Cpt S 422: Software Engineering Principles II

Black-box testing – Part 2

Dr. Venera Arnaoudova

Black-box testing methods

- ✓ Equivalence Class Partitioning
- ✓ Boundary-Value Analysis
- ❑ Category-Partition
- ❑ Decision tables
- ❑ Cause-Effect Graphs
- ❑ Logic Functions

Category-Partition Testing

Steps

- The system is divided into individual “functions” that can be independently tested
- The method identifies the *parameters* of each “function” and, for each parameter, identifies distinct *categories*
- Besides parameters, *environment conditions,* under which the function operates (characteristics of system state), can also be considered
- *Categories* are major properties or characteristics
- The *categories* are further subdivided into *choices* in the same way as equivalence partitioning is applied (possible “values”)

Small Example

- Function: Sorting an array
- Characteristics:
  - Length of array (Len)
  - Type of elements
  - Max value
  - Min value
  - Position of max value (Max pos)
  - Position of min value
- Choices for Len:
  - 0
  - 1
  - 2-100
  - >100

Steps (cont.)

- The *constraints* operating between choices are then identified, i.e., how the occurrence of one choice can affect the existence of another
  - E.g., in the array sorting example, if Len = 0, then the rest does not matter
- *Test frames* are generated which consist of the allowable combinations of choices in the categories (test specifications)
- Test frames are then converted into *test data*

Constraints

- *Properties, Selectors* associated with choices

Category A

ChoiceA1 [property X, Y, Z]

ChoiceA2

Category B

ChoiceB1

ChoiceB2 [if X and Z]

- Special annotation:
  - [Error]
  - [Single]

Complete Example

- Specification:
  - The program prompts the user for a positive integer in the range 1 to 20 and then for a string of characters of that length.
  - The program then prompts for a character and returns the position in the string at which the character was first found or a message indicating that the character was not present in the string.
  - The user has the option to search for more characters.

Parameters and Categories

- Three parameters: integer x (length), the string a, and the character c
- For x the categories are “in-range” (1-20) or “out-of-range”
- Categories for a: minimal, maximal, intermediate length
- Categories for c: character appears at the beginning, middle, end of string, or does not occur in the string

Choices

- Integer x, out-of-range: 0, 21
- Integer x, in-range: 1, 2-19, 20
- String a: 1, 2-19, 20
- Character c: first, middle, last, does not occur
- Note: sometimes only one choice in category

Formal Test Specifications

x:		
x1)	0	[error]
x2)	1	[property stringok, length1]
x3)	2-19	[property stringok, midlength]
x4)	20	[property stringok, length20]
x5)	21	[error]
a:		
a1)	Length 1	[if stringok and length1]
a2)	Length 2-19	[if stringok and midlength]
a3)	Length 20	[if stringok and length20]
c:		
c1)	At first position in string	[if stringok]
c2)	At last position in string	[if stringok and not length1]
c3)	In middle of string	[if stringok and not length1]
c4)	Not in string	[if stringok]

Test Frames and Cases

x 1	x = 0
x 2a1c1	x = 1, a = ‘A’, c = ‘A’
x 2a1c4	x = 1, a = ‘A’, c = ‘B’
x 3a2c1	x = 7, a = ‘ABCDEFG’, c = ‘A’
x 3a2c2	x = 7, a = ‘ABCDEFG’, c = ‘G’
x 3a2c3	x = 7, a = ‘ABCDEFG’, c = ‘D’
x 3a2c4	x = 7, a = ‘ABCDEFG’, c = ‘X’
x 4a3c1	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘A’
x 4a3c2	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘T’
x 4a3c3	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘J’
x 4a3c4	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘X’
x 5	x = 21

Criteria Using Choices

- All Combinations (AC): This is what was shown in the previous example, what is typically done when using category-partition. One value for every choice of every parameter must be used with one value of every (possible) choice of every other category.
- Each choice (EC): This is a weaker criterion. One value from each choice for each category must be used at least in one test case.
- Base Choice (BC): This criterion is a compromise. A base choice is chosen for each category, and a first base test is formed by using the base choice for each category. Subsequent tests are chosen by holding all but one base choice constant (i.e., we select a non-base choice for one category) and forming choice combinations by covering all non-base choices of the selected category. This procedure is repeated for each category.

Conclusions

- Makes testing decisions explicit (e.g., constraints), open for review
- Combine boundary analysis, robustness testing, and equivalence class partitioning
- Once the first step is completed, the technique is straightforward and can be automated
- The technique for test case reduction makes it useful for practical testing
- Identifying parameters and environments conditions, and categories, is heavily relying on the experience of the tester

Black-box testing methods

- ✓ Equivalence Class Partitioning
- ✓ Boundary-Value Analysis
- ✓ Category-Partition
- ❑ Decision tables
- ❑ Cause-Effect Graphs
- ❑ Logic Functions

Decision Table-Based Testing

Motivations

- Help express *test requirements* in a directly usable form
- Easy to understand and support the systematic derivation of tests
- Support automated or manual generation of test cases
- A particular response or response subset is to be selected by evaluating many related conditions
- Ideal for describing situations in which a number of combinations of actions are taken under varying sets of conditions, e.g., control systems

