Test design automation: equivalence classes, boundaries, edges and corner cases

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

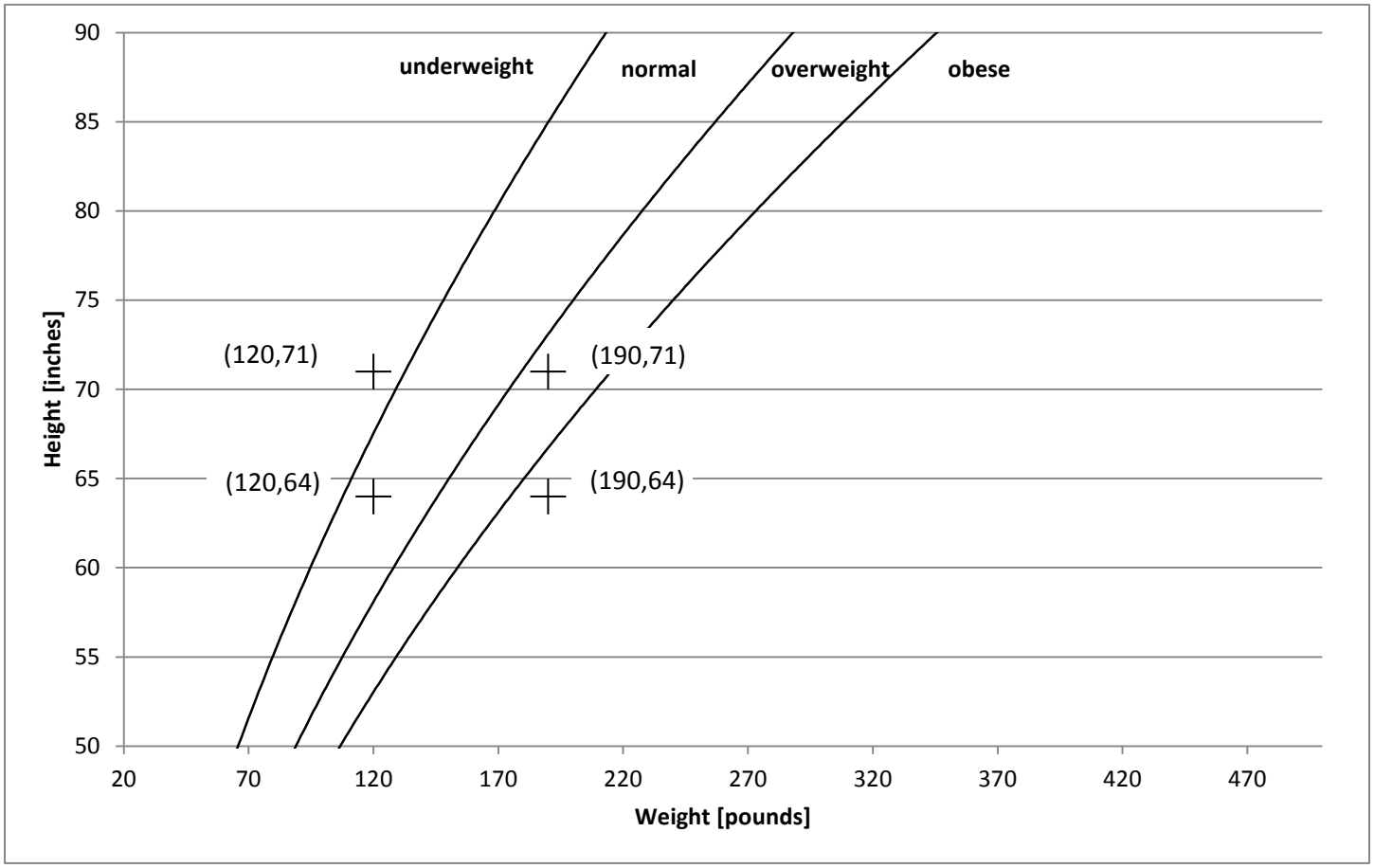
Abstract

An embedded functions feature is under development for the Testcover.com combinatorial test design service. The feature allows functionally dependent relations among test factors to be described as functions in a general purpose programming language. Initial results demonstrate usability improvements enabling pairwise test designs to meet several objectives: Cover equivalence classes of expected results; pair equivalence classes with nondeterminant factor values; verify univariate and multivariate equivalence class boundaries; verify corners among intersecting boundaries and edges. Methods to meet these objectives are described in examples with specified test cases. The results suggest use of embedded functions can improve automation, accuracy, control and flexibility in the test design process.

1. Introduction

Software testing verifies that programs work as expected. Typically there are too many possible test results to try all of them, so testers divide the expected results into equivalence classes (ECs). An *equivalence class* includes the combinations of test factor values leading to one class of expected results. For example, in a Body Mass Index (BMI) report application, the BMI is computed from a person's weight and height. A different type of report is generated based on the person's BMI classification: underweight, normal, overweight or obese. Each report for a BMI classification is processed the same way, so its set of expected results forms an equivalence class. An essential objective here is to verify all four of the report types. And a test design that reaches (or covers) their equivalence classes requires corresponding combinations of the weight and height inputs.

Figure 1. Height-weight values for BMI classifications



A pairwise test design can be used to verify the equivalence classes. Pairwise designs have the property that each value of each test factor (e.g. weight) is contained in a test case with each value of every other factor (e.g. height). So we need to choose values for the weight and height inputs to reach the four equivalence classes. Figure 1 shows how to do this using a graph of the BMI classifications. We choose two weights, 120 pounds and 190 pounds, and two heights, 64 inches and 71 inches, so that each weight-height pair is in a different class. The pairwise design will have a test case for each of these pairs, so the four classes should be covered.

The application has a total of five inputs for three types of reports, as specified in Table I. There is a Medicare report generated when the person's age is over 65 years. There is a child report when age is under 20. The BMI classifications are not used for children because they are still growing. Instead the BMI percentile for the child's age is reported. And there are separate tables for girls and boys because they tend to grow at different rates. Finally there are the adult reports corresponding to the BMI classifications. We choose age values of 15, 42 and 67 to verify the three report types. We choose sex values of female and male to verify both child report classes. And we choose 2000 and 3000 for calorie intake values.

Table I. BMI report requirements

R1.	The listed input data will be stored in the patient database table.		
	a.	Age in years	(integer) $2 \le Age < 130$
	b.	Weight in pounds	(integer) $20 \le Weight < 500$
	c.	Height in inches	(integer) $30 \le Height < 90$
	d.	Sex	(female, male)
	e.	Intake in kilocalories per day	(integer) $1 \le Intake < 10000$
R2.	The BMI will be calculated (in kilograms per meter squared) as $703.06957964\ Weight / Height^2$ and stored in the patient database table.		
R3.	If Age is 65 years or older, the Medicare report will be generated.		
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	$18.5 \le BMI < 25.0$
	c.	Overweight	$25.0 \le BMI < 30.0$
	d.	Obese	$30.0 \le BMI$

These factor values generate Test cases 1a using Testcover.com. Each test case lists the values for the five inputs, as well as the expected class of results for each report type (Medicare, child and adult). The *equivalence class factors* [1-3] do not affect the test case generation in this example. Their values are computed from *equivalence class functions* and are substituted into each test case after the test case is generated. Their purpose is to provide a check for the design, to show which classes are covered. Table II contains a listing of the functions, which are written in PHP [4].

Test cases 1a. Accidental equivalence class coverage

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [s]	***Child*** [s]	***Adult*** [s]
1	67	120	71	male	3000	yes	no	underweight
2	67	190	64	female	2000	yes	no	obese
3	42	190	71	male	2000	no	no	overweight
4	42	120	64	female	3000	no	no	normal
5	15	120	64	male	2000	no	boy	no
6	15	190	71	female	3000	no	girl	no

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Table II. Equivalence class functions

```
function Medicare_report($Age) {               /* Medicare_report equivalence class function */
  if($Age>=65) return('yes');                  /* Medicare_report is expected result */
  if($Age>0) return('no');                     /* Medicare_report is not expected result */
}

function Child_report($Age,$Sex) {             /* Child_report equivalence class function */
  if($Age>0) {
    switch($Sex) {
      case 'female':
          if($Age<20) return('girl');          /* Child_report for girl is expected result */
          else return('no');                   /* Child_report is not expected result */
      case 'male':
        if($Age<20) return('boy');             /* Child_report for boy is expected result */
         else return('no');                    /* Child_report is not expected result */
    }
  }
}

function Adult_report($Age,$Weight,$Height) {  /* Adult_report equivalence class function */
  $bmi_value=BMI($Weight,$Height);             /* Use BMI function to get BMI value */
  if($Age>=20&&$bmi_value>0) {
    if($bmi_value>=30) return('obese');        /* Adult_report for obese is expected result */
    if($bmi_value>=25) return('overweight');   /* Adult_report for overweight is expected result */
    if($bmi_value>=18.5) return('normal');     /* Adult_report for normal is expected result */
    if($bmi_value>0) return('underweight');    /* Adult_report for underweight is expected result */
  }
  if($Age>0&&$bmi_value>0) return('no');       /* Adult_report is not expected result */
}

function BMI($Weight,$Height) {                /* BMI function for Adult_report */
  if($Weight>0&&$Height>0) {
    $bmi_value=703.06957964*$Weight/$Height/$Height;
    return($bmi_value);                        /* return valid BMI value */
  }
}
```

The six test cases include all pairs of input factor values, and they cover all 10 equivalence classes. There are two Medicare report classes (yes, no), three child report classes (boy, girl, no) and five adult report classes (underweight, normal, overweight, obese, no). This is a successful design: It can verify all the equivalence classes. But it happened accidentally. If we order the age values as 42, 67, 15 rather than 15, 42, 67, we get Test cases 1b. This second set of test cases covers all pairs of input factor values, but it does not cover all the equivalence classes. The adult classes underweight and obese are missing.

