Decision Table-Based Testing

Chapter 7

Decision Tables - Wikipedia

- A precise yet compact way to model complicated logic
- Associate conditions with actions to perform
- Can associate many independent conditions with several actions in an elegant way

Decision Table Terminology

Stub	Rule 1	Rule 2	Rules 3,4	Rule 5	Rule 6	Rules 7,8
c1	T	T	T	F	F	F
c2	T	T	F	T	T	F
c3	T	F	-	T	F	-
a1	X	X		X		
a2	X				X	
a3		X		X		
a4			X			X

condition stubs	condition entries
action stubs	action entries

Decision Table Terminology – 2

- Condition entries binary values
  - We have a **limited entry table**

- Condition entries have more than two values
  - We have an **extended entry table**

condition stubs	**condition entries**
action stubs	**action entries**

Printer Troubleshooting DT

Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Heck the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

A complete limited entry table

Test cases for decision tables

- **How are the entries in a decision table interpreted with respect to test cases?**
  - **Condition entries?**
  - **Action entries?**

Test cases for decision tables – 2

- Conditions are interpreted as
  - Input
  - Equivalence classes of inputs
- Actions are interpreted as
  - Output
  - Major functional processing portions
- With complete decision tables
  - Have complete set of test cases

Triangle Decision Table

C1: <a, b,c > forms a triangle?	F	T	T	T	T	T	T	T	T
C3: a = b?	–	T	T	T	T	F	F	F	F
C4: a = c?	–	T	T	F	F	T	T	F	F
C5: b = c?	–	T	F	T	F	T	F	T	F
A1: Not a Triangle	X								
A2: Scalene									X
A3: Isosceles					X		X	X	
A4: Equilateral		X							
A5: Impossible			**X**	**X**		**X**			

Action added by a tester showing impossible rules

Triangle Decision Table – refined

C1-1: a < b+c?	F	T	T	T	T	T	T	T	T	T	T
C1-2: b < a+c?	–	F	T	T	T	T	T	T	T	T	T
C1-3: c < a+b?	–	–	F	T	T	T	T	T	T	T	T
C2: a = b?	–	–	–	T	T	T	T	F	F	F	F
C3: a = c?	–	–	–	T	T	F	F	T	T	F	F
C4: 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			

Similar to equivalence classes we can refine the conditions

Triangle Test Cases

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	???	???	???	Impossible
DT6	???	???	???	Impossible
DT7	2	2	3	Isosceles
DT8	???	???	???	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

NextDate Decision Table

- The NextDate problem illustrates the correspondence between equivalence classes and decision table structure

- The NextDate problem illustrates the problem of dependencies in the input domain
  - Decision tables can highlight such dependencies
  - Impossible dates can be clearly marked as a separate action

NextDate Equivalence Classes – for 1st try

M1 = {month : 1 .. 12 | days(month) = 30 }

M2 = {month : 1 .. 12 | days(month) = 31 }

M3 = {month : {2} }

D1 = {day : 1 .. 28}

D2 = {day : {29} }

D3 = {day : {30} }

D4 = {day : {31} }

Y1 = {year : 1812 .. 2012 | leap_year (year) }

Y2 = {year : 1812 .. 2012 | common_year (year) }

As in discussion for equivalence classes

NextDate decision table with mutually exclusive conditions

C1: month in M1?	T	–	–
C2: month in M2?	–	T	–
C3: month in M3?	–	–	T
A1: Impossible			
A2: Next Date			

Because a month is in an equivalence class
we cannot have T more than one entry. The do not care entries are really F.

NextDate DT (1st try - partial)

How many rules

- **for a complete table?**
- **with don't care entries?**

C1: month in M1?	T	T	T	T	T	T	T	T				
C2: month in M2?									T	T	T	T
C3: month in M3?												
C4: day in D1?	T	T							T	T		
C5: day in D2?			T	T							T	T
C6: day in D3?					T	T						
C7: day in D4?							T	T				
C8: year in Y1?	T		T		T		T		T		T	
C9: year in Y2?		T		T		T		T		T		T
A1: Impossible							X	X				
A2: Next Date	X	X	X	X	X	X			X	X	X	X

NextDate Equivalence Classes – for 2nd try

M1 = {month : 1 .. 12 | days(month) = 30 }

M2 = {month : 1 .. 12 | days(month) = 31 }

M3 = {month : {2} }

D1 = {day : 1 .. 28}

D2 = {day : {29} }

D3 = {day : {30} }

D4 = {day : {31} }

Y1 = {year : {2000} }

Y2 = {year : 1812 .. 2012 | leap_year (year) $\wedge$ year $\neq$ 2000 }

Y3 = {year : 1812 .. 2012 | common_year (year) }

Handle leap year better

NextDate DT (2nd try - part 1)

This table has 16 rules.
How many rules
for a complete table?

