NAWCWD TM 8915

Managing Unit Tests Using Vector Spaces

by
Anthony S. Cantone
Loren C. Edwards
System Safety Engineering Division
Systems Engineering Department

AUGUST 2021

NAVAL AIR WARFARE CENTER WEAPONS DIVISION
China Lake, CA 93555-6100

PREFACE

In the acquisition process, assurance of software integrity for embedded software is often an issue. This technical memorandum (TM) discusses an approach to designing unit tests for code of high complexity. Faulty or incomplete tests have also safety implications that may affect the acquisition process. Unit tests furnish assurance of error-free operation of software code units, enhancing safe operation of code-intensive systems. The underlying vector space of a code unit can be employed to construct a minimal set of unit tests with a basis set of such a vector space. This TM explores the use of basis sets to design a minimal set of unit tests and demonstrates how a basis set of tests is adequate for a logical test of a code unit containing a multitude of paths. The argument is made that this approach is equivalent to Modified Condition/Decision Coverage (MC/DC) coverage testing as defined in DO-178B/C.

This report was reviewed for technical accuracy by Loren Edwards.

K. R. CHIRKIS, *Head*
Systems Engineering Department
17 August 2021

NAWCWD TM 8915, published by Code DC12100,
3 paper, 4 electronic media.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED *(DD-MM-YYYY)*	
		START DATE	END DATE
17-08-2021	Final	22-07-2019	20-07-2021

4. TITLE AND SUBTITLE

Managing Unit Tests Using Vector Spaces

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
N/A	N/A	N/A
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
N/A	NA	N/A

6. AUTHOR(S) *(Last Name, First Name, MI)*

Cantone, Anthony S.; Edwards, Loren C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Air Warfare Center Weapons Division
1 Administration Circle
China Lake, California 93555-6100

8. PERFORMING ORGANIZATION REPORT NUMBER

NAWCWD TM 8915

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
None	N/A	N/A

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

None.

14. ABSTRACT

Unit tests furnish assurance of error free operation of software code units, enhancing safe operation of code-intensive systems. The underlying vector space of a code unit can be used to construct a minimal set of unit tests employing a basis set of such a vector space. This report explores the use of basis sets to design a minimal set of unit tests and demonstrates how a basis set of tests is adequate for a logical test of a code unit containing a multitude of paths. The argument is made that this approach is equivalent to Modified Condition/Decision Coverage (MC/DC) coverage testing as defined in DO-178B/C.

15. SUBJECT TERMS

McCabe, Unit Tests, Vector Spaces

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	c. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	36

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Anthony Cantone	(760) 939-1820

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

CONTENTS

Figures:

Tables:

ACKNOWLEDGMENTS

Thanks to Erica Arroyo, aerospace engineer (Code D572000) at the Naval Air Warfare Center Weapons Division (NAWCWD) China Lake, and Loren Edwards, system safety engineer at NAWCWD, for their editing support.

1.0 INTRODUCTION

In 1976, Thomas J. McCabe introduced the notion of Structured Testing (Watson, 1996 [Reference 1]). In this method, he formed an underlying vector space of a code module by creating vectors whose coordinates were edges in a directed graph derived from the flow diagram of a code module. The dimensions of these underlying vector spaces were related to the cyclomatic complexity of the code module, a measure that reflects ultimately the maintainability of the code. Vector spaces contain a subset of vectors known as a basis set, a set of vectors from which every vector in the vector space may be generated by forming a linear combination of vectors from the basis set, with coefficients from a field. McCabe concluded that a complete test of the code module could be accomplished by executing test paths that corresponded to the vectors in a basis set. He noted that a basis set of a vector space is not necessarily unique, but the number of elements in any basis set, i.e., the dimension of the vector space, is unique. This approach is equivalent to decision coverage testing as defined in DO-178B/C (Reference 2).

1.1 OBJECTIVE

In this technical memorandum (TM), the author and coauthor modify McCabe's approach by again forming a vector space underlying the code unit, but using as coordinates the **decisions** and **conditions** in the code unit instead of the edges in the flow diagram. (Note that the word "Unit" instead of "Module" will be used to represent the smallest compilable entity of code.) The motivation behind this modified approach is to enable Modified Condition/Decision Coverage (MC/DC) coverage as defined in DO-178B/C (Reference 2). The chapter **"What is a Vector Space"** repeats what is found in many books on linear algebra: the general definition of a vector space, its axioms that must be satisfied by the group operation, and the consequences of applying, as coefficients, elements of an associated field. The section **"Vector Space Over a Two-Element Field"** discusses the application of a finite field containing just the elements 0 and 1, which are used as coefficients in forming linear combinations of elements of a basis set. The meaning of basis sets is explored in the section **"Basis Sets."**

The underlying vector space of a code unit is discussed in the chapter **"Underlying Vector Space of a Code Unit;"** path vectors are discussed. The underlying vector space of a code unit and basis sets of the vector space are identified in the section **"Path Vectors."** The arithmetic of path vectors is presented in the section **"Algebra of Path Vectors."** Finally, the basis set of the underlying vector space in a code unit is identified in the section **"Basis Set of a Code Unit Vector Space."**

The application of a basis set to unit testing and its usefulness in minimizing unit tests are discussed in the chapter **"Unit Tests."** The section **"Example"** depicts a code unit with two **simple decisions** and with a single entrance and exit. This simple example will assist in demonstrating that the set of all possible paths through the code unit is a vector space.

Real-world code units contain **compound decisions** that are usually combinations of conditions connected by Boolean operators.

The chapter **"Compound Decisions"** provides a simplified method to determine a basis set by **deconstructing** compound decisions into single conditions. The section **"How to Choose a Basis Set"** gives the reader a technique for determining a basis set by ensuring that each decision in the code unit is exercised in its true and false directions. The basis set should contain exactly the number of vectors equal to the dimension of the vector space.

In the event that a condition appears in more than one decision, the dimension of the underlying vector space may be less than the number of decisions in the code unit because of the dependencies between decisions. Nevertheless, the two occurrences of the condition will be treated as though they are different conditions, thus allowing the code to change the value of a condition between usages.

An efficient procedure for determining a basis set without deconstructing each compound decision is discussed in section **"Compound Decisions Without Deconstruction."** The short section **"Multi-exit Decisions"** describes an approach for handling case or switch statements. This chapter also contains a comprehensive **"Example." "What is Expected in Practice"** is the final section of this chapter. It describes the shortcuts that the author has taken while employing the methods in this TM. Words in bold font are defined in the glossary.

1.2 SCOPE

This TM is designed to investigate a method of defining the underlying vector space in a code unit that makes possible MC/DC testing as defined in DO-178B/C. This is an alternate definition to McCabe's definition in his structured testing methodology. In order not to go beyond the scope of this TM, a loop in the code unit is represented as a loop with a single iteration: the path enters and exits the loop, or it does not. Even though the logic is tested, the iteration limit of the loop is not tested.

2.0 WHAT IS A VECTOR SPACE?

The following definitions can be found in many books and websites on linear algebra.

A **vector space** *V* is a non-empty set of objects called **vectors**, along with two binary **operations** + and ∗, and a field *F* of coefficients for *V*. The "+" operation is between two vectors and yields a third vector, an object in *V*.

For example,

$$u = v + w \tag{2-1}$$

where u, v, and w are vectors in *V*. The "+" operation must of course be defined if the vector space is to be useful.

The "∗" operation is between a scalar and a vector. The scalar belongs to a field *F*. So

$$u = a * v \tag{2-2}$$

where u and v are vectors in the space *V*, and a (**scalar**) is in the **field** *F*. Sometimes this is written as

$$u = av \tag{2-3}$$

The "∗" operation is understood.

If V is a vector space and u, v, and w are vectors in V, there are several **axioms** that must be satisfied. These are

1. + is **associative**. This means $u + (v + w) = (u + v) + w$.
2. + is **commutative**. This means $u + v = v + u$.
3. There is a vector 0 such that $v + 0 = v$. 0 is the **identity** under +. 0 is sometimes called the zero vector.
4. For every vector v in the vector space *V*, there is a vector *w* such that $v + w = 0$. The vector w is sometimes written as $-v$. This vector is called the **inverse** of v with respect to +.
5. There is a binary operation ∗ between elements of *F* and vectors in *V*, which **distributes** over + in *V*. That is, $a * (u + v) = a * u + a * v$, where u and v are in V, and a is in *F*. This can also be written $a(u + v) = au + av$ where ∗ is understood.