Test cases 1b. Incomplete equivalence class coverage

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [s]	***Child*** [s]	***Adult*** [s]
1	15	120	71	male	3000	no	boy	no
2	15	190	64	female	2000	no	girl	no
3	67	190	71	male	2000	yes	no	overweight
4	67	120	64	female	3000	yes	no	normal
5	42	120	64	male	2000	no	no	normal
6	42	190	71	female	3000	no	no	overweight

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

The problem with Test cases 1b is that the weight-height pairs for underweight (120,71) and obese (190,64) occur in test cases with age 15. These test cases are expected to generate the boy and girl reports, not the underweight and obese reports for an adult. This design does not cover all the equivalence classes, but there *are* ways to insure equivalence class coverage.

Two current methods, using partitioning and higher strength, are described in Sections 2 and 3. The methods have a tradeoff: test design analysis work vs. a larger number of test cases. When we consider realistic systems larger and more complex than this example, both the analysis work and the number of test cases can be significant challenges

for the responsible practitioners. And we have decades of experience suggesting incomplete coverage too often is the result [5].

Reference [3] and subsequent investigation suggest that automated evaluation of embedded functions can reduce the manual analysis work and yield pairwise designs that meet various test objectives. Section 4 introduces such designs to verify equivalence classes and their age boundary values (BVs). The analysis is limited to selecting test factors and values, and defining the equivalence class functions. Section 5 shows how BMI boundaries can be verified using boundary value factors and inverting the BMI function. And when all pairs of age and BMI boundary values are covered, we can verify all the resulting corner values (or not, depending on test objectives). Section 6 adds edges (minimum and maximum input values) into the designs, for use as needed. Finally Section 7 outlines the test design implications of embedded functions. Table III provides a summary of all the examples, with the objectives they meet and the methods they use.

Generally the need is to accommodate system complexity, size and diverse objectives in test designs. Embedded functions offer automation improvements for the test design process. These preliminary findings suggest improved accuracy, control and flexibility for designs, which can advance test efficiency and quality.

Table III. Examples

Objectives							Methods					Example		Test cases	Partitions	Strength	Input factors	EC factors	BV factors
CEC											1a	Accidental EC coverage	6	1	2	5^f	3^s	0
											1b	Incomplete EC coverage	6	1	2	5^f	3^s	0
CEC							PRT				2a	EC coverage using partitioning	11	2	2	5^f	3^s	0
CEC		UVB					PRT				2b	Coverage of univariate EC boundaries using partitioning	11	2	2	5^f	3^s	0
CEC	PND	UVB	MVB				PRT				2c	Coverage of EC boundaries using partitioning [1]	34	6	2	5^f	3^s	0
CEC								HST			3a	EC coverage using strength 3	18	1	3	5^f	3^s	0
CEC		UVB						HST			3b	Coverage of univariate EC boundaries using strength 3	24	1	3	5^f	3^s	0
CEC	PND							HST			3c	EC-nondeterminant pairing using strength 4 [1]	24	1	4	5^f	3^s	0
CEC	PND								ECF		4a	EC coverage using EC factors	14	1	2	5^f	3^c	0
CEC	PND	UVB							ECF		4b	Coverage of univariate EC boundaries using EC factors	18	1	2	5^f	3^c	0
CEC	PND	UVB	MVB		CRN	PBE			ECF	BVF	5a	Coverage of EC boundaries & their corners using EC & BV factors with dependent weight values	60	1	2	$4^f 1^c$	3^c	2^f
CEC		UVB	MVB		CRN	PBE				BVF	5b	Coverage of EC boundaries & their corners using BV factors with dependent weight values	48	1	2	$4^f 1^c$	3^s	2^f
CEC		UVB	MVB			PBE			ECF	BVF	5c	EC-boundary factor pairing	20	1	2	$4^f 1^s$	3^c	2^f
CEC		UVB	MVB			PBE				BVF	5d	Boundary factor pairing	12	1	2	$4^f 1^s$	3^s	2^f
				EDG	CRN						6a	Coverage of edges & their corners	6	1	2	5^f	3^s	0
CEC	PND	UVB		EDG	CRN				ECF		6b	Coverage of univariate EC boundaries, edges & their corners	34	1	2	5^f	3^c	0
CEC		UVB	MVB	EDG		PBE			ECF	BVF	6c	EC boundary & edge factor pairing	33	1	2	$4^f 1^{fs}$	$2^c 1^{cc}$	2^f

CEC	Cover equivalence classes of expected results using representative values
PND	Pair equivalence classes with nondeterminant factor values
UBV	Verify univariate (input, nonedge) equivalence class boundaries
MBV	Verify multivariate equivalence class boundaries
EDG	Verify input edges (min/max input values)
CRN	Verify all boundary/edge corner points
PBE	Pair all boundary/edge factor values
PRT	Use partitions to separate equivalence classes
HST	Use higher strength (cover *t*-tuples with *t* > 2)
ECF	Use equivalence class factors (with values from combination equivalence class functions)
BVF	Use boundary value factors (with fixed values) to evaluate boundary value functions for dependent factors

Number of factors using:

c	1 combination function
cc	2 combination functions
f	fixed values
fs	fixed values and 1 substitution function
s	1 substitution function

2. Partitioning

Whether or not the BMI report application reaches a class of expected results depends on the value of one or more inputs (test factors). The equivalence class factors are *dependent* on the input factors. And the *independent* input factors can be either *determinant* or *nondeterminant*, according to the type of report. For example, the Medicare report depends only on age. Age is the determinant factor for the Medicare report classes; all the other factors are nondeterminant. Table II shows that the Medicare, child and adult report classes are determined respectively by 1, 2 and 3 determinant factors, because each function requires the corresponding number of test factor arguments. The child report classes are determined by age and sex; the adult report classes are determined by age, weight and height. The intake factor is nondeterminant for all three of the equivalence class factors.

When an equivalence class factor has 3 or more determinant factors, a pairwise (strength 2) test design may or may not reach all of its classes. Test cases 1a and 1b illustrate both of these outcomes. A pairwise design includes all pairs of factor values, but it might not cover all triples. However, we can insure coverage of all the equivalence classes by using partitions. In this paper a partition refers to one test case generation instance, which may correspond to one or more equivalence classes. A *partition* includes the allowed combinations of factor values for the test case generation instance [1].

Test cases 2a cover all the equivalence classes of the example using 11 test cases in 2 partitions. The partitions are indicated by different test case number series (ma or c). The first partition contains the age inputs for the Medicare and adult reports; the other factor values are as given previously. The second partition uses age 15 for the child report classes. This design covers all the adult classes because (1) both ages in the first partition cover the 4 classes *with* adult reports, and (2) the age in the second partition covers the class *without* an adult report.

Test cases 2a. Equivalence class coverage using partitioning

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [s]	***Child*** [s]	***Adult*** [s]
ma1	42	120	64	female	2000	no	no	normal
ma2	67	190	71	male	3000	yes	no	overweight
ma3	67	190	71	female	2000	yes	no	overweight
ma4	67	120	64	male	3000	yes	no	normal
ma5	42	190	64	male	2000	no	no	obese
ma6	42	120	71	female	3000	no	no	underweight
c1	15	120	64	female	2000	no	girl	no
c2	15	190	71	male	2000	no	boy	no
c3	15	190	71	female	3000	no	girl	no
c4	15	190	64	male	3000	no	boy	no
c5	15	120	71	male	3000	no	boy	no

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Another common test objective is equivalence class boundary verification. Test factors can be ordered (ranked) or enumerated. Age and weight are examples of factors whose values can be ordered; sex is an example of a factor with enumerated values. When an equivalence class has an ordered determinant factor, its value can establish a boundary for the class. Verification of the boundary values can provide additional information about the correctness of the implementation, i.e. whether the program logic separates the classes correctly.