C1: month in	M1	M1	M1	M1	M2	M2	M2	M2
C2: day in	D1	D2	D3	D4	D1	D2	D3	D4
C3: year in	–	–	–	–	–	–	–	–
A1: Impossible				X				
A2: Increment day	X	X			X	X	X	
A3: Reset day			X					X
A4: Increment month			X					???
A5: reset month								???
A6: Increment year								???

Extended entry table – more refined actions

NextDate DT (2nd try - part 2)

C1: month in	M3	M3	M3	M3	M3	M3	M3	M3
C2: day in	D1	D1	D1	D2	D2	D2	D3	D3
C3: year in	Y1	Y2	Y3	Y1	Y2	Y3	–	–
A1: Impossible				X		X	X	X
A2: Increment day		X						
A3: Reset day	X		X		X			
A4: Increment month	X		X		X			
A5: reset month								
A6: Increment year								

New Equivalence Classes – for 3rd try

M1 = {month : 1 .. 12 | days(month) = 30 }

M2 = {month : 1 .. 12 | days(month) = 31 ∧ month ≠ 12 }

M3 = {month : {12} }

M4 = {month : {2} }

D1 = {day : 1 .. 27}

D2 = {day : {28} }

D3 = {day : {29} }

D4 = {day : {30} }

D5 = {day : {31} }

Y1 = {year : 1812 .. 2012 | leap_year (year) }

Y2 = {year : 1812 .. 2012 | common_year (year) }

Handle end of month and year better

NextDate DT (3rd try - part 1)

A 22 rule table

C1: month in	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2
C2: day in	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5
C3: year in	–	–	–	–	–	–	–	–	–	–
A1: Impossible					X					
A2: Increment day	X	X	X			X	X	X	X	
A3: Reset day				X						X
A4: Increment month				X						X
A5: reset month										
A6: Increment year										

NextDate DT (3rd try - part 2)

C1: month in	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
C2: day in	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
C3: year in	–	–	–	–	–	–	Y1	Y2	Y1	Y2	–	–
A1: Impossible										X	X	X
A2: Increment day	X	X	X	X		X	X					
A3: Reset day					X			X	X			
A4: Increment month								X	X			
A5: reset month					X							
A6: Increment year					X							

Don't care entries and rule counts

- Limited entry tables with N conditions have 2^N rules.
- Don't care entries reduce the number of rules by implying the existence of non-explicitly stated rules.

  - **How many rules does a table contain including all the implied rules due to don't care entries?**

Don't care entries and rule counts – 2

- Each don't care entry in a rule doubles the count for the rule
- For each rule determine the corresponding rule count
- Total the rule counts

Don't care entries and rule counts – 3

1	C1-1: a < b+c?	F	T	T	T	T	T	T	T	T	T	T
2	C1-2: b < a+c?	–	F	T	T	T	T	T	T	T	T	T
3	C1-3: c < a+b?	–	–	F	T	T	T	T	T	T	T	T
4	C2: a = b?	–	–	–	T	T	T	T	F	F	F	F
5	C3: a = c?	–	–	–	T	T	F	F	T	T	F	F
6	C4: b = c?	–	–	–	T	F	T	F	T	F	T	F
	Rule count	32	16	8	1	1	1	1	1	1	1	1

+/ = 64

$= 2^6$

Don't care entries and rule counts – 4

- **How many rules do extended entry tables have?**
- **What is the rule count with don't care entries?**
  - See DDT-16, -17 (NextDate 2'nd try)
  - See DDT-19, -20 (NextDate 3'rd try)
  - See Table 7.9, page 107, for a redundant table
    - More rules than combination count of conditions
  - See Table 7.10, page 108, for an inconsistent table
    - More rules than combination count of conditions

Applicability

- The specification is given or can be converted to a decision table .
- The order in which the predicates are evaluated does not affect the interpretation of the rules or resulting action.
- The order of rule evaluation has no effect on resulting action .
- Once a rule is satisfied and the action selected, no other rule need be examined.
- The order of executing actions in a satisfied rule is of no consequence.