6. There is an additive operation + in *F* which distributes over $*$, the operation between elements of *F* and vectors in *V*. That is, $(a + b) * v = a * v + b * v$ sometimes written as $(a + b)v = av + bv$, where a, b are in *F*, the first + is the additive operation in *F*, and the second + is the operation between vectors in the vector space *V*. + can be used for both operations because there is no danger of confusion due to context clues.
7. $(ab)v$ and $a(bv)$ are equal, where a, b are in *F*, v is in V, and bv is in *V*. The operation implied between a and b in the expression ab is the multiplicative operation in *F*.
8. The additive identity 0 in *F* obeys the law $0v = 0$ for every v in *V*, where 0 to the right of the equal sign is the zero vector in *V*.
9. The multiplicative identity 1 in *F* obeys the law $1v = v$ for every v in *V*.

2.1 VECTOR SPACE OVER A TWO-ELEMENT FIELD

Unlike the approach in Watson, 1996 (Reference 1), where the vector components are edges traversed in a directed graph, each vector component in the method used in this TM is the value of a decision encountered. For example, suppose there are three decisions encountered in a **path**, beginning at a unique entrance to the unit and ending at a unique exit of the unit. Then a typical vector could be the following: (0, 1, 1), where 0 represents False and 1 represents True. This vector would represent a path in Figure 2-1 that would traverse the first decision (indicated by "1") using FALSE, the second decision (to the right of the first decision, indicated by "2") using TRUE, and the third decision using TRUE. Note that if a path does not traverse a particular decision, its entry in the vector representing that path will contain a don't-care. Linear combinations of these vectors need coefficients from a field, but only 1's and 0's need be used. Therefore, the field does not need more than those two elements, 0 and 1. Moreover, since the values of the coordinates in each vector carry meaning of "TRUE" and "FALSE," the 0 or 1 value in the vector coordinates can be regarded as coming from the two-element field.

This approach neglects the possibility of a loop being traversed more than once. To keep within the scope of this TM, loops traversed more than once will not be considered. The limit of the loop index will need to be checked with an independent white box test.

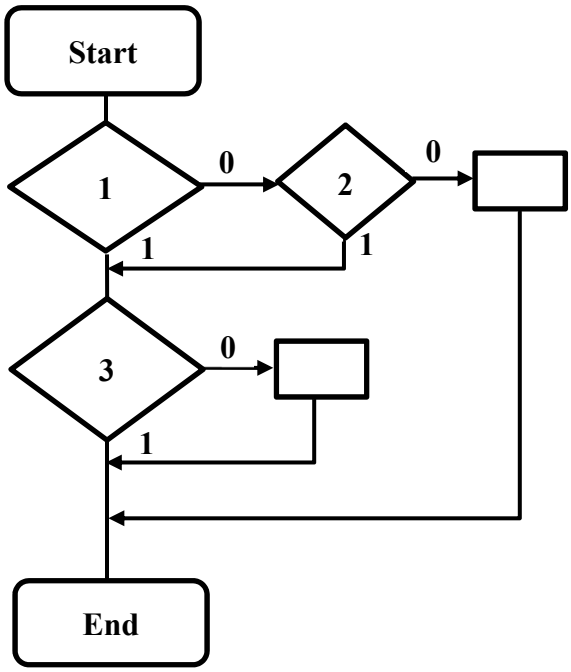

FIGURE 2-1. Defining a Vector in a Code Unit With Three Decisions.

2.2 BASIS SETS

Every vector space contains a basis set (Keremedis, 1996 [Reference 3]). That is, every vector space contains a subset of vectors, called a basis set, such that every vector in the space can be written as a linear combination of vectors from the basis set. Basis sets are not necessarily unique.

Figure 2-2 illustrates a real vector space of dimension 3 (R^3). The vectors in red represent typical vectors, while the vectors in blue are one choice for a basis set. Typically, this vector space uses unit vectors, that is, vectors of one unit long, for basis vectors.

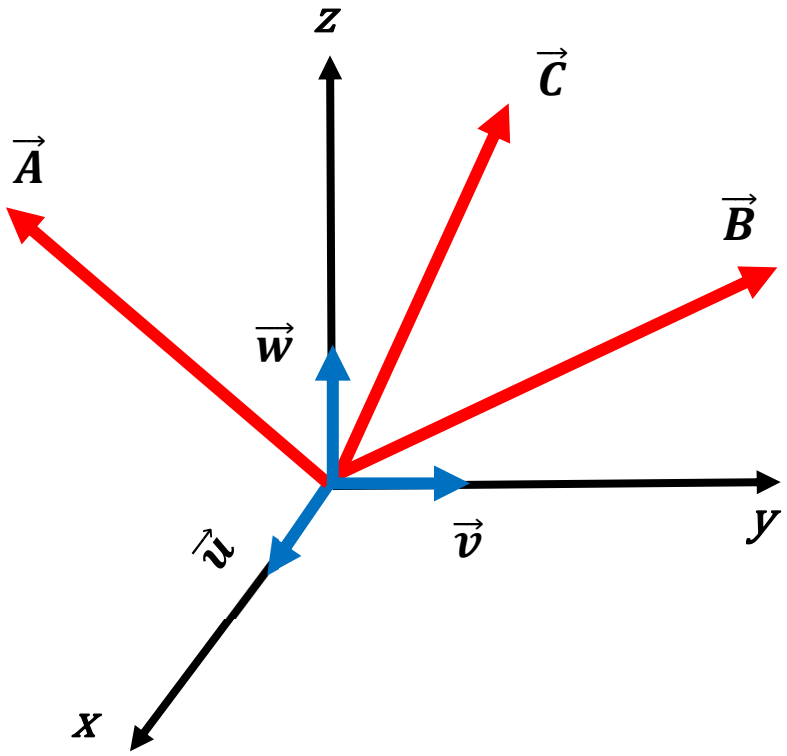

FIGURE 2-2. Three-Dimensional Vector Space.

2.3 USE OF COORDINATES

In the remainder of this TM, coordinates associated with the vectors will be used instead of diagrams using vectors. See the definition of "Coordinate" in the glossary.

3.0 UNDERLYING VECTOR SPACE OF A CODE UNIT

This section discusses the vector space associated with a code unit. The decisions in the code unit are assumed to be simple decisions, i.e., decisions containing just one condition. It is assumed that there is a single entrance into and a single exit out of the code unit. This facilitates the construction of the underlying vector space.

Path vectors are also discussed in this section. The underlying vector space of a code unit is derived. A basis set of the underlying vector space is identified.

3.1 PATH VECTORS

Each path through the code unit, from the unique start to the unique end of the code unit, will correspond to a coordinate vector. The vector will have as many coordinates as there are decisions in the code unit. This number will also be the dimension of the vector space, and the number of vectors in the basis set.

3.2 ALGEBRA OF PATH VECTORS

In a vector space, there is always a binary operation "+" that must be defined. For vectors in a plane, the "+" operation is defined using a parallelogram as an aid. With coordinate vectors, a rule governing how corresponding coordinates are to be combined under the "+" operation needs to be formulated. Refer to Section 4.1 (paragraph under Figure 4-1) for this definition.

3.3 BASIS SET OF A CODE UNIT VECTOR SPACE

The vectors in a basis set have a special significance. Linear combinations of these vectors form other vectors in the vector space that are not in the basis set. Basis sets in general have two properties, one being that by means of linear combinations the basis set **spans** the entire vector space. That is, every vector in the vector space can be written as a linear combination of vectors in the basis set. The second property is that the vectors in the basis set are **linearly independent**. The number of vectors in the basis set is equal to the dimension of the vector space. A collection of vectors containing more vectors than the dimension of the space is always linearly dependent. A collection of vectors containing fewer vectors than the dimension of the space is never a basis set since there will always exist a vector in the space that cannot be expressed as a linear combination of vectors in that collection.

An example of a test for linear independence is given in the first example Section 4.0.

4.0 UNIT TESTS

Unit tests are a programmer's assurance that the function being coded is correct. The unit tests are written against the immediate low-level requirements from which the code unit is developed; however, the programmer's objective is to ensure that every path in the code unit behaves as expected. In code units with many decisions, the number of possible paths can be very large. The test set forcing execution down all the paths is daunting. So a means of limiting the number of tests without sacrificing thoroughness is welcomed. Examining the underlying vector space helps in this direction.