The age factor provides boundaries for all three of the equivalence class factors. For the Medicare report, age 65 is the minimum age for the yes class; age 64 is the maximum age for the no class. For the child report, age 20 is the minimum age for the no class; age 19 is the maximum age for the girl and boy classes. Ages 19 and 20 are the boundary values for the adult report also.

Each of these boundaries depends on one variable, age, so they are *univariate* boundaries. Generating test cases to verify univariate boundaries is straightforward: We replace the determinant factor values with the minimum and maximum values for each class. Test cases 2b illustrate how this can be done. For the Medicare report, age 65, the

minimum for the yes class replaces age 67; age 64, the maximum age for the no class replaces age 42. For the child report, age 20, the minimum age for the no class replaces age 42; age 19, the maximum age for the girl and boy classes replaces age 15. For the adult report, ages 19 and 20 are already included for the child report. So the 3 age values of Test cases 2a are replaced by 4 values in Test cases 2b. The first (ma) partition uses ages 20, 64 and 65; the second (c) partition uses age 19. Test cases 2a happened to be generated in decreasing age order. The age boundaries are indicated by the two horizontal lines.

Test cases 2b. Coverage of univariate equivalence class boundaries using partitioning

Test case	Input factors					Equivalence class factors		
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*[s]	*Child*[s]	*Adult*[s]
ma1	65	120	71	male	3000	yes	no	underweight
ma2	65	190	64	female	2000	yes	no	obese
ma3	64	190	71	male	2000	no	no	overweight
ma4	64	120	64	female	3000	no	no	normal
ma5	20	120	64	male	2000	no	no	normal
ma6	20	190	71	female	3000	no	no	overweight
c1	19	120	64	female	2000	no	girl	no
c2	19	190	71	male	2000	no	boy	no
c3	19	190	71	female	3000	no	girl	no
c4	19	190	64	male	3000	no	boy	no
c5	19	120	71	male	3000	no	boy	no

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Three BMI values, 18.5, 25 and 30, are boundaries for the adult classes also. The BMI values depend on two input variables, weight and height, so the BMI boundaries are *multivariate*. We can verify multivariate boundaries also, but more partitions are needed. Six are used in Example 2c (Table III). Reference [1], Table XII shows the test design to verify the age and BMI boundaries.

Example 2c illustrates another possible test objective also. Each of the equivalence classes is reached according to its determinant input values. But reaching each class does not guarantee the class has a test case with each nondeterminant factor value. In Test cases 2b, case ma1 is the only case for the underweight class. There are no underweight test cases with sex female or with intake 2000. Similarly test case ma2 is the only case for the obese class. There are no obese test cases with sex male or with intake 3000. There is a similar limitation in Test cases 2a. If pairing equivalence classes with their nondeterminant factor values is a test objective, it can be done with partitioning. Example 2c (Table XII [1]) illustrates this point.

Partitioning offers a flexible way to insure coverage of equivalence classes and their boundary values. It can pair classes with their nondeterminant factor values as well. It does require an analysis to construct partitions to meet the test objectives. Automating some of this work can improve test design efficiency.

3. Higher strength

Thus far we have used pairwise (strength 2) designs to meet our test objectives. Alternatively we can use higher strength designs. A strength 3 design includes each 3-tuple of factor values in a test case; a strength 4 design includes each 4-tuple, etc. A strength 3 design includes every combination of the adult class determinant factor values (age, weight and height) in one of its test cases. So the adult equivalence classes are covered. No additional partitioning is needed. The Medicare and child classes are covered also. Test cases 3a illustrate the design, which uses the same factors and values as Example 2a.

Test cases 3a. Equivalence class coverage using strength 3

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [s]	***Child*** [s]	***Adult*** [s]
1	15	120	64	female	2000	no	girl	no
2	15	120	71	male	3000	no	boy	no
3	42	190	64	female	3000	no	no	obese
4	42	190	71	male	2000	no	no	overweight
5	67	120	64	male	2000	yes	no	normal
6	67	120	71	female	3000	yes	no	underweight
7	15	190	64	male	3000	no	boy	no
8	15	190	71	female	2000	no	girl	no
9	42	120	64	male	3000	no	no	normal
10	42	120	71	female	2000	no	no	underweight
11	67	190	64	female	2000	yes	no	obese
12	67	190	71	male	3000	yes	no	overweight
13	15	120	64	female	3000	no	girl	no
14	15	120	64	male	2000	no	boy	no
15	42	120	64	female	2000	no	no	normal
16	42	120	71	female	3000	no	no	underweight
17	67	120	64	female	3000	yes	no	normal
18	67	120	71	female	2000	yes	no	underweight

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Univariate boundary verification can be done with one strength 3 partition also. Test cases 3b illustrate a design using the same factor values as Example 2b. The 24 test cases were generated in the order of the test case numbers. Here they have been sorted by age to show the boundaries.

Test cases 3b. Coverage of univariate equivalence class boundaries using strength 3

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [s]	***Child*** [s]	***Adult*** [s]
7	65	190	64	male	3000	yes	no	obese
8	65	190	71	female	2000	yes	no	overweight
15	65	120	64	female	2000	yes	no	normal
16	65	120	71	male	3000	yes	no	underweight
23	65	120	64	female	3000	yes	no	normal
24	65	120	64	male	2000	yes	no	normal
5	64	120	64	male	2000	no	no	normal
6	64	120	71	female	3000	no	no	underweight
13	64	190	64	female	2000	no	no	obese
14	64	190	71	male	3000	no	no	overweight
21	64	120	64	female	3000	no	no	normal
22	64	120	71	female	2000	no	no	underweight
3	20	190	64	female	3000	no	no	obese
4	20	190	71	male	2000	no	no	overweight
11	20	120	64	male	3000	no	no	normal
12	20	120	71	female	2000	no	no	underweight
19	20	120	64	female	2000	no	no	normal
20	20	120	71	female	3000	no	no	underweight
1	19	120	64	female	2000	no	girl	no
2	19	120	71	male	3000	no	boy	no
9	19	190	64	male	2000	no	boy	no
10	19	190	71	female	3000	no	girl	no
17	19	120	64	female	3000	no	girl	no
18	19	120	71	female	2000	no	girl	no

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

As before, reaching each equivalence class does not guarantee the class has a test case with each nondeterminant factor value. In Test cases 3a, the adult classes underweight and obese do not have test cases with sex male; the overweight class does not have a test case with sex female. If pairing equivalence classes with their nondeterminant factor values is a test objective, it can be done using strength 4 in one partition. Example 3c (Table X [1]) shows the design.

Higher strength offers an alternate way to insure coverage of equivalence classes and univariate boundary values. It can pair classes with their nondeterminant factor values as well. It may require less analysis than partitioning to meet the test objectives. Typically the tradeoff is more test cases, leading to higher test execution costs.

4. Equivalence classes and univariate boundaries

The designs in this section use the equivalence class functions of Table II as embedded *combination functions* [2] rather than substitution functions. Equivalence class factors originally were intended as they are used above, as a tool to assess equivalence class coverage [1]. Comparisons of the number of determinant factors with the design strength enabled statements about which classes would be reached and how they would be associated with nondeterminant factor values. Equivalence class factors showed where partitioning could avoid incomplete coverage. And the assessments were based on the relations among the factors; a test case generation and analysis were not needed for the assessment.

Embedded combination functions [2] were a way to simplify and automate the specification of constraints among input factors. The idea was to describe dependent values of test factors as functions of their determinant factors. All combinations of the determinant factors would be evaluated to specify the allowed values of the dependent factor. The pairwise test case generation would use the dependent factor values with their associated determinant factor combinations. In this way constraints among input factors could be accommodated automatically.

When equivalence class factors are evaluated with combination functions, they automatically constrain the test cases to cover their allowed classes [3]. Equivalence class functions are evaluated for all combinations of input values. Each resulting class is a value for its equivalence class factor. All allowed values must occur in the design, so at least one of its determinant combinations must be present. Moreover, in the pairwise design, each equivalence class value will be paired with its nondeterminant factor values.

Test cases 4a illustrate equivalence class coverage and nondeterminant factor pairing, using equivalence class factors and combination functions. The pairwise design in one partition uses the same factors and values as Example 2a and Example 3a.

