		`true`	`true`	`true`	
t_2		`true`	`false`	`false`	
t_3		`false`	`true`	`false`	
t_4					

3. Fill the first three rows in the column marked C_1 with `true` and the last row with `false`.

Test	C_1	C_2	C_3	C	Comments
t_1	`true`	`true`	`true`	`true`	
t_2	`true`	`true`	`false`	`false`	
t_3	`true`	`false`	`true`	`false`	
t_4	`false`				

4. Fill entries in the last row under columns labeled C_2, C_3, and C with `true`, `true`, and `false`, respectively.

Test	C_1	C_2	C_3	C	Comments
t_1	`true`	`true`	`true`	`true`	Tests t_1 and t_2 cover C_3
t_2	`true`	`true`	`false`	`false`	
t_3	`true`	`false`	`true`	`false`	Tests t_1 and t_3 cover C_2
t_4	`false`	`true`	`true`	`false`	Tests t_1 and t_4 cover C_1

5. We now have a table containing the MC/DC-adequate tests for $C = (C_1$ `and` C_2 `and` $C_3)$ derived from tests for $C = (C_1$ `and` $C_2)$.

The procedure illustrated above can be extended to derive tests for any compound condition using tests for a simpler compound condition (see Exercises 6.14 and 6.15). The important point to note here is that for any compound condition, the size of an MC/DC-adequate test set grows *linearly* in the number of simple conditions. Table 6.5 is reproduced as Table 6.11 with columns added to compare the minimum number of tests required, and the time to execute them, for the multiple condition and the MC/DC-coverage criteria.

Table 6.11 MC/DC adequacy and the growth in the least number of tests required for a condition with n simple conditions

	Minimum tests		Time to execute all tests	
n	MC	MC/DC	MC	MC/DC
1	2	2	2 ms	2 ms
4	16	5	16 ms	5 ms
8	256	9	256 ms	9 ms
16	65,536	17	65.5 s	17 ms
32	4,294,967,296	33	49.5 days	33 ms

6.2.10 Definition of MC/DC Coverage

We now provide a complete definition of the MC/DC coverage. A test set T for program P written to meet requirements R is considered adequate with respect to the MC/DC-coverage criterion if upon execution of P on each test in T, the following requirements are met:

1. Each block in P has been covered.
2. Each simple condition in P has taken both `true` and `false` values.
3. Each decision in P has taken all possible outcomes.
4. Each simple condition within a compound condition C in P has been shown to independently effect the outcome of C. *This is the MC part of the coverage we discussed in detail earlier in this section.*

The first three requirements above correspond to block coverage, condition coverage, and the decision coverage, respectively, and have been discussed in earlier sections. The fourth requirement corresponds to the MC coverage discussed earlier in this section.

Thus, the MC/DC-coverage criterion is a mix of four coverage criteria based on the flow of control. With regard to the second requirement, it is to be noted that conditions that are not part of a decision such as the one in the following statement

$$A = (p < q) \text{ or } (x > y)$$

are also included in the set of conditions to be covered. With regard to the fourth requirement, a condition such as (A and B) or (C and A) poses a problem. It is not possible to keep the first occurrence of A fixed while varying the value of its second occurrence. Here, the first occurrence of A is said to be coupled to its second occurrence. In such cases, an adequate test set need only demonstrate the independent effect of any one occurrence of the coupled condition.

A numerical value can also be associated with a test to determine the extent of its adequacy with respect to the MC/DC criterion. One way to do so is to treat separately each of the four requirements listed above for the MC/DC adequacy. Thus, four distinct coverage values can be associated with T namely the block coverage, the condition coverage, the decision coverage, and the MC coverage. The first three of these four have already been defined earlier. A definition of MC coverage is as follows:

Let $C_1, C_2, \ldots, C_N$ be the conditions in P; each condition could be simple or compound and may appear within the context of a decision. Let n_i denote the number of simple conditions in C_i , e_i the number of simple conditions shown to have independent effect on the outcome of C_i , and f_i the number of infeasible simple conditions in C_i . Note that a simple condition within a compound condition C is considered infeasible if it is impossible to show its independent effect on C while holding constant the remaining simple conditions in C. The MC coverage of T for program P subject to requirements R, denoted by MC_c, is computed as follows:

$$MC_c = \frac{\sum_{i=1}^{N} e_i}{\sum_{i=1}^{N} (e_i - f_i)}$$

Thus, test set T is considered adequate with respect to the MC coverage if MC of T is 1. Having now defined all components of the MC/DC coverage, we are ready for a complete definition of the adequacy criterion based on this coverage.

Modified condition/decision coverage

A test T written for program P subject to requirements R is considered adequate with respect to the MC/DC-coverage criterion if T is adequate with respect to block, decision, condition, and MC coverage.

The next example illustrates the process of generating MC/DC-adequate tests for a program that contains three decisions, one composed of a simple condition and the remaining two composed of compound conditions.

Example 6.23: Program P6.14 is written to meet the following requirements:

R_1: Given coordinate positions `x`, `y`, and `z`, and a direction value d, the program must invoke one of the three functions `fire-1`, `fire-2`, and `fire-3` as per conditions below: