Functional Dependence and Equivalence Class Factors in Combinatorial Test Designs

George B. Sherwood
Testcover.com, LLC
Colts Neck, NJ USA
sherwood@testcover.com

***Abstract*—Functionally dependent (FD) mappings permeate software systems and impose constraints on designs for verification. This paper develops concepts for accommodating FD relations in combinatorial test designs and derives rules to determine combinatorial coverage for FD factors. FD equivalence class factors are introduced to assess coverage of expected results classes before test case generation. Examples are given for test designs with FD factors, for applications of the coverage rules, and for coverage improvement techniques. A nondeterminant strength classification is formulated.**

Keywords—combinatorial testing; coverage analysis; equivalence class; equivalence partitioning; functional dependence; interaction testing; test design

I. INTRODUCTION

Software systems comprise innumerable processes and data records with deterministic relationships. Functions that accept input data and return specific results are essential to many programming languages. The idea that data values can be determined by other values is central to the design of relational databases [1]. This paper aims to use properties of these ubiquitous relations for better treatment in combinatorial test designs.

Functionally dependent (FD) relations appear as constraints in test models. A test factor is functionally dependent when its value is identified by the values of its determinant factors. For example, to test an application with a date input, the value last_day(month,year) may be needed to verify month boundaries. To test a state model for a shopping cart application, an event QTY(i,q) may be required to change the i^{th} item's quantity to q. The general model that test input and configuration values map to an expected result is an FD relation itself.

Combinatorial testing, as commonly practiced, involves generating one partition of test cases for multiple equivalence classes. A *partition* includes the allowed combinations of test factor values for a test case generation instance. An *equivalence class* includes the combinations of test factor values leading to one class of expected results. The prospect of a partition yielding results in multiple equivalence classes is clear from the invitation to this workshop [2], which notes "the difficulty in determining expected results for the generated tests." This paper provides a formulation for minimum coverage of each equivalence class from a single partition. The formulation is based on the number of determinant test factors in the FD relation of the equivalence class. An analysis of generated test cases is not needed to find the minimum coverage. Thus, the test designer has a new tool to evaluate the partitioning and strength of a test design, as it relates to some or all equivalence classes.

Six coverage rules are derived for FD test factors and are illustrated with examples. In particular, the nondeterminant strength rule is the basis for the equivalence class coverage formulation. A body mass index (BMI) report application provides examples of equivalence classes in various test designs. Combinations of five input factors (Age, Weight, Height, Sex, Intake) cause the application to generate one or two reports. These are the Medicare report (depending on Age), the Child report (depending on Age and Sex), and the Adult report (depending on Age, Weight, and Height). The nondeterminant strength rule applied to an FD equivalence class factor for each report gives the minimum coverage for the report's equivalence classes. A strength-3 test design using a single partition for all classes is shown to have limited coverage for the Adult report equivalence classes (underweight, normal, overweight, obese). The limitation is improved two ways, with

- An increase to a strength-4 design, or
- A repartitioned strength-2 design aligning partitions with selected equivalence classes.

The repartitioning offers additional freedom to choose values, e.g. to verify equivalence class boundaries.

Use of the FD coverage rules with equivalence class factors enables coverage assessment for individual equivalence classes, which can lead to more design choices and more effective testing. Moreover, the classification of designs according to nondeterminant strength may offer some clarification for the testing challenges of large software systems [3].

II. FUNCTIONALLY DEPENDENT TEST INPUTS

A. Equivalence Classes

Equivalence partitioning is a well-known technique for separating test factor combinations into classes of similar expected results. For example, test inputs to a program that converts Celsius temperatures to Fahrenheit may have two classes, one for valid input temperatures, like 20 ^{o}C, and an invalid class for inputs below absolute zero. Inputs for the first class are converted to Fahrenheit temperatures according to the usual formula. Inputs for the second are required to give an exception result instead. The partitions divide the inputs

978-0-7695-5194-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ICSTW.2014.12

according to their different classes of results, and inputs in the same partition should yield equivalent results.

Organizing test factor combinations into equivalence classes provides a framework for test design, which can help to avoid masking [4,5,6] and to identify test cases at the boundaries between classes. Generally mixing test cases from different classes can obscure what is or is not tested. Moreover, if test cases for different classes *are* combined by one partition, some classes of results might receive less than the intended test coverage.

Each equivalence class has a set of allowed combinations. In the valid class of this example, 20 °C is an allowed input, but –300 °C is disallowed because it is below absolute zero. In the invalid class –300 °C is allowed, but 20 °C is disallowed.

B. Functionally Dependent Factors

Each partition in the test model has a test array with k factors, and the j^{th} factor has v_j values. The partition's N-by-k test array has strength t when every subarray of t factors has every combination of values (every t-tuple) in at least one test case. Constraints arise when some combinations are disallowed in a partition. In the valid class of the example, the minimum allowed temperature represents a constraint. It sets the boundary between valid and invalid input temperatures. So for integer inputs the minimum valid temperature is –273 °C; the maximum invalid temperature is –274 °C.

The j^{th} factor is *functionally dependent* when its value is identified by the values of l_j *determinant* factors. Other *independent* factors which are not part of this relation are *nondeterminant* (ND). There are three input factors in the example:

- inScale, the scale for the input temperature (Celsius, Fahrenheit or Kelvin)
- inTemp, a number of degrees on the input scale
- outScale, the scale for the converted temperature (Celsius, Fahrenheit or Kelvin)

Now the minimum valid inTemp value depends on inScale. That temperature is –273 °C, –459 °F or 0 °K. Thus,

- inScale is a determinant factor
- inTemp is an FD factor
- outScale is an ND factor

The functional dependence can be expressed as inScale → minimum or as minimum(inScale).

Typically FD factors are *composite*. They may include fixed values and possibly multiple functions. In the date input example, the day factor may need the first day and an intermediate day, as well as the last day: 1, 10, and last_day(month,year). In addition to QTY(i,q), the event factor for the shopping cart application may need a CHECK(i) function to mark the i^{th} item for deletion and UPDATE to complete the change. In the temperature conversion example, the inTemp factor has the following values.

- minimum(inScale); a function to test the equivalence class boundary
- 20, 68; two fixed values
- freezing(inScale), boiling(inScale); two functions representing the freezing and boiling temperatures of water on inScale

These values are chosen so that any combination of the input values is in the valid equivalence class[1].

The specification of values in *functionally dependent form* [7] can clarify the constraint analysis in a test design by making the relations among factor values explicit. The last_day(month,year) function has one meaning although it represents four values: 28, 29, 30, and 31. Without the minimum(inScale) designation it is not so clear when –273 and 0 are boundary values. And without the index notation of CHECK(i), the behavior of the shopping cart is more challenging to analyze.

C. Functionally Dependent Test Cases

Direct Product Block (DPB) notation [7,8] specifies test combinations and their constraints using blocks of values. All combinations from each block are allowed. The combinations from all the blocks define the partition's allowed combinations, independent of the test design strength.

Table I describes the temperature conversion partition in FD form, using DPB notation. The first line contains a title, and the next three label the test factors. The line starting with a number sign (#) marks the beginning of the partition, which contains a block of three lines. Each test factor takes the values on its respective line. All combinations of values in the block are allowed.

Typically the partition in Table I is converted from FD form to *fixed values form* [9] before the test cases are generated. The individual values of the functions are represented in separate blocks conforming to the mappings of the functions. Table II shows the temperature conversion partition in fixed values form. Now the block from Table I is split into three blocks according to the values of inScale. In Table II the lines starting with a plus (+) start the three blocks. Again, all combinations of values in each block are allowed[2].

Table III lists the strength-2 test cases generated from the partition of Table II. All pairs from each block are included, but other pairs are not included due to the constraints. E.g.

TABLE I. PARTITION WITH FUNCTIONALLY DEPENDENT COMPOSITE FACTOR

```
Temperature Conversion - functionally dependent form
inScale
inTemp
outScale
# inTemp functions depend on inScale values
Celsius Fahrenheit Kelvin
minimum(inScale) 20 68 freezing(inScale) boiling(inScale)
Fahrenheit Kelvin Celsius
```

[1] Equivalence classes and factor values in this paper are chosen to illustrate points briefly. A complete test plan would address all equivalence classes and include other factor values.

[2] The values of each function can be expressed in terms of its domain or range, e.g. boiling(Fahrenheit) or 212. Using the range has advantages of possibly fewer blocks, and of avoiding the need for another substitution after test case generation. In this example, values in terms of the domain are used so that the FD relations continue to be explicit.

TABLE II. PARTITION WITH FIXED VALUES

```
Temperature Conversion - fixed values form
inScale
inTemp
outScale
# block split by inScale values
+Celsius inScale
Celsius
minimum(Celsius) 20 68 freezing(Celsius) boiling(Celsius)
Fahrenheit Kelvin Celsius
+Fahrenheit inScale
Fahrenheit
minimum(Fahrenheit) 20 68 freezing(Fahrenheit) boiling(Fahrenheit)
Fahrenheit Kelvin Celsius
+Kelvin inScale
Kelvin
minimum(Kelvin) 20 68 freezing(Kelvin) boiling(Kelvin)
Fahrenheit Kelvin Celsius
```

inScale Kelvin is not paired with boiling(Fahrenheit) or 212. There are 11 fixed values for the inTemp factor, so the number of test cases is the minimum ($33 = 11 \cdot 3$).

One question that may arise is what happens if test cases are generated from factor values in FD form? The answer for this example is given in Table IV, which shows the strength-2 test cases generated from the partition of Table I. In Table IV the inTemp minimum(inScale) function is paired with all three inScale values, so the three boundary inputs are present. Each of the other two functions and the two fixed values for inTemp also are paired with the three inScale values. Thus, all allowed

TABLE III. TEST CASES GENERATED WITH FIXED VALUES

	inScale	*inTemp*	*outScale*
1	Celsius	freezing(Celsius)	Fahrenheit
2	Celsius	minimum(Celsius)	Kelvin
3	Celsius	20	Celsius
4	Fahrenheit	freezing(Fahrenheit)	Fahrenheit
5	Fahrenheit	minimum(Fahrenheit)	Kelvin
6	Fahrenheit	68	Celsius
7	Kelvin	freezing(Kelvin)	Fahrenheit
8	Kelvin	minimum(Kelvin)	Kelvin
9	Kelvin	boiling(Kelvin)	Celsius
10	Celsius	boiling(Celsius)	Kelvin
11	Celsius	68	Kelvin
12	Fahrenheit	boiling(Fahrenheit)	Kelvin
13	Fahrenheit	20	Fahrenheit
14	Kelvin	20	Kelvin
15	Kelvin	68	Fahrenheit
16	Celsius	minimum(Celsius)	Fahrenheit
17	Celsius	freezing(Celsius)	Celsius
18	Celsius	boiling(Celsius)	Fahrenheit
19	Celsius	minimum(Celsius)	Celsius
20	Celsius	freezing(Celsius)	Kelvin
21	Celsius	boiling(Celsius)	Celsius
22	Fahrenheit	minimum(Fahrenheit)	Fahrenheit
23	Fahrenheit	freezing(Fahrenheit)	Celsius
24	Fahrenheit	boiling(Fahrenheit)	Fahrenheit
25	Fahrenheit	minimum(Fahrenheit)	Celsius
26	Fahrenheit	freezing(Fahrenheit)	Kelvin
27	Fahrenheit	boiling(Fahrenheit)	Celsius
28	Kelvin	boiling(Kelvin)	Kelvin
29	Kelvin	minimum(Kelvin)	Fahrenheit
30	Kelvin	freezing(Kelvin)	Celsius
31	Kelvin	boiling(Kelvin)	Fahrenheit
32	Kelvin	minimum(Kelvin)	Celsius
33	Kelvin	freezing(Kelvin)	Kelvin

TABLE IV. TEST CASES GENERATED WITH FUNCTIONALLY DEPENDENT COMPOSITE FACTOR

	inScale	*inTemp*	*outScale*
1	Celsius	minimum(inScale)	Fahrenheit
2	Fahrenheit	20	Celsius
3	Kelvin	68	Kelvin
4	Fahrenheit	freezing(inScale)	Kelvin
5	Kelvin	boiling(inScale)	Celsius
6	Celsius	20	Kelvin
7	Kelvin	20	Fahrenheit
8	Celsius	68	Celsius
9	Fahrenheit	68	Fahrenheit
10	Fahrenheit	minimum(inScale)	Kelvin
11	Kelvin	minimum(inScale)	Celsius
12	Celsius	freezing(inScale)	Fahrenheit
13	Kelvin	freezing(inScale)	Celsius
14	Celsius	boiling(inScale)	Fahrenheit
15	Fahrenheit	boiling(inScale)	Kelvin

value pairs of the determinant and FD factors occur.

It is also evident that all pairs of determinant and ND values are present. However the coverage of the FD and ND values is not so strong. Each outScale value is paired with each of the inTemp fixed values, but it is paired with each function only once. For example, the outScale value Fahrenheit is paired with minimum(Celsius) in test case 1, but it is not paired with minimum(Fahrenheit) or minimum(Kelvin).

III. COVERAGE WITH FUNCTIONALLY DEPENDENT FACTORS

A. Individual Factor Rules

The rules in this section describe coverage of test cases generated in FD form for one FD factor. The factor may be composite, with fixed values and/or a number of different functions with the same l_j determinant factors and $l_j < t$.

In a partition's test array, a subarray consisting of a functionally dependent factor j and its n_j nondeterminant factors is a *nondeterminant subarray J*. *J* has *nondeterminant strength* s_j when every subarray of s_j factors *including j* has every combination of values (every s_j-tuple) in at least one test case. The ND strength has an upper bound: $s_j \le n_j + 1$.

The definition of ND strength applies when *J* is a covering array of strength s_j. It also applies when the subarray of ND factors covers with strength $s_j - 1$ and the FD factor is a single-value factor [7]. In this case every (s_j–1)-tuple of ND factors is associated with the FD factor's value.

C1. *Determinant coverage rule:* The subarray of the FD factor and its determinant factors covers all allowed (l_j+1)-tuples.

All allowed l_j-tuples of the determinant factors are covered because $l_j < t$. These combinations are associated with each of the FD factor's functions because $l_j + 1 \le t$. These combinations also determine the allowed values of the dependent functions. Thus, all the allowed (l_j+1)-tuples are covered.

C1 shows that in FD form the coverage of a composite FD factor with its determinant factors can be less than or equal to the generation strength *t*. In the temperature conversion example $l_j = 1$, so all

allowed pairs (2-tuples) of inScale and inTemp values are covered.

C2. *Independent coverage rule:* Every allowed (t–1)-tuple of independent (determinant and nondeterminant) factors is associated with at least one value of each of the FD factor's functions.

The rule follows from the strength-t test case generation with each function represented as one value.

C2 shows that in FD form the ND subarray of a composite FD factor has ND strength 1. In the temperature conversion example each value of each inTemp function is associated with only one outScale value. Rule C1 specifies stronger coverage for inScale-inTemp pairs than C2.

The coverage of the FD factor with its ND factors could be stronger if the FD factor were not composite, i.e. if it had just one function and no fixed values. If every l_j-tuple of the determinant factors is associated with that function in every test case, then the function could be associated with the ND combinations also. This is the idea behind single-function partitions in the next section.

B. Single-function Partitions

A partition in FD form is repartitioned into *single-function partitions* when each factor j in each partition

- has values of one function identified by the fixed values of l_j determinant factors, or
- has fixed values.

Table I shows the temperature conversion partition for the valid equivalence class in FD form. The inTemp factor is composite, containing three functions of inScale and two fixed values. If this partition is split into four partitions, as in Table V, each of the inTemp functions specifies the only inTemp values in its respective single-function partition. The other partition contains the fixed values of inTemp[3].

TABLE V. PARTITION SPLIT INTO SINGLE-FUNCTION PARTITIONS

Temperature Conversion - single-function partitions
inScale
inTemp
outScale
#min minimum inTemp function
Celsius Fahrenheit Kelvin
minimum(inScale)
Fahrenheit Kelvin Celsius
#fxd fixed inTemp values
Celsius Fahrenheit Kelvin
20 68
Fahrenheit Kelvin Celsius
#frz freezing inTemp function
Celsius Fahrenheit Kelvin
freezing(inScale)
Fahrenheit Kelvin Celsius
#bol boiling inTemp function
Celsius Fahrenheit Kelvin
boiling(inScale)
Fahrenheit Kelvin Celsius

[3] When there are multiple FD factors the partition splitting is repeated for each.

The new partitions still correspond to the valid equivalence class. The repartitioning is just a tactic to force the generation of test cases which associate ND and FD values.

Table VI gives the strength-2 test cases generated from the partitions of Table V. There are nine test cases from each partition for a total of 36, three more than the minimum shown in Table III.

Now, in the test arrays of each of the single-function partitions, when the outScale value is paired with the inScale value, it is always paired with the FD value in the same test case. Thus all pairs of inTemp and outScale values are covered. The inTemp-outScale subarrays have ND strength 2.

C. Single-function Partition Rules

The following rules describe coverage of test cases generated in FD form for a total of M FD factors in a single-function partition. A *functionally dependent set* is any subset of m FD factors whose values are identified by $l \leq t$ determinant factors. The FD set has n ND factors with fixed values. Thus $l + n + M = k$.

C3. *Determinant coverage rule:* Any subarray consisting of an FD set of m factors and its l determinant factors covers all allowed (l+m)-tuples.

TABLE VI. TEST CASES GENERATED WITH SINGLE-FUNCTION PARTITIONS

	inScale	*inTemp*	*outScale*
min1	Kelvin	minimum(inScale)	Celsius
min2	Celsius	minimum(inScale)	Kelvin
min3	Fahrenheit	minimum(inScale)	Fahrenheit
min4	Kelvin	minimum(inScale)	Fahrenheit
min5	Fahrenheit	minimum(inScale)	Celsius
min6	Celsius	minimum(inScale)	Celsius
min7	Kelvin	minimum(inScale)	Kelvin
min8	Fahrenheit	minimum(inScale)	Kelvin
min9	Celsius	minimum(inScale)	Fahrenheit
fxd1	Kelvin	20	Celsius
fxd2	Celsius	20	Kelvin
fxd3	Kelvin	68	Fahrenheit
fxd4	Celsius	68	Celsius
fxd5	Fahrenheit	68	Kelvin
fxd6	Fahrenheit	20	Fahrenheit
fxd7	Fahrenheit	20	Celsius
fxd8	Kelvin	20	Kelvin
fxd9	Celsius	68	Fahrenheit
frz1	Kelvin	freezing(inScale)	Celsius
frz2	Celsius	freezing(inScale)	Kelvin
frz3	Fahrenheit	freezing(inScale)	Fahrenheit
frz4	Kelvin	freezing(inScale)	Fahrenheit
frz5	Fahrenheit	freezing(inScale)	Celsius
frz6	Celsius	freezing(inScale)	Celsius
frz7	Kelvin	freezing(inScale)	Kelvin
frz8	Fahrenheit	freezing(inScale)	Kelvin
frz9	Celsius	freezing(inScale)	Fahrenheit
bol1	Kelvin	boiling(inScale)	Celsius
bol2	Celsius	boiling(inScale)	Kelvin
bol3	Fahrenheit	boiling(inScale)	Fahrenheit
bol4	Kelvin	boiling(inScale)	Fahrenheit
bol5	Fahrenheit	boiling(inScale)	Celsius
bol6	Celsius	boiling(inScale)	Celsius
bol7	Kelvin	boiling(inScale)	Kelvin
bol8	Fahrenheit	boiling(inScale)	Kelvin
bol9	Celsius	boiling(inScale)	Fahrenheit

The m-tuple of the FD set is identified by each l-tuple of its determinant factors. When the partition contains all of the allowed l-tuples, all the allowed (l+m)-tuples are covered.

C3 shows that in a single-function partition in FD form, the coverage of an FD factor with its determinant factors may or may not be the same as the generation strength t. In each single-function partition of the temperature conversion example, $l = 1$ and $m = 1$. So all allowed pairs of inScale and inTemp values are covered.

C4. *Independent coverage rule*: Any subarray consisting of an FD set of m factors, its l determinant factors, and $t - l$ ND factors covers all allowed (t+m)-tuples. When $t \geq l + n$, all allowed (l+n+m)-tuples for the FD set are covered; the allowed k-tuples of the single function partition are covered also.

When $t \leq l + n$, there are at least $t - l$ ND factors. The m-tuple of the FD set is identified by each l-tuple of its determinant factors. Each l-tuple also is associated with the allowed ND (t–l)-tuples because the independent factors cover with strength t. When the partition contains all allowed l-tuples, they are associated with the allowed FD m-tuples and with the allowed ND (t–l)-tuples. Thus all allowed (t+m)-tuples are covered. When $t \geq l + n$, each l-tuple is associated with all FD m-tuples and with all ND n-tuples, so all allowed (l+n+m)-tuples are covered. Since the partition contains all the allowed independent (l+n)-tuples, all M FD factors are determined, and all k-tuples are covered.

C4 shows that in a single-function partition in FD form, the coverage of an FD factor with its independent factors can be greater than the generation strength t. In the single-function partitions of the temperature conversion example, all allowed 3-tuples are covered.

C5. *Nondeterminant strength rule*: A nondeterminant subarray J has nondeterminant strength $s_j = t - l_j + 1$ when $t \leq l_j + n_j$; $s_j = n_j + 1$, when $t \geq l_j + n_j$.

C5 follows from C4 when $m = 1$, and the l_j determinant factors are not included in the subarray.

C5 shows that in a single-function partition in FD form, the ND strength might be less than the generation strength t due to constraints of the FD relation. In all the single-function partitions of the temperature conversion example, all allowed inTemp-outScale pairs are covered, and the ND strength is 2.

C6. *Equivalent coverage rule*: When each factor in a set of M FD factors has exactly one determinant factor, all allowed t-tuples are covered, and each nondeterminant subarray has nondeterminant strength $s_j = t$.

When a t-tuple contains values of m FD factors, it contains values of $t - m$ independent factors also. The m FD factors are identified by $l \leq m$ determinant factors because each $l_j = 1$. Thus the t-tuple is determined by at most $l + (t - m) \leq t$ independent, covered factor values. I.e. all allowed t-tuples are covered. Each nondeterminant subarray has ND strength $s_j = t$ by C5 because $l_j = 1$.

C6 shows the conditions when test cases generated from a single-function partition in FD form have equivalent strength as those generated from a partition in fixed values form. In the single-function partitions of the temperature conversion example, $l_j = 1$. All allowed pairs are covered.

IV. Equivalence Class Factors

In a deterministic system every mapping of input and configuration values to a result can be considered a functionally dependent relation. And if each result maps to one equivalence class, then that class is determined by the input and configuration values as well.

input, configuration values → result
→ equivalence class

In the temperature conversion example, an equivalence class factor can be defined as in Table VII. This factor is functionally dependent on two input factors, inScale and inTemp.

The equivalence class factor characterizes expected results, so it can be useful for analyzing coverage of test designs. The idea is to use equivalence class factors defined by requirements as FD factors in single-function partitions. Thus the functionally dependent coverage rules apply, and coverage of the equivalence classes can be assessed, even before test cases are generated.

A. Equivalence Classes from Requirements

Requirements for a body mass index (BMI) report application include the following.

R1. The listed input data will be stored in the patient database table.
   a. Age in years
   b. Weight in pounds
   c. Height in inches
   d. Sex (female, male)
   e. Intake in kilocalories per day

R2. The BMI will be calculated (in kilograms per meter squared) as $703.06957964 \text{ Weight} / \text{Height}^2$ and stored in the patient database table.

R3. If Age is 65 years or older, the Medicare report will be generated.

TABLE VII. Temperature Conversion Equivalence Class Factor Definition

Input factors		Equivalence class factor
inScale	*inTemp*	
Kelvin	≤ –1	invalid
Kelvin	≥ 0	valid
Celsius	≤ –274	invalid
Celsius	≥ –273	valid
Fahrenheit	≤ –460	invalid
Fahrenheit	≥ –459	valid

R4. If Age is younger than 20 years, the Child report containing the BMI percentile will be generated for the corresponding listed classification.
    a. Girl, from the female BMI-age table
    b. Boy, from the male BMI-age table
R5. If Age is 20 years or older, the Adult report will be generated for the corresponding listed classification.
    a. Underweight, BMI < 18.5
    b. Normal, 18.5 ≤ BMI < 25.0
    c. Overweight, 25.0 ≤ BMI < 30.0
    d. Obese, 30.0 ≤ BMI

The BMI calculation is an expected result of R2. It is an FD factor of Weight and Height. R3, R4, and R5 suggest three equivalence class factors which are functionally dependent on inputs:

Age	→ Medicare classes
Age, Sex	→ Child classes
Age, Weight, Height	→ Adult classes

These FD relations are used to assess coverage for their equivalence classes.

B. *Strength-3 Design*

If a strength-3 test design is chosen using the five inputs of R1 in one partition, the coverage rules imply the following for the FD equivalence class factors Medicare, Child, and Adult.

C3 and C4 apply to any set of m FD factors for which $l \leq t$. For example, the two FD factors, Medicare and Child have two determinant factors, Age and Sex. C3 shows that all their allowed 4-tuples are covered. C4 indicates that when one ND factor (Weight, Height or Intake) is included in the subarray, all allowed 5-tuples are covered.

C5 applies to individual FD factors. It describes the coverage of each FD factor with its ND factors. For the Medicare, Child, and Adult factors, the ND strengths are 3, 2, and 1 respectively, indicating 3-tuples, 2-tuples, and 1-tuples are covered in the corresponding ND subarray.

C6 does not apply to the BMI report example because $l_j > 1$ for the Child and Adult factors.

Table VIII shows the partition to generate test cases in this example. It has the five input factors from R1 and the three equivalence class factors from R3, R4, and R5. Three Age values are chosen to verify R3, R4, and R5. Weight and Height values are chosen so that all four BMI classifications result.

Table IX lists the test cases for this partition. The input factor values from R1 cover with strength-3. The corresponding fixed values for the equivalence class factors are given. The expected BMI is shown for reference.

The Medicare value (yes or no) indicates the Medicare report is or is not an expected result. Each equivalence class is associated with the allowed values of its determinant factor, Age. The ND subarray of Weight, Height, Sex, Intake, and Medicare factors has ND strength 3. I.e. all allowed pairs of the Medicare ND factors are associated with each equivalence class.

TABLE VIII. PARTITION WITH FIXED INPUT VALUES AND FUNCTIONALLY DEPENDENT EQUIVALENCE CLASS FACTORS

Body Mass Index Report Application
Age
Weight
Height
Sex
Intake
Medicare equivalence class
Child equivalence class
Adult equivalence class
#
19 42 67
131 180
64 71
female male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)

The Child value (girl or boy) indicates the Child report is an expected result for the indicated class; no means no Child report is expected. Each equivalence class is associated with the allowed pairs of its determinant factors, Age and Sex. The ND subarray of Weight, Height, Intake, and Child factors has ND strength 2. I.e. all allowed values of the Child ND factors are associated with each equivalence class.

The Adult value (underweight, normal, overweight or obese) indicates the Adult report is an expected result for the indicated class; no means no Adult report is expected. Each equivalence class is associated with the allowed 3-tuples of its determinant factors, Age, Weight, and Height. The ND subarray of Sex, Intake, and Adult factors has ND strength 1. I.e. all allowed values of the Adult ND factors might not be associated with each equivalence class. Examination of the test cases in Table IX shows that only six of the eight Sex-Adult pairs are present; female-obese and male-underweight are missing. Similarly two of the eight Intake-Adult pairs, 3000-obese and 2000-underweight, are missing.

C. *Strength-4 Design*

If one of the test design goals is to associate each equivalence class with its ND factor values, the test cases of Table IX do not meet that goal. One straightforward response is to increase the strength of the design. Table X lists test cases for the same partition shown in Table VIII. The input factor values from R1 cover with strength 4 now. The corresponding fixed values for the equivalence class factors are given. The expected BMI is shown also.

Now the ND strengths of the Medicare, Child, and Adult factors are 4, 3, and 3 respectively. In particular the strength-4 design of Table X achieves ND strength of 3 for the Adult factor, which exceeds the minimum given by C5.

Now all allowed $(t–l)$-tuples (3-tuples) of the Medicare ND factors are associated with each equivalence class. Similarly all allowed *pairs* of the Child ND factors are associated with each equivalence class. And all allowed *pairs* of the Adult ND factors are associated with each equivalence class.

TABLE IX. STRENGTH-3 TEST CASES AND CORRESPONDING EQUIVALENCE CLASS FACTOR VALUES

	Input factors					Equivalence class factors			An expected result
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*	*Child*	*Adult*	*BMI*
1	19	131	64	female	2000	no	girl	no	22.5
2	19	131	64	male	3000	no	boy	no	22.5
3	19	131	71	male	2000	no	boy	no	18.3
4	19	180	64	female	3000	no	girl	no	30.9
5	19	180	71	female	2000	no	girl	no	25.1
6	19	180	71	male	3000	no	boy	no	25.1
7	42	131	64	female	2000	no	no	normal	22.5
8	42	131	64	male	3000	no	no	normal	22.5
9	42	131	71	female	3000	no	no	underweight	18.3
10	42	180	64	male	2000	no	no	obese	30.9
11	42	180	71	female	2000	no	no	overweight	25.1
12	42	180	71	male	3000	no	no	overweight	25.1
13	67	131	64	female	2000	yes	no	normal	22.5
14	67	131	64	male	3000	yes	no	normal	22.5
15	67	131	71	female	3000	yes	no	underweight	18.3
16	67	180	64	male	2000	yes	no	obese	30.9
17	67	180	71	female	2000	yes	no	overweight	25.1
18	67	180	71	male	3000	yes	no	overweight	25.1

D. Strength-2 Design

If one of the test design goals is to associate each equivalence class with its ND factor values, another straightforward response is to align test case generation partitions with their equivalence classes as needed for the desired coverage. When a partition is designed for results within one equivalence class, the corresponding FD factor is a single-value factor of that partition. It is associated with all allowed t-tuples, including its ND factors. Thus, when a partition is aligned with one equivalence class, its ND strength is the lesser of $t + 1$ or $n_j + 1$. (The ND strength cannot exceed $n_j + 1$ by definition.) The following strength-2 design illustrates the repartitioning for the Child and Adult equivalence classes.

In this design, the Medicare equivalence classes do not have their own partitions. According to C5 for $t = 2$, the ND subarray of Weight, Height, Sex, Intake, and Medicare factors has ND strength 2, so all allowed values of the Medicare ND factors can be associated with each equivalence class in one partition. The choice of an Age value of 65 or older in an Adult partition can insure the coverage.

TABLE X. STRENGTH-4 TEST CASES AND CORRESPONDING EQUIVALENCE CLASS FACTOR VALUES

	Input factors					Equivalence class factors			An expected result
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*	*Child*	*Adult*	*BMI*
1	19	131	64	female	2000	no	girl	no	22.5
2	19	131	64	male	3000	no	boy	no	22.5
3	19	131	71	female	3000	no	girl	no	18.3
4	19	131	71	male	2000	no	boy	no	18.3
5	19	180	64	female	3000	no	girl	no	30.9
6	19	180	64	male	2000	no	boy	no	30.9
7	19	180	71	female	2000	no	girl	no	25.1
8	19	180	71	male	3000	no	boy	no	25.1
9	42	131	64	female	3000	no	no	normal	22.5
10	42	131	64	male	2000	no	no	normal	22.5
11	42	131	71	female	2000	no	no	underweight	18.3
12	42	131	71	male	3000	no	no	underweight	18.3
13	42	180	64	female	2000	no	no	obese	30.9
14	42	180	64	male	3000	no	no	obese	30.9
15	42	180	71	female	3000	no	no	overweight	25.1
16	42	180	71	male	2000	no	no	overweight	25.1
17	67	131	64	female	2000	yes	no	normal	22.5
18	67	131	64	male	3000	yes	no	normal	22.5
19	67	131	71	female	3000	yes	no	underweight	18.3
20	67	131	71	male	2000	yes	no	underweight	18.3
21	67	180	64	female	3000	yes	no	obese	30.9
22	67	180	64	male	2000	yes	no	obese	30.9
23	67	180	71	female	2000	yes	no	overweight	25.1
24	67	180	71	male	3000	yes	no	overweight	25.1

For the Child equivalence classes, C5 shows that the ND subarray of Weight, Height, Intake, and Child factors has ND strength 1. I.e. the Child ND factor values might not be associated with each equivalence class in one partition. Including partitions for the girl and boy equivalence classes can insure this association. Since each class is a single-value factor, it is associated with all allowed pairs of ND factors. So the Child ND subarray has ND strength 3 in each partition. Similar reasoning suggests separate partitions for the Adult equivalence classes, underweight, normal, overweight, and obese.

Table XI shows the six partitions to generate test cases in this example. As before, it has the five input factors from R1 and the three equivalence class factors from R3, R4, and R5. The six partitions are for the Child girl and boy classes; and for the Adult underweight, normal, overweight, and obese classes. Age, Weight, and Height values are chosen so that expected results conform to their equivalence classes, and boundaries between classes can be verified[4].

Table XII lists the test cases for the partitions shown in Table XI. The input factor values cover with strength 2. The corresponding fixed values for the equivalence class factors are given. The expected BMI is shown also.

In this design each Medicare equivalence class is expected in each of the third through sixth partitions. In each of these test arrays, in accordance with C5, each Medicare ND factor (Weight, Height, Sex, Intake) has its values associated with each equivalence class.

In the first partition the Child class factor has a single value, girl. This value is associated with all allowed pairs of the independent factors (Age, Weight, Height, Sex, Intake). Similarly, in the second partition the Child class boy is associated with all allowed pairs.

In each of the remaining partitions an Adult equivalence class factor is a single-value factor. Thus it is associated with all allowed pairs in its respective partition's test array.

Finally, the use of multiple partitions in this design offers additional freedom to choose values that verify boundaries between equivalence classes. Tests for the following boundaries are included in Table XII.

- Medicare: lower Age
- Child: upper Age
- Adult: lower Age
- Adult, underweight: upper BMI
- Adult, normal: lower and upper BMI
- Adult, overweight: lower and upper BMI
- Adult, obese: lower BMI

For example, at Height 70, test case uw2 with Weight 128 is in the underweight class; test case nm1 with Weight 129 is in the normal class. At Height 66, test case nm2 with Weight 154 is in the normal class; test case ow1 with Weight 155 is in the overweight class.

[4] This example uses integer inputs; BMI applications often use fractions for Age, Weight and Height.

TABLE XI. PARTITIONS WITH FIXED INPUT VALUES AND FUNCTIONALLY DEPENDENT EQUIVALENCE CLASS FACTORS

Body Mass Index Report Application
Age
Weight
Height
Sex
Intake
Medicare equivalence class
Child equivalence class
Adult equivalence class
#gl Child girl
15 19
115 135
63 67
female
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)
#by Child boy
15 19
125 168
66 71
male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)
#uw Adult underweight
20 64 65
118 128
70 72
female male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)
#nm Adult normal
20 64 65
129 154
66 70
female male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)
#ow Adult overweight
20 64 65
155 174
64 66
female male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)
#ob Adult obese
20 64 65
175 211
61 64
female male
2000 3000
Medicare_report(Age)
Child_report(Age,Sex)
Adult_report(Age,Weight,Height)

TABLE XII. STRENGTH-2 TEST CASES AND CORRESPONDING EQUIVALENCE CLASS FACTOR VALUES

	Input factors					Equivalence class factors			An expected result
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*	*Child*	*Adult*	*BMI*
gl1	15	115	63	female	2000	no	girl	no	20.4
gl2	19	135	67	female	2000	no	girl	no	21.1
gl3	19	135	63	female	3000	no	girl	no	23.9
gl4	19	115	67	female	3000	no	girl	no	18.0
gl5	15	135	67	female	3000	no	girl	no	21.1
by1	15	125	66	male	2000	no	boy	no	20.2
by2	19	168	71	male	2000	no	boy	no	23.4
by3	19	168	66	male	3000	no	boy	no	27.1
by4	19	125	71	male	3000	no	boy	no	17.4
by5	15	168	71	male	3000	no	boy	no	23.4
uw1	65	118	72	male	3000	yes	no	underweight	16.0
uw2	65	128	70	female	2000	yes	no	underweight	18.4
uw3	64	128	72	male	2000	no	no	underweight	17.4
uw4	20	118	70	male	2000	no	no	underweight	16.9
uw5	64	118	70	female	3000	no	no	underweight	16.9
uw6	20	128	72	female	3000	no	no	underweight	17.4
nm1	65	129	70	male	3000	yes	no	normal	18.5
nm2	65	154	66	female	2000	yes	no	normal	24.9
nm3	64	154	70	male	2000	no	no	normal	22.1
nm4	20	129	66	male	2000	no	no	normal	20.8
nm5	64	129	66	female	3000	no	no	normal	20.8
nm6	20	154	70	female	3000	no	no	normal	22.1
ow1	65	155	66	male	3000	yes	no	overweight	25.0
ow2	65	174	64	female	2000	yes	no	overweight	29.9
ow3	64	174	66	male	2000	no	no	overweight	28.1
ow4	20	155	64	male	2000	no	no	overweight	26.6
ow5	64	155	64	female	3000	no	no	overweight	26.6
ow6	20	174	66	female	3000	no	no	overweight	28.1
ob1	65	175	64	male	3000	yes	no	obese	30.0
ob2	65	211	61	female	2000	yes	no	obese	39.9
ob3	64	211	64	male	2000	no	no	obese	36.2
ob4	20	175	61	male	2000	no	no	obese	33.1
ob5	64	175	61	female	3000	no	no	obese	33.1
ob6	20	211	64	female	3000	no	no	obese	36.2

V. DISCUSSION

The FD coverage rules and their application to FD equivalence class factors formulate how constraints from FD relations can alter coverage from the nominal generation strength *t*. Of particular interest is C5, the nondeterminant strength rule, because it explains how equivalence class coverage might be less than *t*. Specifically each equivalence class having l_j determinant factors is associated with at least $(t–l_j)$-tuples of ND factors. Two types of responses to this reduction are suggested.