Test cases 4a. Equivalence class coverage using equivalence class factors

Test case	Input factors					Equivalence class factors		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare*** [c]	***Child*** [c]	***Adult*** [c]
1	42	190	71	female	2000	no	no	overweight
2	67	120	64	male	3000	yes	no	normal
3	15	120	64	male	2000	no	boy	no
4	15	190	64	female	3000	no	girl	no
5	67	120	71	female	2000	yes	no	underweight
6	42	190	64	male	3000	no	no	obese
7	67	190	71	male	3000	yes	no	overweight
8	42	120	71	male	3000	no	no	underweight
9	42	120	64	female	2000	no	no	normal
10	15	190	71	male	2000	no	boy	no
11	67	190	64	female	2000	yes	no	obese
12	15	120	64	female	2000	no	girl	no
13	15	190	64	male	3000	no	boy	no
14	15	190	71	female	2000	no	girl	no

c Factor is evaluated before test case generation using combination function.
All other factors are evaluated using fixed values.

All 10 classes of expected results are present in the design. Each adult class in which a report is expected occurs in two test cases which include both sex values and both intake values.

Univariate boundary verification can be done similarly. Test cases 4b illustrate a design using the same factor values as Example 2b and Example 3b. The 18 test cases were generated in the order of the test case numbers. Here they have been sorted by age to show the boundaries.

Test cases 4b. Coverage of univariate equivalence class boundaries using EC factors

Test case	Input factors					Equivalence class factors		
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*[c]	*Child*[c]	*Adult*[c]
2	65	120	64	male	3000	yes	no	normal
6	65	120	71	female	2000	yes	no	underweight
8	65	190	71	male	3000	yes	no	overweight
12	65	190	64	female	2000	yes	no	obese
4	64	190	64	female	3000	no	no	obese
7	64	120	71	male	2000	no	no	underweight
15	64	190	71	male	2000	no	no	overweight
16	64	120	64	male	2000	no	no	normal
1	20	190	71	female	2000	no	no	overweight
9	20	120	64	female	2000	no	no	normal
10	20	190	64	male	3000	no	no	obese
13	20	120	71	male	3000	no	no	underweight
3	19	120	64	male	2000	no	boy	no
5	19	120	71	female	3000	no	girl	no
11	19	190	71	male	2000	no	boy	no
14	19	120	64	female	2000	no	girl	no
17	19	190	64	male	3000	no	boy	no
18	19	190	71	female	2000	no	girl	no

c Factor is evaluated before test case generation using combination function.
All other factors are evaluated using fixed values.

5. Multivariate boundaries and corners

BMI boundary verification depends on two variables, weight and height, so it is more involved than verifying age boundaries. However introducing constraints from equivalence class factors enabled us to cover the classes. A similar plan using boundary value factors can enable boundary value coverage. In these designs the weight factor is dependent on height and the boundary value factors. A BMI boundary factor (with fixed values 18.5, 25 and 30) indicates the boundary to verify. An input limit factor (with fixed values min and max) indicates which BMI value will be verified. Weight and height have integer inputs, so the BMI value typically will not equal the boundary exactly. The input limit factor shows which BMI value to use: the minimum value greater than or equal to the boundary, or the maximum value less than the boundary. The dependent weight values are specified by the weight boundary function in Table IV.

Table IV. Boundary value functions

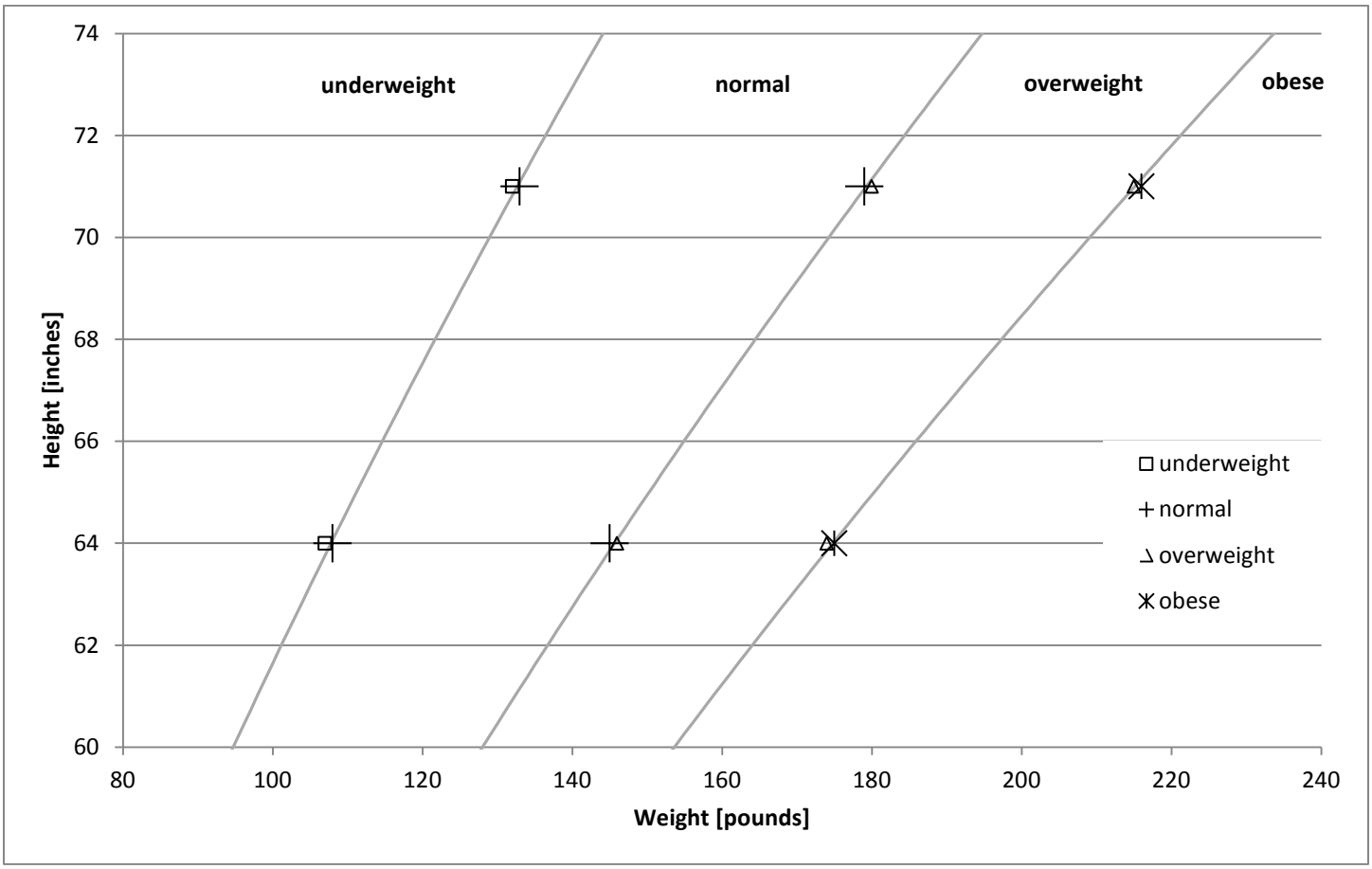
```
function Weight_boundary($Height,$BMI_boundary,$Input_limit) {     /* Weight boundary value function */
  if($Height>0&&$BMI_boundary>0) {
    $w_hi=ceil($Height*$Height*$BMI_boundary/703.06957964);        /* round up $w_hi so BMI >= $BMI_boundary */
    switch($Input_limit) {
      case 'min':
        return($w_hi);                                 /* return minimum integer value for higher BMI class */
      case 'max':
        $w_lo=$w_hi-1;
        return $w_lo;                                  /* return maximum integer value for lower BMI class */
    }
  }
}

function Adult_report_bv($Age,$BMI_boundary,$Input_limit) { /* Adult_report EC function for BMI boundaries*/
  if($Age>=20&&$BMI_boundary>0) {
    switch($Input_limit) {
      case 'min':
        if($BMI_boundary==30) return('obese');       /* Adult_report for obese is expected result */
        if($BMI_boundary==25) return('overweight');  /* Adult_report for overweight is expected result */
        if($BMI_boundary==18.5) return('normal');    /* Adult_report for normal is expected result */
      case 'max':
        if($BMI_boundary==30) return('overweight');  /* Adult_report for overweight is expected result */
        if($BMI_boundary==25) return('normal');      /* Adult_report for normal is expected result */
        if($BMI_boundary==18.5) return('underweight'); /* Adult_report for underweight is expected result */
    }
  }
  if($Age>0&&$BMI_boundary>0) return('no');          /* Adult_report is not expected result */
}
```

The adult report function (Table II) is dependent on the weight value and must be evaluated *after* the weight value is. Some of the test designs in this section use the weight boundary function as a substitution function, to be evaluated *after* test case generation. Some of the designs use the adult report function as a combination function, to be evaluated *before* test case generation. So the previous adult report function cannot be a combination function when the weight boundary function is a substitution function. To avoid any conflict the revised adult report function in Table IV is used for the designs in this section. Here the adult report class is determined directly by the age and boundary factors. The Medicare report and child report equivalence class functions used earlier (Table II) are reused here.

There are three BMI boundaries and two input limits, so the weight boundary function can return 3 · 2 = 6 values for each height input value. The four examples in this section all use the previous height values 64 and 71. Thus there are 6 · 2 = 12 allowed boundary value points (with 12 different weights), as shown in Figure 2. The number of test cases in each example varies, but they all project to these points on the weight-height plane.

Figure 2 BMI boundary values



In Figure 2 the two leftmost points at height 71 inches correspond to the input values for BMI boundary 18.5. The two weight-height pairs are (132,71), for the maximum underweight BMI, 18.41, and (133,71), for the minimum normal BMI, 18.55. These weight-height pairs appear as factor values in Example 5a, test cases 3 and 5, and elsewhere.

Example 5a contains the four age boundary values from the earlier examples, as well as the previous values for sex and intake. When the 12 weight values are paired with the four age values, we can expect 48 corner cases (test cases including both age and BMI boundary values). Example 5a has 60 test cases, as listed below.

Test cases 5a. Coverage of EC boundaries & their corners using EC & BV factors with dependent weight values

Test case	Input factors					EC factors			BV factors	
	Age	*Weight*[C]	*Height*	*Sex*	*Intake*	*Medicare*[C]	*Child*[C]	*Adult*[C]	*BMI*	*Limit*
1	20	180	71	female	2000	no	no	overweight	25	min
2	19	174	64	male	3000	no	boy	no	30	max
3	65	132	71	male	3000	yes	no	underweight	18.5	max
4	65	175	64	female	2000	yes	no	obese	30	min
5	19	133	71	female	2000	no	girl	no	18.5	min
6	64	145	64	male	2000	no	no	normal	25	max
7	64	216	71	female	3000	no	no	obese	30	min
8	20	107	64	female	2000	no	no	underweight	18.5	max
9	65	146	64	male	3000	yes	no	overweight	25	min
10	20	108	64	male	3000	no	no	normal	18.5	min
11	19	179	71	male	2000	no	boy	no	25	max
12	20	215	71	female	2000	no	no	overweight	30	max
13	65	179	71	female	3000	yes	no	normal	25	max
14	19	145	64	female	3000	no	girl	no	25	max
15	19	175	64	male	3000	no	boy	no	30	min
16	19	132	71	male	2000	no	boy	no	18.5	max
17	65	133	71	male	3000	yes	no	normal	18.5	min
18	65	174	64	female	2000	yes	no	overweight	30	max
19	19	146	64	female	2000	no	girl	no	25	min
20	19	180	71	male	3000	no	boy	no	25	min
21	19	216	71	male	2000	no	boy	no	30	min
22	19	215	71	male	3000	no	boy	no	30	max
23	19	107	64	male	3000	no	boy	no	18.5	max
24	19	108	64	female	2000	no	girl	no	18.5	min
25	64	132	71	female	3000	no	no	underweight	18.5	max
26	20	216	71	male	3000	no	no	obese	30	min
27	64	180	71	male	2000	no	no	overweight	25	min
28	65	180	71	female	2000	yes	no	overweight	25	min
29	19	216	71	female	2000	no	girl	no	30	min
30	65	216	71	female	2000	yes	no	obese	30	min
31	65	215	71	female	2000	yes	no	overweight	30	max
32	65	107	64	female	2000	yes	no	underweight	18.5	max
33	65	145	64	female	2000	yes	no	normal	25	max
34	65	108	64	female	2000	yes	no	normal	18.5	min
35	19	180	71	female	2000	no	girl	no	25	min
36	20	132	71	female	2000	no	no	underweight	18.5	max
37	19	132	71	female	2000	no	girl	no	18.5	max
38	20	179	71	female	2000	no	no	normal	25	max
39	64	179	71	male	2000	no	no	normal	25	max
40	19	179	71	female	2000	no	girl	no	25	max
41	64	215	71	male	2000	no	no	overweight	30	max
42	19	215	71	female	2000	no	girl	no	30	max
43	20	133	71	female	2000	no	no	normal	18.5	min
44	64	133	71	male	2000	no	no	normal	18.5	min
45	19	133	71	male	2000	no	boy	no	18.5	min
46	20	174	64	female	2000	no	no	overweight	30	max
47	64	174	64	male	2000	no	no	overweight	30	max
48	19	174	64	female	2000	no	girl	no	30	max
49	64	107	64	male	2000	no	no	underweight	18.5	max
50	19	107	64	female	2000	no	girl	no	18.5	max
51	20	146	64	female	2000	no	no	overweight	25	min
52	64	146	64	male	2000	no	no	overweight	25	min
53	19	146	64	male	2000	no	boy	no	25	min
54	20	145	64	female	2000	no	no	normal	25	max

Test cases 5a. Coverage of EC boundaries & their corners using EC & BV factors with dependent weight values

Test case	Input factors					EC factors			BV factors	
	Age	*Weight*[c]	*Height*	*Sex*	*Intake*	*Medicare*[c]	*Child*[c]	*Adult*[c]	*BMI*	*Limit*
55	19	145	64	male	2000	no	boy	no	25	max
56	20	175	64	female	2000	no	no	obese	30	min
57	64	175	64	male	2000	no	no	obese	30	min
58	19	175	64	female	2000	no	girl	no	30	min
59	64	108	64	male	2000	no	no	normal	18.5	min
60	19	108	64	male	2000	no	boy	no	18.5	min

c Factor is evaluated before test case generation using combination function.
All other factors are evaluated using fixed values.

The reason for the additional test cases can be seen in Corner map 5a, which organizes the test case numbers according to the age and BMI boundary values. The age boundaries are shown as horizontal lines increasing from bottom to top. The BMI boundaries are shown as vertical lines increasing from left to right. The shaded areas contain the test case numbers for the four corners at the boundary intersections. For example test case 3 has the minimum age for the Medicare report and the maximum BMI for the underweight class at height 71. Test cases 17, 25 and 44 correspond to the other 3 corners at the intersection. It is clear from the corner map that there are 12 additional cases for age 19: Both boy and girl reports appear at all 12 corners.

Corner map 5a.

		BMI	18.5		25		30	
Height	*Age*	*Limit*	max	min	max	min	max	min
71	**65**	min	3	17	13	28	31	30
	64	max	25	44	39	27	41	7
	20	min	36	43	38	1	12	26
	19	max	37[g] 16[b]	5[g] 45[b]	40[g] 11[b]	35[g] 20[b]	42[g] 22[b]	29[g] 21[b]
64	**65**	min	32	34	33	9	18	4
	64	max	49	59	6	52	47	57
	20	min	8	10	54	51	46	56
	19	max	50[g] 23[b]	24[g] 60[b]	14[g] 55[b]	19[g] 53[b]	48[g] 2[b]	58[g] 15[b]

b Expected result is in boy equivalence class.
g Expected result is in girl equivalence class.

This test design uses equivalence class factors with combination functions. So each class is paired with its nondeterminant factor values. There are 12 weight values paired with the boy and girl classes. These associations lead to the additional test cases.

We can relax this constraint by using substitution functions for the equivalence class factors. The resulting all-corners design is shown in Test cases 5b.

Test cases 5b. Coverage of EC boundaries & their corners using BV factors with dependent weight values