Since every vector in the underlying vector space can be written as a linear combination of vectors in the basis set, every path in the code unit can be written as a linear combination of paths in the basis set of paths. A test set that forces execution along the paths in the basis set is thought to be an adequate logical test of the entire code unit. This is reinforced by the fact that every decision is toggled both ways. Basis sets typically have 10 through 15 vectors. Therefore, for a code unit of perhaps 10 or more decisions where the possible paths through that code unit may number in the thousands, a test of just the basis set paths may be a large savings in time and money.

4.1 EXAMPLE 1

What follows is a very simple example. This example is designed to show a code unit with two simple decisions: a single entrance to the unit and a single exit from the unit. The example will show all possible paths through the unit and will show the underlying vector space. A vector space basis set will be specified and its linear independence will be demonstrated. It will be demonstrated that every vector in the vector space (i.e., every path through the code) is a linear combination of vectors (i.e., paths) in the basis set. Many definitions will be repeated for clarity.

Consider Figure 4-1, a flow diagram of a **structured code** unit.

Coordinate vectors are **ordered pairs**, for example, (0, 1). This ordered pair represents a path through the code, from start to end. The first number in the ordered pair represents the value of the first decision. The second number represents the value of the second decision. Then the red line in Figure 4-1 is the path that (0, 1) represents. Recall from Section 2.1 that we are using the values in the field F as the coordinate values in the vectors.

What are all the possible paths through this code unit? There are only four possible paths. The vectors for these paths are (0, 0), (0, 1), (1, 0), and (1, 1).

Let us designate the letter V as the set containing these four vectors. If it can be shown that V satisfies the axioms for a vector space, then V is the underlying vector space for the code unit.

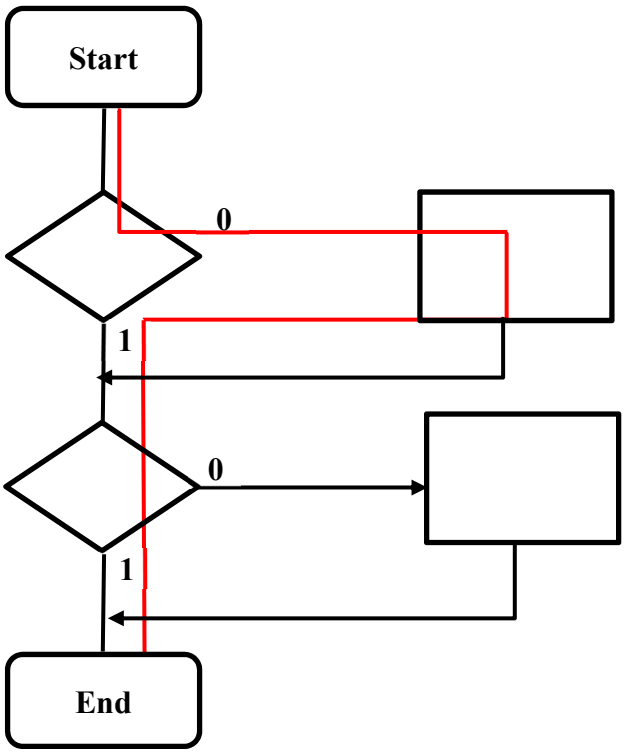

FIGURE 4-1. Flow Diagram of Code Unit.

We have to decide whether these four vectors in *V* form a vector space. But first, we have to create the operation "+". What does it mean to add two vectors? For example, what does (0, 1) + (1, 1) mean? The operation "+" must produce a vector that is one of the possible paths through this unit.

The best approach for defining the meaning of the binary operation "+" between vectors is defining this operation in terms of the field operations "+" and "∗".

For 0 = False and 1 = True, we can use the XOR operation to define "+" in the field. Table 4-1 defines this operation in the field *F*.

TABLE 4-1. The "+" Operation in *F*.

+	**0**	**1**
0	0	1
1	1	0

Therefore, the meaning of $(m_1, n_1) + (m_2, n_2)$ is $(m_1 + m_2), (n_1 + n_2)$, where the + in the sum vector is defined in Table 4-1 and the elements (m_1, n_1) and (m_2, n_2) are in *F*.

We can use this to form the table that defines all possible sums of the vectors in *V*. See Table 4-2.

TABLE 4-2. Addition of Vectors.

+	**(0, 0)**	**(0, 1)**	**(1, 0)**	**(1, 1)**
(0, 0)	(0, 0)	(0, 1)	(1, 0)	(1, 1)
(0, 1)	(0, 1)	(0, 0)	(1, 1)	(1, 0)
(1, 0)	(1, 0)	(1, 1)	(0, 0)	(0, 1)
(1, 1)	(1, 1)	(1, 0)	(0, 1)	(0, 0)

We see that the operation "+" is closed, that is, always produces a vector that is in the original set of vectors. However, does this definition of "+" satisfy the axioms of a vector space?

Fortunately, the table of all possible sums is small, so it is easy to check that "+" is **associative**. An example of associativity is the following:

$$(1, 0) + ((1, 1) + (0, 1)) = ((1, 0) + (1, 1)) + (0, 1) \qquad (4\text{-}1)$$

The expression on each side of the equal sign is equal to (0, 0). Thus associativity is satisfied for the three vectors (1, 0), (1, 1) and (0, 1), since the equality is true. It is easy to check that associativity is true for any three vectors.

Commutativity is even easier. Is $(m_1, n_1) + (m_2, n_2) = (m_2, n_2) + (m_1, n_1)$, where m_1, n_1, m_2, and n_2 are numbers 0 and 1, corresponding respectively to False, True, for all possible pairs of vectors? If one notices the symmetry of Table 4-2 about the diagonal from upper left to lower right, that is all that is needed.

Is there a **zero vector** or an **identity**? Yes, (0, 0) satisfies that need, since for any vector $(m, n), (m, n) + (0, 0) = (m, n)$ from Table 4-2. Every vector has an inverse with respect to (0, 0). In fact, every vector is its own inverse. All this is obvious in Table 4-2. For example, what is the inverse of (1, 1)? In other words, what adds to (1, 1) to get the zero vector (0, 0)? Answer from Table 4-2: (1, 1).

The "+" binary operation can be considered analogous to the additive operation in the fields with which mathematicians commonly work. What about a binary operation analogous to the multiplicative operation in the field? We can call it "$*$". For this operation, we will use the logical AND. The multiplication table is Table 4-3.

TABLE 4-3. The "$*$" Operation in *F*.

$*$	**0**	**1**
0	0	0
1	0	1

Define another "$*$" between elements of *F* and *V*. Does "$*$" **distribute** over "+" in *V* i.e., $a * (u + v) = a * u + a * v$? One way to show that the two expressions are equivalent is by showing that their values, as determined in Table 4-4, are identical for the same values of a. The columns under $*$ on the left and + on the right are the values of each of the expressions.

TABLE 4-4. "$*$" Between *F* and *V* Distributes Over "+" in *V*.

a	$*$	$(u+v)$	a	$*$	u	$+$	a	$*$	v
0	0	$u+v$	0	0	u	0	0	0	v
1	$u+v$	$u+v$	1	u	u	$u+v$	1	v	v

We will leave it to the reader to demonstrate that "+" in *F* distributes over "$*$" between *F* and *V* (that is, $(a + b) * v = a * v + b * v$) and to check that there are scalars 0 and 1 in the field such that $0v = 0$ and $1v = v$ for all v in *V*. However, 0 and 1 are the only elements in *F*.

Note that the "$*$" operation between elements in *F* will be used in the following manner: the coordinates of vectors in *V* will be expressed as elements of *F*. For expression such as $a * v = a * (m, n)$, where $a \in$ is an element of *F*, v is an element of *V*; m and n are elements of *F* will be $(a * m)(a * n)$. For the term $a * v$, if v is in the vector space, then so is $a * v$. a is either 0 or 1, $0 * v$ is the 0 vector, and $1 * v$ is v.

"$*$" distributes over "+" in the vector space as in Equation 4-2.

$$a * \big((m_1, n_1) + (m_2, m_2)\big) \;=\; a * (m_1, n_1) \;+\; a * (m_2, n_2), \tag{4-2}$$

a and b from *F*, and "$*$" is multiplication between elements *F* and *V* as in Equation 4-3.

$$(a + b) * \;(m, n) \;=\; a * \;(m, n) \;+\; b * (m, n)\,, \tag{4-3}$$

where a and b are in *F*, and "$*$" is multiplication between elements in *F* and *V*. Note that the operation "+" is used in two ways, as the operation between vectors and the operation between elements of *F*. There should be no confusion because of context clues. Similarly with "$*$".

Table 4-5 presents a summary of the operations defined previously.

TABLE 4-5. Binary Operations Defined in This TM.

Binary Operation	Description
+	Binary operation in the field *F*, defined as XOR. Reference Table 4-1.
*	Binary operation in the field *F,* defined as AND. Reference Table 4-3.
+	Binary operation between vectors in *V*, defined on the coordinates in *V* in terms of the operations of + in the field *F*, e.g., $(m_1, n_1) + (m_2, n_2) = (m_1 + m_2), (n_1 + n_2)$
*	Binary operation between elements of *F* and vectors in *V*. Defined as follows: $a * (m, n) = (a * m)(a * n)$
+	Binary operation between conditions, defined as inclusive "OR". Reference Table 5-1.
*	Binary operation between conditions, defined as "AND". Reference Table 5-2.

In the Equations 4-2 and 4-3, the "$*$" operation is usually dropped, as in $a(m, n)$ instead of $a * (m, n)$, or av instead of $a * v$, the "$*$" operation being understood.

What is a **basis set** for *V*? This is a subset of *V* such that every vector in *V* can be expressed as a linear combination of elements in this subset. Expressed in another way, every path in the code unit is a linear combination of paths in the basis set of paths.

For example, suppose the vectors (1, 0), (0, 1) form a basis set. Then the vector (1, 1) can be written as the linear combination 1(1, 0) + 1(0, 1), where the coefficient 1 is an element of the field *F*.

Finally, $a(bv)$ is the same as $(ab)v$.

With these definitions, *F* is a field, used for coefficients of the vectors in the vector space.

What is the **dimension** of *V*? This is the number of coordinates in the vector $v = (v_1, v_2, \cdots, v_n)$. The dimension is 2 for this example. Why is this important? This is important because the dimension of the vector space is also the number of vectors in the basis set of *V*.

Can we choose a basis set? This should be easy. There are two requirements for a basis set: that the basis set should span the entire space (i.e., that every vector in the space is a linear combination of vectors in the basis set), and that the vectors in the basis set be linearly independent.

This first requirement means that if m_i, for i = 1 to n, are the vectors in a candidate basis set containing n vectors, then the equation,

$$\sum_1^n a_i m_i = v \tag{4-4}$$

where v is any arbitrary vector in V, and the a_i (from F) are the coefficients of the vectors in the candidate basis set, will have a solution for the a_i in terms of m_{11}, m_{12}, …, m_{1n}, m_{21}, m_{22}, …, m_{2n}, …, m_{n1}, m_{n2}, …, m_{nn}.

This can be seen by replacing Equation 4-4 with its equivalent matrix Equation 4-5.

$$\begin{bmatrix} m_{11} & \dots & m_{1n} \\ \dots & \dots & \dots \\ m_{n1} & \dots & m_{nn} \end{bmatrix} \begin{bmatrix} a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} v_1 \\ \dots \\ v_n \end{bmatrix} \tag{4-5}$$

Briefly, Equation 4-5 states that any vector in the vector space can be expressed as a linear combination of vectors in the candidate basis set with coefficients in the field F. The goal is the solve for the coefficients a_1, a_2, …, a_n of the vectors in the basis set. Solving for the vector a requires that the m matrix be invertible. That is exactly the requirement that the vectors in the candidate basis set span the vector space V. That is, the linear combination of vectors in the candidate basis set that expresses the arbitrary vector v in the vector space has been found.

This last requirement means that if m_i, for i = 1 to n, are the vectors in a basis set containing n vectors (for vector space of dimension n), then the equation $\sum_1^n a_i m_i = 0$, where 0 is the zero vector, implies that all the $a_i = 0$. If there are any other solutions for the a_i, then the set of vectors is not linearly independent, hence does not form a basis set.

Returning to the underlying two-dimensional vector space of Figure 4-1, recall that the four vectors (0, 0), (0, 1), (1, 0), and (1, 1) represent the four paths. What could be the space unit for this vector space? Let us choose the two vectors (1, 0) and (0, 1). Is this a basis set? Do they span the space? The four vectors in the space can be written as linear specifications of these vectors as follows

$$\begin{aligned} (0, 0) &= 0(1, 0) + 0(0, 1) \\ (0, 1) &= 0(1, 0) + 1(0, 1) \\ (1, 0) &= 1(1, 0) + 0(0, 1) \\ (1, 1) &= 1(1, 0) + 1(0, 1) \end{aligned} \tag{4-6}$$

Formally, verification that (1, 0) and (0, 1) span the vector space is equivalent to showing that the matrix $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ is not singular. We see that from the solution of Equation 4-7 which is equivalent to the requirement that (1, 0) and (0, 1) be linearly independent:

$$a(1,0) + b(0,1) = (0,0) \tag{4-7}$$

where a and b are from the field *F*.

This implies the matrix equation

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} = \begin{bmatrix}0\\ 0\end{bmatrix} \tag{4-8}$$

The square matrix is the unit matrix, which has an inverse, and so the equation can be solved for a and b: $a = b = 0$. That is the only solution, so the vectors are linearly independent. Therefore, a basis set for the vector space has been found.

This means that every path in the code unit is a linear combination of paths represented by the vectors in the basis set. Moreover, each decision is toggled both ways. Thus, a complete logical test of the code unit can be accomplished with just two instead of four tests. This is why this test is regarded as a complete logical test. In this example, MC/DC is also satisfied. This is true because not only is each decision in the code unit toggled both ways but the MC/DC properties of holding all decisions constant except the one that is toggled is satisfied trivially. However, this is not always the case. It may be necessary in particular cases to supplement the basis set of paths with other vectors that toggle those decisions than has been exercised in just a singular direction (true or false) in order to be sure that each decision in the code unit is toggled in both directions. However, if one recalls the definition of MC/DC test, it is not enough to add vectors that complete the toggling of each decision, but they do so while holding other decisions constant.

Figure 4-1 is a simple example. In a code unit with 10 decisions, there may be as many as 1,024 possible paths, hence 1,024 vectors in the underlying vector space. If a basis set for this vector space can be found, then a set of 10 tests can do the work of testing 1,024 paths.

5.0 COMPOUND DECISIONS

The above development assumes decisions based on a single condition. In reality, code generally contains **compound decisions** based on several conditions connected by Boolean operators. The above development applies if each compound decision is deconstructed into decisions containing only a single **condition**.

DO-178B/C (Reference 2) is a set of guidelines created by the Radio Technical Commission for Aeronautics (RTCA) concerning commercial safety of flight. There are three levels of code coverage depending on the severity of consequence in the event of software failure. The most severe case requires MC/DC. A less severe case requires only decision coverage, and the least severe case, statement coverage. The tests written using the path vectors as defined so far in this report perform decision coverage. If it can be shown that whenever a condition is toggled while all other conditions remain constant, the result of the decision is toggled, then the coverage MC/DC is established. Otherwise, McCabe's definition of path vectors leads to tests that fall between Decision Coverage and MC/DC as defined in DO-178B/C.

If compound decisions are deconstructed into simple decisions, then the conditions are tested directly. That is, vectors are created for each of the conditions.

Consider the following example (Figure 5-1) of a code unit with three decisions, two of which are compound decisions. That is, the compound decisions consist of two conditions connected with logical operators + (inclusive OR) AND $*$ or juxtaposition, where these operations are defined in Tables 5-1 and 5-2, respectively (reference Table 4-5).