Structure

- Condition section lists *conditions* and combinations thereof
- Condition expressed relationship among *decision variables*
- Action section list *responses* to be produced when corresponding combinations of conditions are true
- Actions are *independent* of input order and the order in which conditions are evaluated
- Actions may appear more than once but each combination of conditions is unique

Table Structure

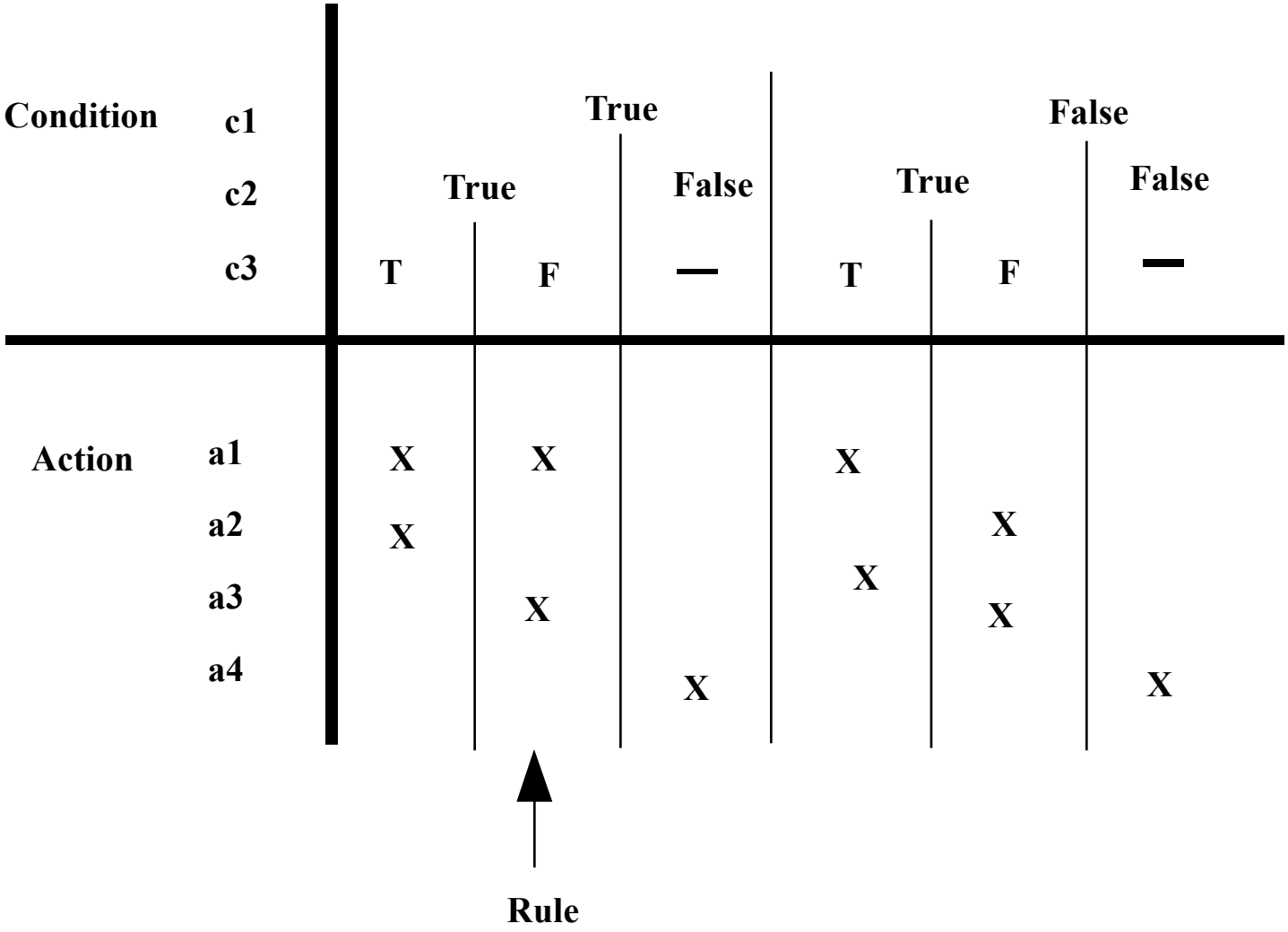
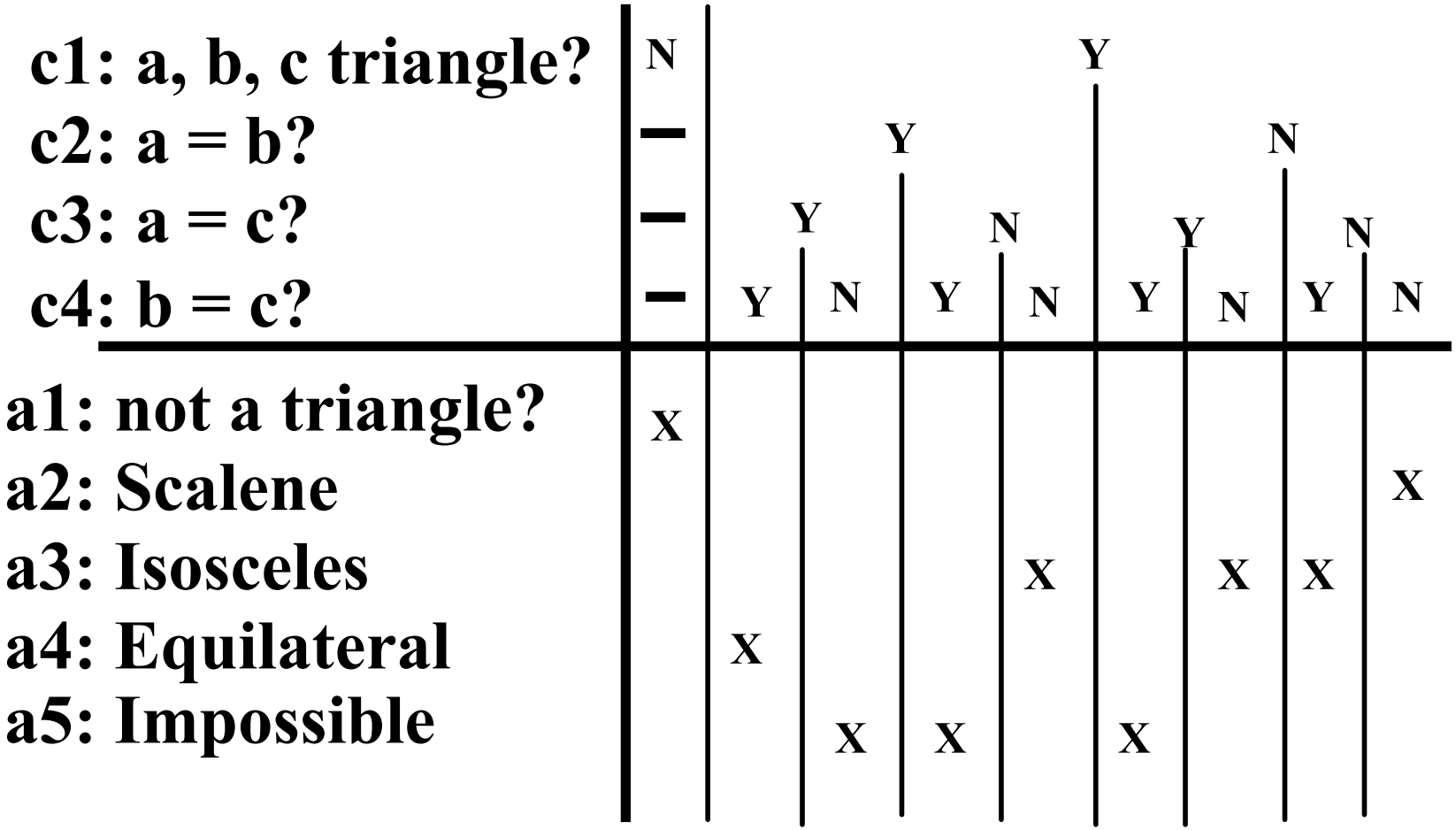