Test case	Input factors					EC factors			BV factors	
	Age	*Weight*[c]	*Height*	*Sex*	*Intake*	*Medicare*[s]	*Child*[s]	*Adult*[s]	*BMI*	*Limit*
1	19	180	71	female	2000	no	girl	no	25	min
2	65	174	64	male	3000	yes	no	overweight	30	max
3	20	132	71	male	2000	no	no	underweight	18.5	max
4	64	108	64	female	3000	no	no	normal	18.5	min
5	20	216	71	female	3000	no	no	obese	30	min
6	19	145	64	male	3000	no	boy	no	25	max
7	64	215	71	male	2000	no	no	overweight	30	max
8	65	179	71	female	2000	yes	no	normal	25	max
9	20	146	64	male	2000	no	no	overweight	25	min
10	65	133	71	female	2000	yes	no	normal	18.5	min
11	19	107	64	female	2000	no	girl	no	18.5	max
12	19	175	64	female	2000	no	girl	no	30	min
13	64	146	64	female	3000	no	no	overweight	25	min
14	65	180	71	male	3000	yes	no	overweight	25	min
15	64	132	71	female	3000	no	no	underweight	18.5	max
16	19	179	71	male	3000	no	boy	no	25	max
17	64	216	71	male	2000	no	no	obese	30	min
18	20	215	71	female	3000	no	no	overweight	30	max
19	19	133	71	male	3000	no	boy	no	18.5	min
20	19	174	64	female	2000	no	girl	no	30	max
21	65	107	64	male	3000	yes	no	underweight	18.5	max
22	65	145	64	female	2000	yes	no	normal	25	max
23	65	175	64	male	3000	yes	no	obese	30	min
24	20	108	64	male	2000	no	no	normal	18.5	min
25	20	180	71	male	2000	no	no	overweight	25	min
26	64	180	71	female	3000	no	no	overweight	25	min
27	19	132	71	female	2000	no	girl	no	18.5	max
28	65	132	71	female	2000	yes	no	underweight	18.5	max
29	20	179	71	male	2000	no	no	normal	25	max
30	64	179	71	female	3000	no	no	normal	25	max
31	19	216	71	female	2000	no	girl	no	30	min
32	65	216	71	female	2000	yes	no	obese	30	min
33	19	215	71	female	2000	no	girl	no	30	max
34	65	215	71	female	2000	yes	no	overweight	30	max
35	20	133	71	male	2000	no	no	normal	18.5	min
36	64	133	71	female	3000	no	no	normal	18.5	min
37	20	174	64	male	2000	no	no	overweight	30	max
38	64	174	64	female	3000	no	no	overweight	30	max
39	20	107	64	male	2000	no	no	underweight	18.5	max
40	64	107	64	female	3000	no	no	underweight	18.5	max
41	19	146	64	female	2000	no	girl	no	25	min
42	65	146	64	female	2000	yes	no	overweight	25	min
43	20	145	64	male	2000	no	no	normal	25	max
44	64	145	64	female	3000	no	no	normal	25	max
45	20	175	64	male	2000	no	no	obese	30	min
46	64	175	64	female	3000	no	no	obese	30	min
47	19	108	64	female	2000	no	girl	no	18.5	min
48	65	108	64	female	2000	yes	no	normal	18.5	min

c Factor is evaluated before test case generation using combination function.

s Factor is evaluated after test case generation using substitution function.

All other factors are evaluated using fixed values.

Corner map 5b shows that one test case appears in each of the 48 corners. One test case number appears in each shaded corner.

Corner map 5b.

		BMI	18.5		25		30	
Height	***Age***	***Limit***	max	min	max	min	max	min
71	**65**	min	28	10	8	14	34	32
	64	max	15	36	30	26	7	17
	20	min	3	35	29	25	18	5
	19	max	27[g]	19[b]	16[b]	1[g]	33[g]	31[g]
64	**65**	min	21	48	22	42	2	23
	64	max	40	4	44	13	38	46
	20	min	39	24	43	9	37	45
	19	max	11[g]	47[g]	6[b]	41[g]	20[g]	12[g]

b Expected result is in boy equivalence class.
g Expected result is in girl equivalence class.

The numbers of test cases for Examples 5a and 5b can be reduced to 30 and 24 respectively by using one height value rather than two. The smaller designs still cover all corners, but there are half as many corners.

To cover some corners but not all of them, we can use a substitution function for the weight boundary values (Table IV) instead of the combination function. In such a design the weight factor is considered as a single fixed value during test case generation. After the test cases are generated, the appropriate weight input value is substituted into each test case. Example 5c uses combination functions for the equivalence class factors. Thus, all pairs of values will be covered for the input factors (except weight), for the equivalence class factors and for the boundary value factors. Test cases 5c list the 20 test cases for this equivalence class-boundary value pairing design. The BMI-age corners are graphed in Figure 3, and Corner map 5c relates the corners to the test cases.

Test cases 5c. Equivalence class-boundary factor pairing

Test case	**Input factors**					**Equivalence class factors**			**Boundary factors**	
	Age	***Weight***[s]	***Height***	***Sex***	***Intake***	***Medicare***[c]	***Child***[c]	***Adult***[c]	***BMI***	***Limit***
1	64	174	64	female	2000	no	no	overweight	30	max
2	19	133	71	male	3000	no	boy	no	18.5	min
3	65	179	71	male	2000	yes	no	normal	25	max
4	65	216	71	female	3000	yes	no	obese	30	min
5	19	146	64	female	2000	no	girl	no	25	min
6	20	107	64	female	2000	no	no	underweight	18.5	max
7	20	146	64	male	3000	no	no	overweight	25	min
8	64	133	71	female	3000	no	no	normal	18.5	min
9	19	174	64	male	3000	no	boy	no	30	max
10	65	107	64	male	3000	yes	no	underweight	18.5	max
11	20	175	64	female	2000	no	no	obese	30	min
12	64	180	71	male	2000	no	no	overweight	25	min
13	19	174	64	female	3000	no	girl	no	30	max
14	20	145	64	female	2000	no	no	normal	25	max
15	64	216	71	male	2000	no	no	obese	30	min
16	64	132	71	male	2000	no	no	underweight	18.5	max
17	19	146	64	male	2000	no	boy	no	25	min
18	19	133	71	female	3000	no	girl	no	18.5	min
19	65	174	64	female	2000	yes	no	overweight	30	max
20	20	215	71	male	2000	no	no	overweight	30	max

c Factor is evaluated before test case generation using combination function.
s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Figure 3 BMI-age corners (5c)

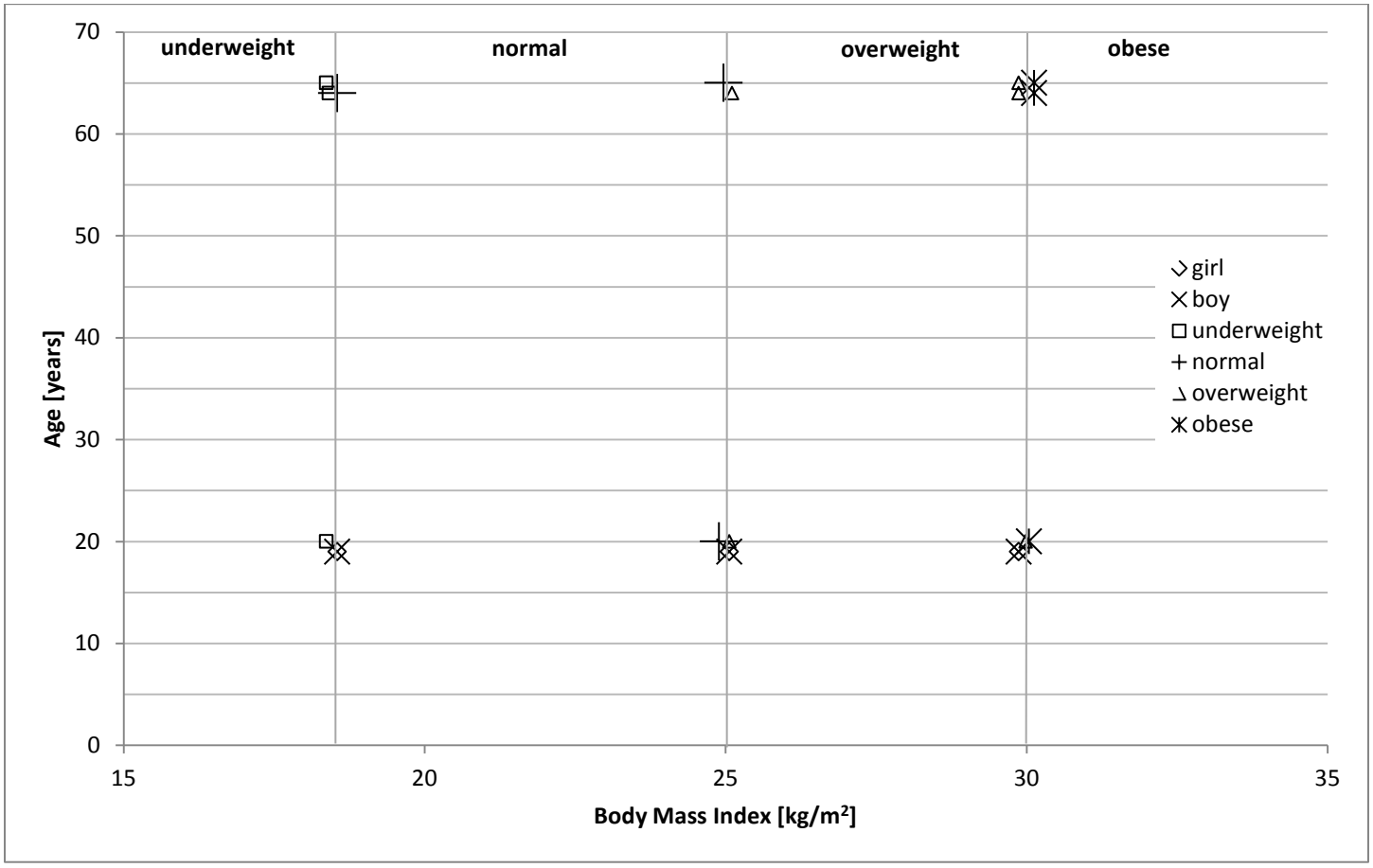


Figure 3 shows that at the boundary intersection of BMI 18.5 and age 65, three of the four corners are covered. Corner map 5c shows that these are test cases 16 and 8 for age 64, and test case 10 for age 65. Test cases 5c give the weight-height pairs respectively: (132,71) for BMI 18.41, (133,71) for BMI 18.55, (107,64) for BMI 18.37.