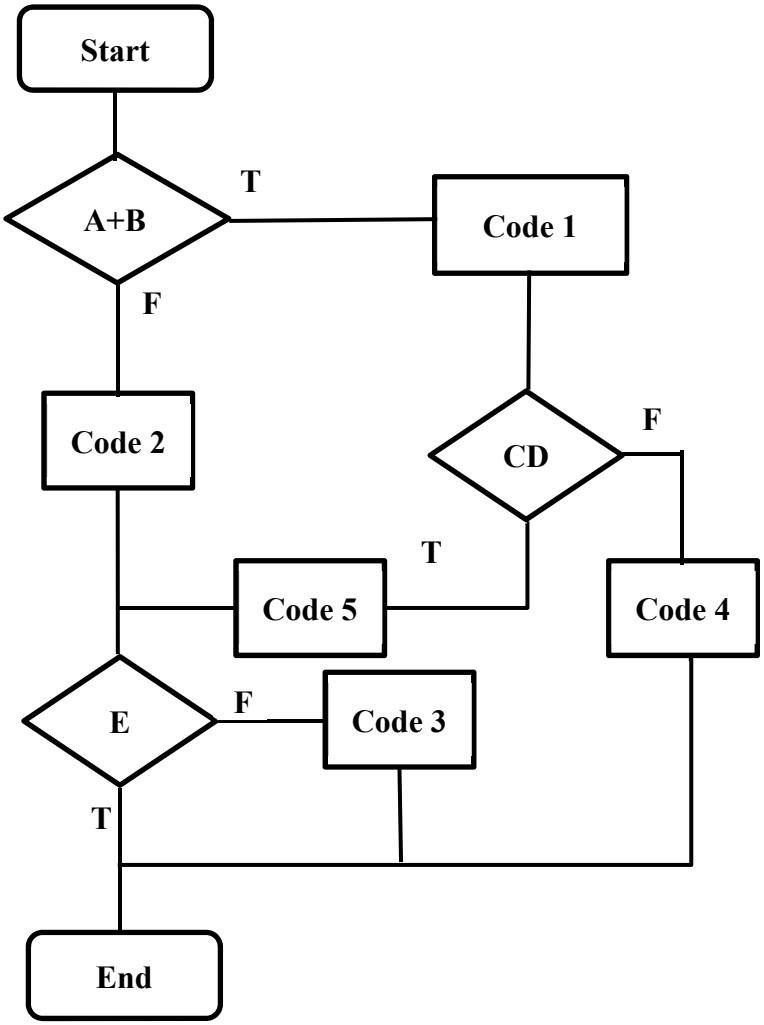

FIGURE 5-1. Flow Diagram of Code Unit With Compound Decisions.

TABLE 5-1. The "+" Operation for Conditions.

+	**0**	**1**
0	0	1
1	1	1

TABLE 5-2. The "∗" Operation for Conditions.

∗	**0**	**1**
0	0	0
1	0	1

Figure 5-2 is a revised version of Figure 5-1 with each compound decision deconstructed, so that each decision contains just one condition with the logic remaining the same. The next step is to conclude that there are 2^5 vectors in the underlying vector space of a code unit containing five simple decisions.

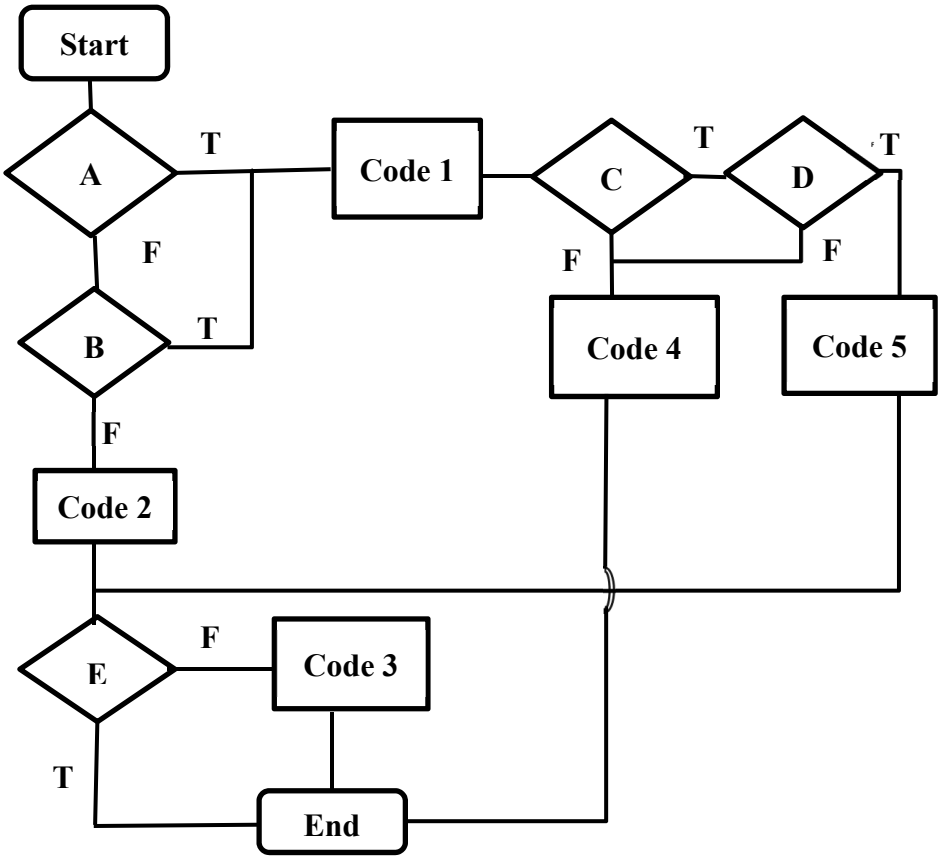

FIGURE 5-2. Flow Diagram From Figure 5-1 of Code Unit
With Deconstructed Decisions.

Table 5-3 provides a list of the vectors based on Figure 5-2. The vectors in the underlying vector space each contain five coordinates. The first coordinate corresponds to decision A, the second to decision B, etc. Each coordinate has the value 1 for True, 0 for False, and x for don't–care; x is used when a decision is skipped and does not affect the path.

TABLE 5-3. List of Vectors in Underlying Vector Space of Code Unit in Figure 5-2.

Vectors in Vector Space	Number of Vectors
(0, 0, x, x, 1)	4
(0, 0, x, x, 0)	4
(1, x, 1, 1, 1)	2
(1, x, 1, 1, 0)	2
(1, x, 1, 0, 1)	2
(1, x, 1, 0, 0)	2
(1, x, 0, x, x)	8
(0, 1, 1, 1, 1)	1
(0, 1, 1, 1, 0)	1
(0, 1, 1, 0, 1)	1
(0, 1, 1, 0, 0)	1
(0, 1, 0, x, x)	4

Considering that there are five conditions, there are $2^5 = 32$ vectors in this vector space as was expected. Recall that the value for x is either 0 or 1. Therefore, a vector such as (1, x, 1, 1, 1) is a concise method for representing the two vectors (1, 0, 1, 1, 1) and (1, 1, 1, 1, 1).

The verification that these vectors form a vector space follows from the explanation in the example preceding Figure 4-1. The "+" operation between vectors is defined in the same way, and the coefficients from the field *F* are the same, 0 and 1. The next step is to choose a candidate basis set (see Table 5-4) and show that this set of vectors is linearly independent. The method for choosing a basis set will be discussed in the section "How to Choose a Basis Set."

TABLE 5-4. Candidate Basis Set for Underlying Vector Space of Figure 4-1.

Vector Name	Vector
B_1	(1, 1, 0, 1, 1)
B_2	(0, 1, 1, 1, 1)
B_3	(0, 1, 1, 0, 1)
B_4	(0, 0, 1, 1, 1)
B_5	(0, 0, 1, 1, 0)

How do we show these vectors are linearly independent? We construct a linear combination with coefficients a_1, a_2, a_3, a_4, and a_5 from the field F and equate to the 0 vector, which is expressed in Equation 5-1. Then we solve for the a_i coefficients, noting that a solution is possible and that the only solution is $a_i = 0$ for $i = 1 \cdots 5$. That is sufficient to show linear independence. This is a necessary step, since if the n vectors in the basis set were linearly dependent, then there would not be n independent vectors in the basis set and not all vectors in the vector space would be expressible as a linear combination of vectors from the basis set.

$$\sum_1^5 a_i B_i = 0 \tag{5-1}$$

Considering the components of each of the B_i, this equation can be written in matrix form:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5-2}$$

The determinant of the coefficient matrix being 1, the coefficient matrix is non-singular. Therefore, a solution for the a_i exists. Multiplying on the left by the inverse of the coefficient matrix, we obtain Equation 5-3.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5-3}$$

This equation shows that the only solutions for the a_i are 0.

If the 0 vector in Equation 5-2 is replaced by a general vector in the underlying vector space, Equation 5-4 is reached.