The first is to increase the strength of the test case generation from *t* to *t'*. Its effect is to associate each equivalence class with ND $(t'–l_j)$-tuples. When l_j is large, an even larger strength is indicated.

The second response is to align test case generation partitions with their equivalence classes as needed for the desired coverage. This repartitioning associates the equivalence class with ND *t*-tuples.

The coverage difference between the two partitioning methods can be significant. According to C5, in the strength-3 design the Adult equivalence classes are not expected to be associated with all ND values. The test cases of Table IX show that only 50% of the *values* are covered in two of the classes. C5 implies that the strength-4 design raises the minimum ND coverage to all *values* with each class. In contrast, all *pairs* of ND factors can be expected with each repartitioned class in the strength-2 design.

Table XIII shows the ND strength achieved by the three BMI example designs. All of the listed values are consistent with coverage rule C5. The strength-4 design of Table X achieves ND strength 3 for the Adult factor, which exceeds the minimum given by C5. The strength-2 design has ND strength 3 for the Child and Adult factors because their partitions are aligned with their equivalence classes.

The point of applying the coverage rules to FD equivalence class factors is to enable testers to make informed decisions about designs. And the fact that these choices can be made before the generation of test cases can make the design process more efficient.

TABLE XIII. NONDETERMINANT STRENGTH FOR EXAMPLE DESIGNS

Design	Equivalence class factors		
	Medicare	*Child*	*Adult*
Table IX	3	2	1
Table X	4	3	3
Table XII, all partitions	2	3	3

However the choices can be helpful later in the process also. If the Adult report fails whenever the inputs are for an underweight male, the strength-3 test cases of Table IX cannot detect that failure. When analysis shows that a combination of four inputs, Age, Weight, Height, and Sex induce the problem, responses can include either of the two described above:

- Increasing the strength to 4 using appropriate values in a single-partition design (as in Table X)
- Choosing a strength-2 design with appropriate values and repartitioning (as in Table XII)

Both responses increase the ND strength for the Adult equivalence class factor to 3, but there are additional choices. Generation strength 3 could be retained. With appropriate values and repartitioning for the Adult classes, the ND strength would increase to 3 (which is $n_j + 1$, the maximum value for the Adult factor). Moreover, different choices can be made for different equivalence class factors.

To make use of these choices test case generation tools need to support the use of multiple partitions having different factor values and constraints, to align partitions with equivalence classes as needed. Without this capability the only recourse appears to be an escalation of strength, yielding test plans of increasing cost to execute. Of course the test oracle problem—determination of expected results—contributes greatly to the challenge [2,3]. However, when information about equivalence classes and expected results is available, test design tools and processes should make the best use of it.

Finally the formulation of the equivalence class coverage rules suggests new combinatorial classifications may be useful. Empirical research examining the effectiveness of designs using different ND strengths may clarify the outcomes of test design choices.

REFERENCES

[1] Codd, E. F. (1972). Further normalization of the data base relational model. In *Data Base Systems* (pp. 33-64). Upper Saddle River, NJ: Prentice Hall.

[2] IBM. (2014). *The 3rd International Workshop on Combinatorial Testing (IWCT 2014).* Retrieved February 12, 2014, from Research.ibm.com: https://www.research.ibm.com/haifa/Workshops/iwct2014.

[3] Kuhn, D. R., Kacker, R. N., & Lei, Y. (2013). Combinatorial Methods in Testing. In D. R. Kuhn, R. N. Kacker, & Y. Lei, *Introduction to Combinatorial Testing* (pp. 1-20). Boca Raton, FL: CRC Press.

[4] Tatsumi, K. (1987). Test case design support system. *Proceedings of the International Conference on Quality Control (ICQC'87)*, (pp. 615-620). Tokyo.

[5] Sherwood, G. (1994). Efficient testing of factor combinations. *Proceedings of the Third International Conference on Software Testing, Analysis, and Review*, (pp. 151-166). Washington, DC.

[6] Czerwonka, J. (2006). Pairwise testing in the real world: Practical extensions to test-case scenarios. *Proceedings of the Twenty-fourth Annual Pacific Northwest Software Quality Conference*, (pp. 419-430). Portland, OR.

[7] Sherwood, G. (2013). Managing System State. In D. R. Kuhn, R. N. Kacker, & Y. Lei, *Introduction to Combinatorial Testing* (pp. 113-141). Boca Raton FL: CRC Press.

[8] Testcover.com, LLC. (2008). *Submitting Test Case Generator Requests.* Retrieved January 6, 2014, from Testcover.com: http://testcover.com/pub/how2in.php.

[9] Testcover.com, LLC. (2013). *Fixed Values Procedure.* Retrieved January 6, 2014, from Testcover.com: http://www.testcover.com/pub/fvproc.php.