Corner map 5c.

		BMI	18.5		25		30	
Height	***Age***	***Limit***	max	min	max	min	max	min
71	**65**	min			3			4
	64	max	16	8		12		15
	20	min					20	
	19	max		$18^{g}\ 2^{b}$				
64	**65**	min	10				19	
	64	max					1	
	20	min	6		14	7		11
	19	max				$5^{g}\ 17^{b}$	$13^{g}\ 9^{b}$	

b Expected result is in boy equivalence class.

g Expected result is in girl equivalence class.

The final example in this section removes the explicit pairing of equivalence classes with other factor values. The weight function and the equivalence class functions all are substitution functions now. Example 5d pairs all the input factors (except weight) with both of the boundary factors. Twelve test cases result, as shown in Test cases 5d. The BMI-age corners are graphed in Figure 4, and Corner map 5d relates the corners to the test cases.

Test cases 5d. Boundary factor pairing

Test case	Input factors					Equivalence class factors			Boundary factors	
	Age	*Weight*[s]	*Height*	*Sex*	*Intake*	*Medicare*[s]	*Child*[s]	*Adult*[s]	*BMI*	*Limit*
1	65	179	71	male	2000	yes	no	normal	25	max
2	19	216	71	female	3000	no	girl	no	30	min
3	20	107	64	male	2000	no	no	underweight	18.5	max
4	64	145	64	female	3000	no	no	normal	25	max
5	64	216	71	male	2000	no	no	obese	30	min
6	65	108	64	female	3000	yes	no	normal	18.5	min
7	20	216	71	female	3000	no	no	obese	30	min
8	19	108	64	female	2000	no	girl	no	18.5	min
9	20	146	64	male	3000	no	no	overweight	25	min
10	19	179	71	male	2000	no	boy	no	25	max
11	65	174	64	male	2000	yes	no	overweight	30	max
12	64	132	71	male	3000	no	no	underweight	18.5	max

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

Figure 4 BMI-age corners (5d)

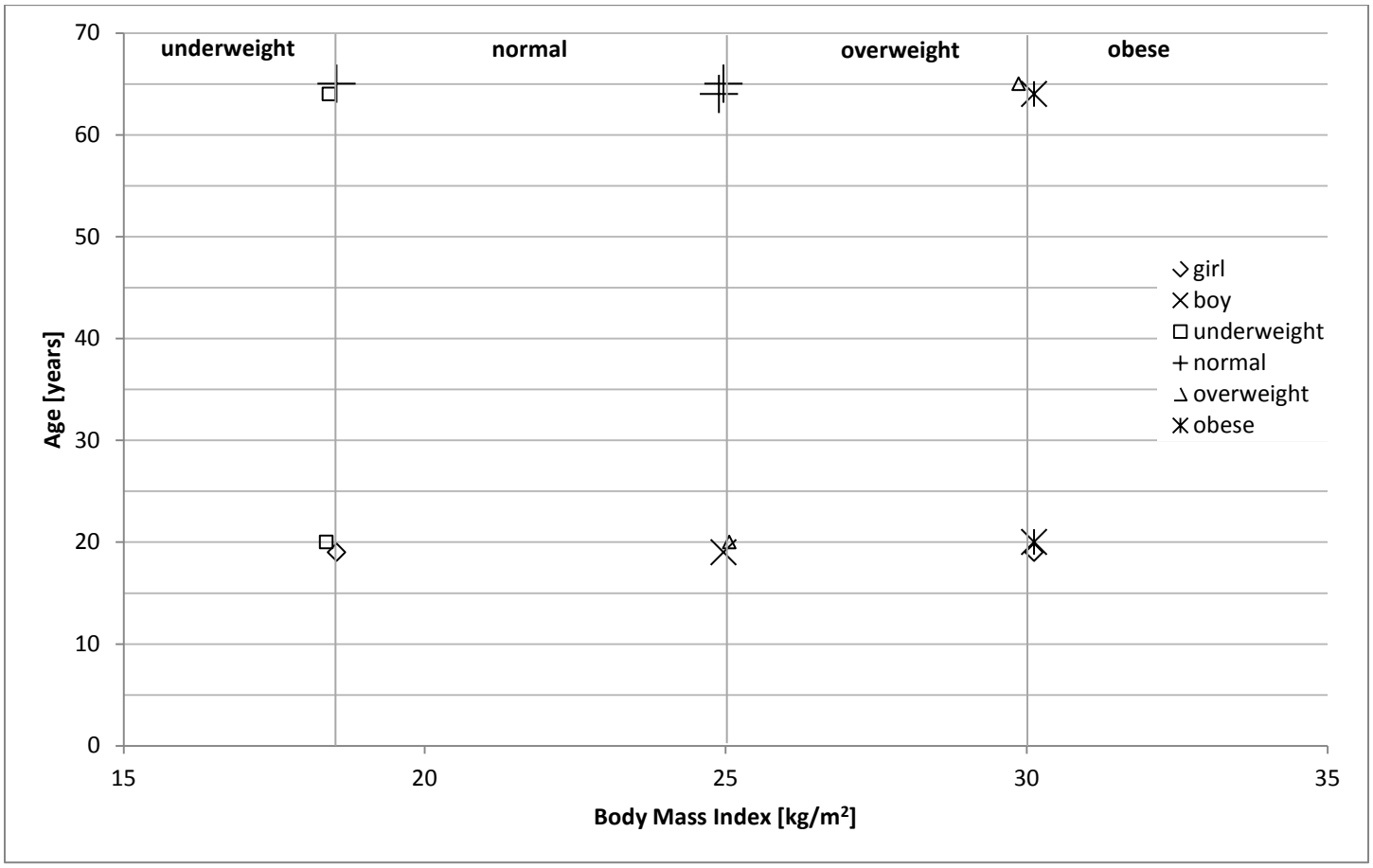


Figure 4 shows that at each of the boundary intersections of BMI and age, two of the four corners are covered. Both sides of each BMI boundary are covered, and both sides of each age boundary are covered.

Corner map 5d.

		BMI	18.5		25		30	
Height	***Age***	***Limit***	max	min	max	min	max	min
71	**65**	min			1			
	64	max	12					5
	20	min						7
	19	max			10[b]			2[g]
64	**65**	min		6			11	
	64	max			4			
	20	min	3			9		
	19	max		8[g]				

b Expected result is in boy equivalence class.

g Expected result is in girl equivalence class.

6. Edges, boundaries and corners

This section introduces edges (minimum and maximum input values, from Table I) into the designs. A simple way to cover the edges is to use two fixed values (minimum and maximum) for each ordered factor. Test cases 6a cover all pairs of factor values, so all of the edges' corners are covered.

Test cases 6a. Coverage of edges & their corners

Test case	**Input factors**					**Equivalence class factors**		
	Age	***Weight***	***Height***	***Sex***	***Intake***	***Medicare***[s]	***Child***[s]	***Adult***[s]
1	2	20	30	female	1	no	girl	no
2	129	499	89	male	9999	yes	no	obese
3	129	499	89	female	1	yes	no	obese
4	129	20	30	male	9999	yes	no	underweight
5	2	499	30	male	1	no	boy	no
6	2	20	89	female	9999	no	girl	no

s Factor is evaluated after test case generation using substitution function.
All other factors are evaluated using fixed values.

All Medicare and child equivalence classes are covered. The adult classes corresponding to extreme BMI classes, underweight and obese, are covered, but the intermediate classes are not.

Edge values can be introduced into other designs as well. When the edge values are included in Example 4b, the edges are paired with the representative (nonedge) input factor values and with the allowed equivalence classes. (Equivalence class functions from Table II are used.) The age factor uses boundary values, so its pairs with edge values cover the corresponding corners. Test cases 6b show the design. The 34 test cases were generated in the order of the test case numbers. Here they have been sorted by age to show the boundaries.