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} \tag{5-4}$$

The resulting equation is also seen to have a solution for the a vector (coefficients of the basis set) as functions of the coordinates of the general vector in the underlying vector space, since the coefficient matrix is not singular. This satisfies the requirement that the candidate basis set spans the vector space.

5.1 HOW TO CHOOSE A BASIS SET

A candidate basis set is chosen by selecting paths each of which traverses conditions or decisions not traversed by any other path. The candidate basis set should toggle each of the conditions in the code unit both ways (True/False).

In Figure 5-3, the vector (1, x, 0, x, x) is chosen as a candidate vector in the basis set. Considering the don't-cares, this vector actually represents eight vectors. Arbitrarily, we choose (1, 1, 0, 1, 1). This is the first vector in Table 5-4. The remaining four vectors in Table 5-4 are chosen in a similar fashion: (0, 1, 1, 1, 1); (0, 1, 1, 0, x) → (0, 1, 1, 0, 1); (0, 0, x, x, 1) →(0, 0, 1, 1, 1); (0, 0, x, x, 0) →(0, 0, 1, 1, 0). Each of these choices traverses at least one condition True/False that was not traversed by any previous choice.

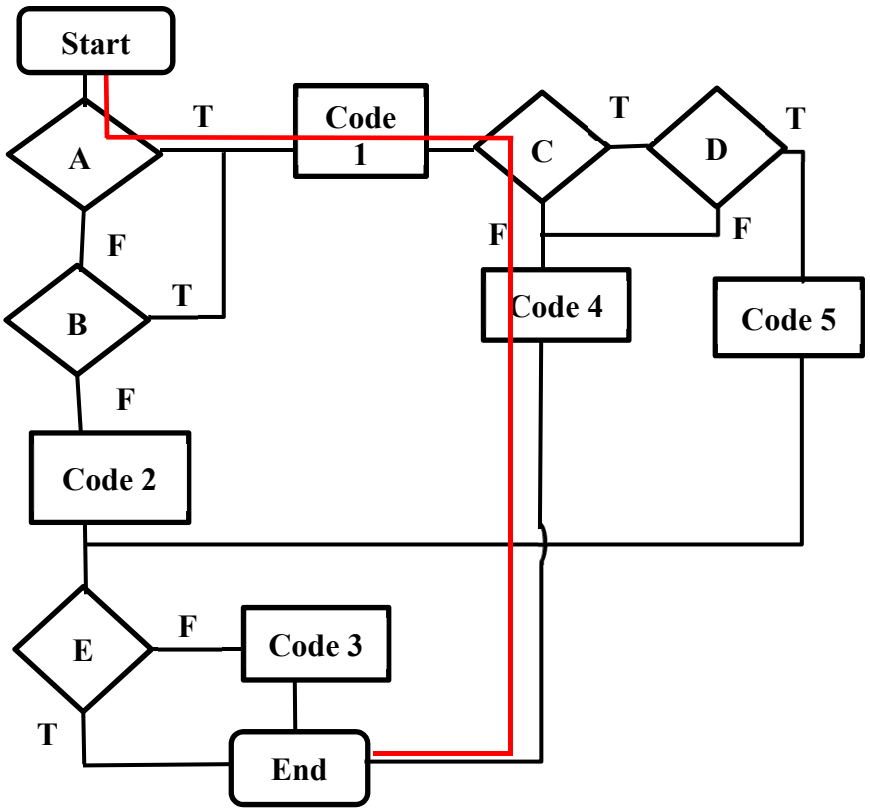

FIGURE 5-3. Vector (1, x, 0, x, x) in Underlying Basis Set.

The existence of don't-cares indicates that there are decisions in the code unit that are not triggered in some of the paths. The don't-cares in these vectors can be ignored.

5.2 COMPOUND DECISIONS WITHOUT DECONSTRUCTION

Decisions with more than one condition are common. These are what are called compound decisions. A condition is an entity that has a Boolean value. One could go through the process of deconstructing each compound decision into several decisions, each containing just one condition, but much more efficient is a process where such deconstruction is unnecessary. The following steps are designed to do just that. Consult Figure 5-1.

Procedure:

1. Determine the dimension of the underlying vector space. That is, determine the number of independent conditions in all of the decisions in the code unit. The dimension tells us two things: (1) the number of vectors in a basis set, and (2) the maximum number of vectors in the underlying vector space.

2. Construct a candidate basis set for this vector space. Determine that all the decisions are exercised both ways. In the case of compound decisions, the logic in each of the decisions must be considered to determine that the basis path vector actually exits the decision in the direction indicated by the values of the coordinates corresponding to the remainder of the coordinates in the basis path vector. For example, let us consider if there are vectors in the basis set that test condition B in Figure 5-1. Table 5-4 has the basis set vectors, where the don't-cares have been chosen as 1 (True). Are there vectors in this table that exercise condition B for B = True and B = False? We looked for vector(s) in the basis set that indicate(s) that B takes on both True and False values. We found (0, 0, 1, 1, 1) and (0, 1, 1, 1, 1).

3. Show that this set is also linearly independent. Consult Equations 5-1 and 5-2 for the method.

4. Show that this candidate basis set constructed in step 2 covers the vector space. This should be trivial if step 3 is done. For example:

    a. If in step 3 linear independence is shown for a set of vectors numbering the same as the dimension of the vector space, this is possible only if all the vectors in the space can be expressed as linear combinations of the vectors in the candidate basis set.

    b. On the other hand, if a linearly independent set has not been found, then there is a vector missing from the candidate basis set. This means that one or more vectors in the space cannot be expressed as a linear combination of vectors from the candidate basis set. Then the candidate basis set does not span the space. If coverage is achieved, then we have a basis set.

5. If MC/DC coverage is desired, check that for the two vectors that toggle a particular condition, other conditions in that decision remain fixed. Otherwise, adjust the basis set so that this is true.

6. Construct a test for each of the vectors in the basis set. That is, for each of the vectors in the basis set, find a set of inputs that drives the execution path along a path represented by that vector. Hopefully, we will find a test for B = True and a test for B = False.

7. Check with the low-level requirements that drive the design of the code unit that the result of the test is correct.

5.3 MULTI-EXIT DECISIONS

Decisions that have multiple exits, such as case or switch statements, can be deconstructed. That is, code can be written that would artificially create a decision for each of the multiple exits. However, this is the same as creating a vector in the underlying vector space for each of the exits. For each of the exits, the corresponding decision will be yes = 1, that particular exit was taken, or no = 0, it was not.

5.4 EXAMPLE 2

Example 2 (see Figure 5-4) is a modified version of Figure 5-1 with an added condition G to the flow diagram. Example 2 will demonstrate that there are 2^6 vectors in the underlying vector space of a code unit containing six conditions. Table 5-5 provides a list of the vectors in the vector space for Figure 5-4. The underlying vector space contains six coordinates with the first coordinate corresponding to condition G, the second to condition A, third to condition B, fourth to condition C, fifth to condition D, and the sixth to condition E. Each coordinate has the value 1 for True, 0 for False, and x for don’t–care; x indicates when a decision is skipped and does not affect the path.

Since there are six conditions in the code unit, Table 5-5 demonstrates that there are 64 possible vectors in the vector space.