Table Example



Truth Table

conditions											
c1: a < b + c?	F	T	T	T	T	T	T	T	T	T	T
c2: b < a + c?	-	F	T	T	T	T	T	T	T	T	T
c3: c < a + b?	-	-	F	T	T	T	T	T	T	T	T
c4: a = b?	-	-	-	T	T	T	T	F	F	F	F
c5: a = c?	-	-	-	T	T	F	F	T	T	F	F
c6: b = c?	-	-	-	T	F	T	F	T	F	T	F
a1: Not a triangle	X	X	X								
a2: Scalene											X
a3: Isosceles							X		X	X	
a4: Equilateral				X							
a5: Impossible					X	X		X			

Test Cases

Case ID	a	b	c	Expected Output
TC1	4	1	2	Not a triangle
TC 2	1	4	2	Not a triangle
TC 3	1	2	4	Not a triangle
TC 4	5	5	5	Equilateral
TC 5	?	?	?	Impossible
TC 6	?	?	?	Impossible
TC 7	2	2	3	Isosceles
TC 8	?	?	?	Impossible
TC 9	2	3	2	Isosceles
TC 10	3	2	2	Isosceles
TC 11	3	4	5	Scalene

Ideal Usage Conditions

- One of several distinct responses is to be selected according to distinct cases of input variables
- These cases can be modeled by mutually exclusive boolean expressions on the input variables
- The response to be produced does not depend on the order in which input variables are set or evaluated (e.g., events are received)
- The response does not depend on prior inputs or outputs

Scale

- For n conditions, there may be at most 2^n *variants* (unique combinations of conditions and actions)
- But, fortunately, there are usually much fewer *explicit* variants …
- "Don't care" values in decision tables help reduce the number of variants
- "Don't care" can correspond to several cases:
  - The inputs are necessary but have no effect
  - The inputs may be omitted