Test cases 6b. Coverage of univariate equivalence class boundaries, edges and their corners

Test case	Input factors					Equivalence class factors		
	Age	*Weight*	*Height*	*Sex*	*Intake*	*Medicare*[c]	*Child*[c]	*Adult*[c]
5	129	180	71	female	3000	yes	no	overweight
6	129	131	64	male	2000	yes	no	normal
24	129	499	89	male	1	yes	no	obese
25	129	20	71	female	9999	yes	no	underweight
34	129	499	30	male	2000	yes	no	obese
3	65	20	30	male	9999	yes	no	underweight
11	65	499	64	female	1	yes	no	obese
19	65	180	89	female	2000	yes	no	underweight
28	65	180	71	male	3000	yes	no	overweight
29	65	131	64	male	1	yes	no	normal
7	64	180	30	male	1	no	no	obese
10	64	20	89	female	3000	no	no	underweight
15	64	499	71	female	9999	no	no	obese
16	64	180	71	male	2000	no	no	overweight
17	64	131	64	female	3000	no	no	normal
1	20	499	89	female	2000	no	no	obese
12	20	131	71	female	1	no	no	underweight
18	20	131	30	male	3000	no	no	obese
20	20	20	64	female	9999	no	no	underweight
21	20	180	71	female	9999	no	no	overweight
30	20	131	64	female	9999	no	no	normal
32	20	180	71	male	1	no	no	overweight
4	19	20	64	female	1	no	girl	no
9	19	499	30	male	3000	no	boy	no
22	19	131	64	male	2000	no	boy	no
23	19	180	64	female	3000	no	girl	no
27	19	499	71	female	1	no	girl	no
31	19	20	89	male	3000	no	boy	no
33	19	180	30	male	9999	no	boy	no
2	2	131	71	male	3000	no	boy	no
8	2	131	89	female	9999	no	girl	no
13	2	180	89	male	1	no	boy	no
14	2	20	30	female	2000	no	girl	no
26	2	499	64	male	9999	no	boy	no

c Factor is evaluated before test case generation using combination function.
All other factors are evaluated using fixed values.

Example 6c is Example 5c with edge values included. Example 5c paired input factors, equivalence class factors and boundary factors. The weight values were evaluated from the height and boundary factors using a substitution function (Table IV). The Medicare and child equivalence class functions were from Table II; the adult function was from Table IV. This part of the design is reused in Example 6c, with the edge values for age, height and intake included. It is described as one set of test values (one *block* [1-3]) for the test case generator. However the weight edge values are fixed, not computed from the BMI boundary values.

Two additional blocks are included in the partition to allow for the weight edge values. One block lists the minimum weight value 20; the other lists the maximum 499. The adult equivalence class function is from Table II because it is determined by age, weight and height, not by the boundary factors. The two additional blocks use the BMI value edge to indicate it is not a boundary value; the limit value is min or max for weight value 20 or 499 respectively. The three blocks have the same values for the other input factors, and the same functions for the Medicare and child equivalence classes.

Test cases 6c. Equivalence class boundary and edge factor pairing

Test case	Input factors					Equivalence class factors			Boundary factors	
	Age	*Weight*[fs]	*Height*	*Sex*	*Intake*	*Medicare*[c]	*Child*[c]	*Adult*[cc]	*BMI*	*Limit*
1	64	337	89	female	2000	no	no	overweight	30	max
2	2	20	64	male	3000	no	boy	no	edge	min
3	129	133	71	male	1	yes	no	normal	18.5	min
4	65	499	30	male	9999	yes	no	obese	edge	max
5	19	33	30	female	2000	no	girl	no	25	min
6	20	107	64	female	9999	no	no	underweight	18.5	max
7	2	499	71	female	1	no	girl	no	edge	max
8	129	20	89	female	3000	yes	no	underweight	edge	min
9	65	146	64	male	3000	yes	no	overweight	25	min
10	20	216	71	female	3000	no	no	obese	30	min
11	2	281	89	male	9999	no	boy	no	25	max
12	64	20	71	male	2000	no	no	underweight	edge	min
13	19	174	64	male	1	no	boy	no	30	max
14	64	32	30	female	3000	no	no	normal	25	max
15	129	38	30	female	9999	yes	no	overweight	30	max
16	65	209	89	female	2000	yes	no	normal	18.5	min
17	20	20	30	female	1	no	no	underweight	edge	min
18	19	20	89	female	9999	no	girl	no	edge	min
19	20	499	64	male	2000	no	no	obese	edge	max
20	2	23	30	male	3000	no	boy	no	18.5	max
21	64	499	89	male	1	no	no	obese	edge	max
22	20	180	71	male	1	no	no	overweight	25	min
23	19	499	71	male	3000	no	boy	no	edge	max
24	64	108	64	female	9999	no	no	normal	18.5	min
25	2	337	89	male	2000	no	boy	no	30	max
26	19	132	71	female	9999	no	girl	no	18.5	max
27	129	146	64	female	2000	yes	no	overweight	25	min
28	65	215	71	male	1	yes	no	overweight	30	max
29	20	281	89	female	2000	no	no	normal	25	max
30	19	174	64	female	1	no	girl	no	30	max
31	65	20	89	female	2000	yes	no	underweight	edge	min
32	129	499	64	male	3000	yes	no	obese	edge	max
33	2	23	30	female	3000	no	girl	no	18.5	max

c Factor is evaluated before test case generation using combination function.
cc Factor is evaluated before test case generation using two different combination functions.
fs Factor is evaluated using fixed edge values and using a substitution function for boundary values.
All other factors are evaluated using fixed values.

Corner map 6c relates the BMI-age corners to the test cases. Test cases with minimum or maximum weight values are listed in the min[w] or max[w] columns respectively. Intake corners are not shown.

Corner map 6c.

		BMI	18.5		25		30			
Height	***Age***	***Limit***	min[w]	max	min	max	min	max	min	max[w]
89	**129**	max	8							
	65	min	31		16					
	64	max						1		21
	20	min				29				
	19	max	18[g]							
	2	min				11[b]		25[b]		
71	**129**	max			3					
	65	min						28		
	64	max	12							
	20	min					22		10	
	19	max		26[g]						23[b]
	2	min								7[g]
64	**129**	max					27			32
	65	min					9			
	64	max			24					
	20	min		6						19
	19	max						30[g] 13[b]		
	2	min	2[b]							
30	**129**	max						15		
	65	min								4
	64	max				14				
	20	min	17							
	19	max					5[g]			
	2	min		33[g] 20[b]						

b Expected result is in boy equivalence class.
g Expected result is in girl equivalence class.
w Corresponding values are min/max for weight factor.

7. Implications

These examples illustrate how specifying equivalence class and boundary requirements as simple functions, in a language familiar to software engineers, can automate much of the analysis for a particular test design. The same functions can be reused for different types of designs, and they can enhance consistency and accuracy as test factors and values change.

In the examples of Sections 4, 5 and 6, multiple partitions are not needed: Their equivalence class combination functions are evaluated for all combinations of their determinant factor values, to insure all classes are covered. And the resulting designs have strength 2, so they typically have fewer test cases than those using higher strength. These properties enable accurate coverage with lower test execution cost.

The control and flexibility offered by embedded functions are of particular note. For example, they enable us to verify multivariate boundaries. Four different BMI boundary designs were presented in Section 5. All of them use the weight boundary function of Table IV. Using a combination function we can pair the weight values with the other factors (Examples 5a and 5b). The same code as a substitution function enables us to skip pairing this 12-value factor to reduce the number of test cases (Examples 5c and 5d). However, each of the function's determinant factors is paired with the other factors. And other designs are possible: E.g. a dependent height function could be used instead of weight.

Equivalence class coverage has similar flexibility. Classes can be reached using equivalence class factors and/or boundary value factors. The three equivalence class factors in these examples all were evaluated as substitution functions or as combination functions together. Clearly other choices (some of each) are possible. And boundary value factors may be sufficient to cover classes when all the determinant factors are ordered (e.g. adult classes).

Embedded functions offer software test design improvements in automation, accuracy, control and flexibility. These findings indicate application of the new technology can advance test efficiency and quality.

References

[1] G. B. Sherwood, "Functional dependence and equivalence class factors in combinatorial test designs," Proceedings of the 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops, Cleveland, OH: 2014, pp. 108-117.

[2] G. B. Sherwood, "Embedded functions in combinatorial test designs," Proceedings of the 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops, Graz: 2015, pp. 1-10.

[3] G. B. Sherwood, "Embedded functions for constraints and variable strength in combinatorial testing," Proceedings of the 2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops, Chicago: 2016, pp. 65-74.

[4] M. Achour, F. Betz, A. Dovgal, et al., PHP Manual. Retrieved June 19, 2016, from the PHP Group: http://php.net/manual/en/index.php.

[5] P. G. Neumann, moderator, "The risks digest, forum on risks to the public in computers and related systems." Retrieved June 19, 2016, from http://catless.ncl.ac.uk/Risks.