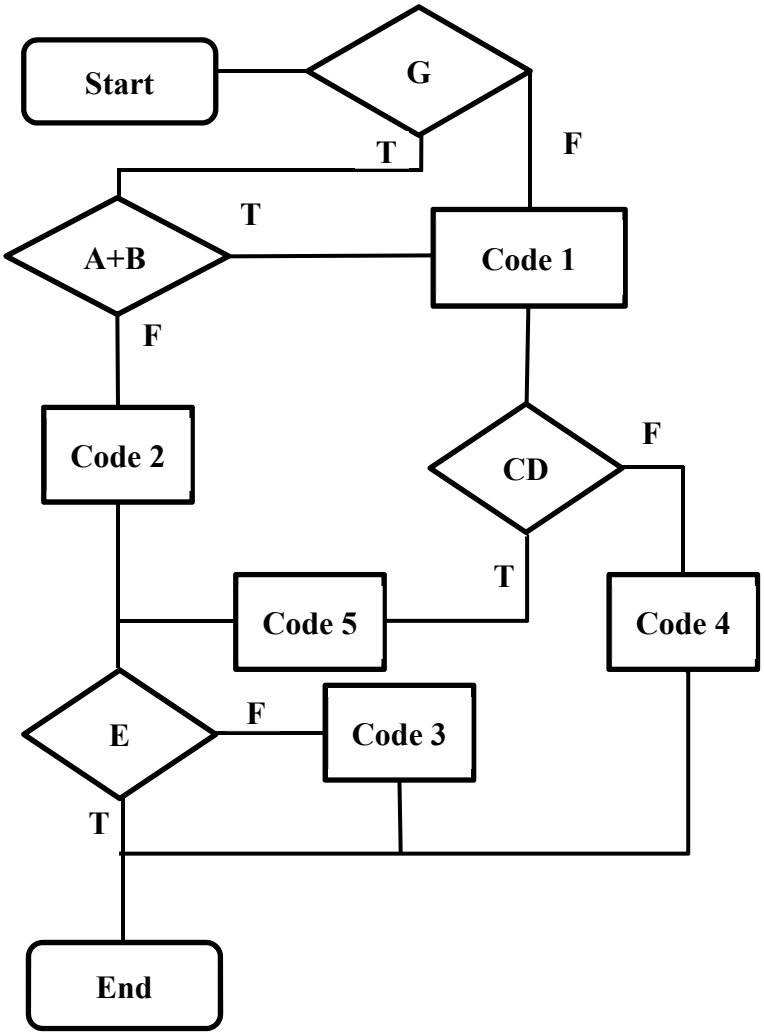

FIGURE 5-4. Example 2 Flow Chart.

TABLE 5-5. List of Vectors in Underlying Vector Space of Code Unit in Figure 5-4.

Vectors in Vector Space	Number of Vectors
(0, x, x, 1, 1, 1)	4
(0, x, x, 1, 1, 0)	4
(0, x, x, 0, x, x)	16
(0, x, x, 1, 0, x)	8
(1, 1, 1, 1, 1, 1)	1
(1, 1, 1, 1, 1, 0)	1
(1, 1, 1, 0, x, x)	4
(1, 1, 1, 1, 0, x)	2
(1, 0, x, x, x, 1)	8
(1, 0, x, x, x, 0)	8
(1, 1, 0, x, x, 1)	4
(1, 1, 0, x, x, 0)	4

Table 5-6 displays the candidate basis set for Example 2. The next step is to demonstrate that the chosen candidate basis set spans the vector space.

TABLE 5-6. Candidate Basis Set for Underlying Vector Space of Figure 5-4.

Vector Name	Vector
B_1	(1, 1, 1, 1, 1, 1)
B_2	(1, 0, 1, 1, 1, 1)
B_3	(0, 1, 1, 1, 1, 0)
B_4	(1, 1, 0, 1, 1, 0)
B_5	(1, 1, 1, 0, 1, 1)
B_6	(1, 1, 1, 1, 0, 1)

The chosen candidate basis set is expressed in matrix form using Equation 5-3 (Figure 5-5), which will assist in validating that the candidate basis set spans the vector space by finding the inverse of the selected candidate basis set.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

FIGURE 5-5. Candidate Basis Set in Matrix Form.

The inverse of a selected candidate basis set was found using the Gauss-Jordan elimination method by applying the XOR addition operation in the field associated with the underlying vector space (see Table 5-7) to reduce the candidate basis set into reduced row echelon form.

TABLE 5-7. The "+'' XOR Operation
for Vector Field.

+	**0**	**1**
0	0	1
1	1	0

The inverse of the candidate basis set expressed in matrix form is demonstrated in Figure 5-6.

$$A^{-1} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

FIGURE 5-6. Example 2 Inverse Matrix.

The next step is to establish that the designated candidate basis set expressed in matrix form is nonsingular. That is established by multiplying the inverse A^{-1} (from Figure 5-6) by the established candidate basis set in matrix form (Figure 5-5) to obtain the identity matrix. This is done using the XOR field operation with the AND field operation (see Table 5-8).

TABLE 5-8. The "*''AND Operation for
Vectors in a Field.

*	**0**	**1**
0	0	0
1	1	1

Table 5-9 demonstrates the method of multiplying the inverse matrix by the candidate basis set using both the field AND multiplication and field XOR addition operation to validate the identity matrix.

The inverse of the matrix form of the chosen candidate basis set for Example 2 does yield the identity matrix (reference Figure 5-7), which validates there exists only one solution and that the vectors are linearly independent.

Next is to verify that the chosen candidate basis set spans the vector space; this is demonstrated by determining if the matrix is non-singular.

The vectors in the basis set were written in matrix form using Equation 5-2 and entering the coefficient matrix into a Cramer Ruler calculator. The determinant of the coefficient matrix resulted in being -1 and hence the coefficient matrix is non-singular and demonstrated that a solution exists.

Example 2 contains 64 possible vectors, which would require a large test set to ensure that every path in the code unit is initiated. Following the steps provided in this report reduced Example 2 from 64 possible vectors to just 6 vectors, which would decrease the number of tests without sacrificing thoroughness.

TABLE 5-9. Example 2 Sample Calculations.

	"*" AND and " + " XOR operation	Result
a_{11}	1(1) + 1(1) + 0(0) + 1(1) + 1(1) + 1(1) = 1 + 1 + 0 + 1 + 1 + 1	1
a_{12}	1(1) + 1(0) + 0(1) + 1(1) + 1(1) + 1(1) = 1 + 0 + 0 + 1 + 1 + 1	0
a_{13}	1(1) + 1(1) + 0(1) + 1(0) + 1(1) + 1(1) = 1 + 1 + 0 + 0 + 1 + 1	0
a_{14}	1(1) + 1(1) + 0(1) + 1(1) + 1(0) + 1(1) = 1 + 1 + 0 + 1 + 0 + 1	0
a_{15}	1(1) + 1(1) + 0(1) + 1(1) + 1(1) + 1(0) = 1 + 1 + 0 + 1 + 1 + 0	0
a_{16}	1(1) + 1(1) + 0(0) + 1(0) + 1(1) + 1(1) = 1 + 1 + 0 + 0 + 1 + 1	0
a_{21}	1(1) + 1(1) + 0(0) + 0(1) + 0(1) + 0(1) = 1 + 1 + 0 + 0 + 0 + 0	0
a_{22}	1(1) + 1(0) + 0(1) + 0(1) + 0(1) + 0(1) = 1 + 0 + 0 + 0 + 0 + 0	1
a_{23}	1(1) + 1(1) + 0(1) + 0(0) + 0(1) + 0(1) = 1 + 1 + 0 + 0 + 0 + 0	0
a_{24}	1(1) + 1(1) + 0(1) + 0(1) + 0(0) + 0(1) = 1 + 1 + 0 + 0 + 0 + 0	0
a_{25}	1(1) + 1(1) + 0(1) + 0(1) + 0(1) + 0(0) = 1 + 1 + 0 + 0 + 0 + 0	0
a_{26}	1(1) + 1(1) + 0(0) + 0(0) + 0(1) + 0(1) = 1 + 1 + 0 + 0 + 0 + 0	0
a_{31}	1(1) + 1(1) + 1(0) + 0(1) + 1(1) + 1(1) = 1 + 1 + 0 + 0 + 1 + 1	0
a_{32}	1(1) + 1(0) + 1(1) + 0(1) + 1(1) + 1(1) = 1 + 0 + 1 + 0 + 1 + 1	0
a_{33}	1(1) + 1(1) + 1(1) + 0(0) + 1(1) + 1(1) = 1 + 1 + 1 + 0 + 1 + 1	1
a_{34}	1(1) + 1(1) + 1(1) + 0(1) + 1(0) + 1(1) = 1 + 1 + 1 + 0 + 0 + 1	0
a_{35}	1(1) + 1(1) + 1(1) + 0(1) + 1(1) + 1(0) = 1 + 1 + 1 + 0 + 1 + 0	0
a_{36}	1(1) + 1(1) + 1(0) + 0(0) + 1(1) + 1(1) = 1 + 1 + 0 + 0 + 1 + 1	0
a_{41}	1(1) + 0(1) + 0(0) + 0(1) + 1(1) + 0(1) = 1 + 0 + 0 + 0 + 1 + 0	0
a_{42}	1(1) + 0(0) + 0(1) + 0(1) + 1(1) + 0(1) = 1 + 0 + 0 + 0 + 1 + 0	0
a_{43}	1(1) + 0(1) + 0(1) + 0(0) + 1(1) + 0(1) = 1 + 0 + 0 + 0 + 1 + 0	0
a_{44}	1(1) + 0(1) + 0(1) + 0(1) + 1(0) + 0(1) = 1 + 0 + 0 + 0 + 0 + 0	1
a_{45}	1(1) + 0(1) + 0(1) + 0(1) + 1(1) + 0(0) = 1 + 0 + 0 + 0 + 1 + 0	0
a_{46}	1(1) + 0(1) + 0(0) + 0(0) + 1(1) + 0(1) = 1 + 0 + 0 + 0 + 1 + 0	0
a_{51}	1(1) + 0(1) + 0(0) + 0(1) + 0(1) + 1(1) = 1 + 0 + 0 + 0 + 0 + 1	0
a_{52}	1(1) + 0(0) + 0(1) + 0(1) + 0(1) + 1(1) = 1 + 0 + 0 + 0 + 0 + 1	0
a_{53}	1(1) + 0(1) + 0(1) + 0(0) + 0(1) + 1(1) = 1 + 0 + 0 + 0 + 0 + 1	0
a_{54}	1(1) + 0(1) + 0(1) + 0(1) + 0(0) + 1(1) = 1 + 0 + 0 + 0 + 0 + 1	0
a_{55}	1(1) + 0(1) + 0(1) + 0(1) + 0(1) + 1(0) = 1 + 0 + 0 + 0 + 0 + 0	1
a_{56}	1(1) + 0(1) + 0(0) + 0(0) + 0(1) + 1(1) = 1 + 0 + 0 + 0 + 0 + 1	0
a_{61}	0(1) + 1(1) + 1(0) + 1(1) + 1(1) + 1(1) = 0 + 1 + 0 + 1 + 1 + 1	0
a_{62}	0(1) + 1(0) + 1(1) + 1(1) + 1(1) + 1(1) = 0 + 0 + 1 + 1 + 1 + 1	0
a_{63}	0(1) + 1(1) + 1(1) + 1(0) + 1(1) + 1(1) = 0 + 1 + 1 + 0 + 1 + 1	0
a_{64}	0(1) + 1(1) + 1(1) + 1(1) + 1(0) + 1(1) = 0 + 1 + 1 + 1 + 0 + 1	0
a_{65}	0(1) + 1(1) + 1(1) + 1(1) + 1(1) + 1(0) = 0 + 1 + 1 + 1 + 1 + 0	0
a_{66}	0(1) + 1(1) + 1(0) + 1(0) + 1(1) + 1(1) = 0 + 1 + 0 + 0 + 1 + 1	1

$$A^{-1}A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 5-7. Example 2 Inverse Matrix Calculation.

5.5 WHAT IS EXPECTED IN PRACTICE

What happens in practice? The reality is that decisions have generally many conditions. A path per condition in each decision is required. The tester does not bother to verify that his/her definition of the vector binary operation in fact establishes a vector space. Instead, the tester creates a candidate basis by verifying that every member of this basis adds a new condition that no other member exercised. There is the necessity to show that the candidate basis set covers the underlying vector space, or equivalently, that there are no missing vectors in the basis set. This is typically done by verifying that there is at least one vector in the basis set that covers each condition in all the decisions in the code unit. Linear independence is generally taken for granted, except if an experienced mathematician is performing the testing. Then the tester writes one test for each of the vectors in the basis set.

The method of constructing basis sets as described above is believed to create a test on the level of MC/DC as defined in DO-178B/C (Reference 2). This is in contrast with McCabe's approach, which is just decision coverage (DAL 2).

6.0 SUMMARY

Managing Unit Tests with Vector Spaces uses the idea that on every code unit an underlying vector space can be constructed, with a basis set that serves as a guide to a set of unit tests. This set of tests is a complete logical test of the code unit, since every decision is tested in both directions. Moreover, the two Boolean values of every condition each belong to different vectors in the basis set.

This ensures that there will be a different result for each value of the condition, similar to what is expected in MC/DC testing. This is MC/DC testing according to DO-178B/C (Reference 2) provided all other conditions remain fixed.

Methods for proving that the set of all vectors form a vector space and that a candidate basis set spans the space and is a linearly independent set are provided. The use of basis sets to design a minimal set of unit tests and to demonstrate that a basis set of tests are adequate for a logical test of a code unit containing a multitude of paths is introduced. The argument is made that this approach is equivalent to MC/DC testing as defined in DO-178B/C.

REFERENCES

1. National Institute of Standards and Technology. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*, by Arthur H. Watson and Thomas J. McCabe. Gaithersburg, Maryland, NIST, August 1996. (NIST Special Publication 500-235.)

2. Radio Technical Commission for Aeronautics. *DO-178B, Software Considerations in Airborne Systems and Equipment Certification*. Washington, D.C., RTCA, December 1992.

3. K. Keremedis. "Bases for Vector Spaces Over the Two-Element Field and the Axiom of Choice," *Proc. Amer. Math. Soc.*, Vol. 124 (1996), pp. 2527-2531.

GLOSSARY

Item	Definition
Associative	Refers to the associative law, or the axiom known as the associative law: $a + (b + c) = (a + b) + c$.
Axiom	A proposition that is accepted as being true without proof.
Basis Set	A subset of a vector space from which all vectors in this space can be generated.
Commutative	A property of a binary operation where changing the order of the operands does not change the result: $a + b = b + a$.
Compound Decision	A decision containing more than one condition.
Condition	The smallest entity that takes on a Boolean value.
Coordinate	Also known as components. A coordinate/component of a vector is the projection of the vector onto a decision/condition.
Coordinate Vector	A representation of a vector as an ordered list of coordinates.
Decision	An entity, consisting of one or more conditions connected by Boolean operations, and taking on a Boolean value. A decision consisting of more than one condition is also known as a compound decision.
Deconstruction	The process of representing a compound decision as several simple decisions, each of which contains just one condition.
Dimension	For a coordinate vector space, the dimension is the number of coordinates.
Distributes	A property of two binary operations where one distributes over the other. For example, $a * (b + c) = a * b + a * c$.
Field	A set of elements on which the abelian group operations of addition and multiplication are defined, and where addition distributes over multiplication.
Inverse	The inverse of an element with respect to a binary operation is an element such that when combined with the first element, the result is the zero element with respect to that binary operation. If the element is a vector v and the binary operation is "+", then w is an inverse of v if and only if $v + w = w + v = 0$, where 0 is the zero vector.
Linearly Independent	A set of vectors is linearly independent if a linear combination of this set being equated to zero implies that the coefficients of the linear combination are zero. For example, $\sum_1^n a_i v_i = 0 \rightarrow a_i = 0$ for *i* from 1 to n.
Ordered Pairs	Two numbers written in a particular order e.g., (3, 4) ≠ (4, 3).
Path	An order of execution for statements in a computer program.

Item	Definition
Path Vector	A vector that defines a path in a code unit by identifying the decisions through which the control moves.
Simple Decision	A decision consisting of only one condition.
Software Code Unit	A smallest compliable code entity.
Span	The property of a basis set that all vectors in the vector space can be generated from a linear combination of vectors in that basis set.
Structured Code	A code method where no jumps into or out of loops are permitted.
Underlying Vector Space	A vector space constructed on a code unit such that the coordinates of the vectors represent the decisions in the code unit.
Unit Test	A test of the lowest level requirements used to develop a code unit.
Vector	A quantity with several coordinates/ components.
Vector Space	A set of vectors and a coefficient field that satisfy certain axioms.
Zero Vector	A vector that when added to another vector yields the other vector.

BIOGRAPHY

Anthony S. Cantone, BS Aeronautical Engineering, PhD Mathematics, International System Safety Society (ISSS) Member, System Safety Analyst at the Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, CA. Phone: 805-717-3631; email: anthony_cantone@yahoo.com

Loren C. Edwards, BS Aerospace Engineering, System Safety Engineer at NAWCWD, China Lake, CA. Phone: 831-234-4406; email: LCEdwards@gmail.com

This page intentionally left blank.

INITIAL DISTRIBUTION

1 Defense Technical Information Center, Fort Belvoir, VA

ON-SITE DISTRIBUTION

1 Code DC12100 (file copy)
1 Code D5J4000 (archive copy)
2 Code D510000, Edwards, L.
2 Code D511000, Cantone, A.