Applicability – 2

- The restrictions do not in reality eliminate many potential applications.
  - In most applications, the order in which the predicates are evaluated is immaterial.
  - Some specific ordering may be more efficient than some other but in general the ordering is not inherent in the program's logic.

Decision Tables – Properties

- You have constructed a decision table
  - **Before deriving test cases, what properties should the decision table have?**

Decision Tables – Properties – 2

- Before deriving test cases, ensure that
  - The rules are complete
    - Every combination of predicate truth values is explicit in the decision table
  - The rules are consistent
    - Every combination of predicate truth values results in only one action or set of actions

Guidelines and Observations

- Decision Table testing is most appropriate for programs where one or more of the conditions hold.
  - **What are those conditions?**

Guidelines and Observations – 2

- Decision Table testing is most appropriate for programs where
  - There is a lot of decision making
  - There are important logical relationships among input variables
  - There are calculations involving subsets of input variables
  - There are cause and effect relationships between input and output
  - There is complex computation logic (high cyclomatic complexity)

Guidelines and Observations – 3

- **What are some problems with using decision tables?**

Guidelines and Observations – 4

- Decision tables do not scale up very well
  - May need to
    - Use extended entry decision tables
    - Algebraically simplify tables

- Decision tables need to be iteratively refined
  - The first attempt may be far from satisfactory
    - Similar to using equivalence classes

Guidelines and Observations – 5

- Redundant rules
  - More rules than combination count of conditions
  - Actions are the same
  - Too many test cases
  - See Table 7.9, page 107
- Inconsistent rules
  - More rules than combination count of conditions
  - Actions are different for the same conditions
  - See Table 7.10, page 108
- Missing rules
  - Incomplete table

Variable Negation Strategy

- An approach that can help with the scaling problems of decision table-based testing
- Applicable when the system under test can be represented as a truth table (binary input and output)
- Designed to select a small subset of the 2^N test cases

Example truth table

Z = F (A, B, C, D)

Variant Number	Normal Pressure	Call For Heat	Damper Shut	Manual Mode	Ignition Enable
	A	B	C	D	Z
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Deriving the Logic Function

- Boolean algebra expressions
  - **A B** = A and B
  - **A + B** = A or B
  - **~A** = not A

- A logic function maps N Boolean input variables to a Boolean output variable

- A truth table is an enumeration of all possible input and output values

Logic function

- The logic function for the example is

  Z = A B ~C + A D

- Several techniques to derive it
  - Karnaugh maps
  - Cause-effect graphs

- A compact logic function will produce more powerful test cases

Variable Negation Strategy

- Designed to reveal faults that hide in a don't care
- The test suite contains:
  - **Unique true points**: A variant per term t, so that t is True and all other terms are False
    - In the expression **A B ~C + A D** , **A B ~C** and **A D** are terms
  - **Near False Points**: A variant for each literal in a term. The variant is obtained by negating the literal and is selected only if it makes Z = 0
- Each term variant creates a test candidate set

True points

- Unique true point candidate sets in boiler example
  - Variants in the set {12} make **A B ~C** true but not **A D**
    - Variant 13 makes both **A B ~C** and **A D** true and as a consequence is not included in the set
  - Variants in the the set {9,11,15} make **A D** true but not **A B ~C**
    - Variant 13 makes both **A B ~C** and **A D** true and as a consequence is not included in the set

Near false points

Candidate set number	Term negation	Function variants containing this negation	Function variants containing this negation where Z = 0
1 Org. term	**A B ~C**	**–**	**12**
2	A B C	14, 15	14
3	A ~B ~C	8, 9	8
4	~A B ~C	4, 5	4, 5
5 Org. term	**A D**	**–**	**9, 11, 15**
6	A ~D	8, 10, 12, 14	8, 10, 14
7	~A D	1, 3, 5, 7	1, 3, 5, 7

Near false points are in black, candidate set numbers 2, 3, 4, 6 and 7. In green are true points.

Selecting the test cases

- At least one variant from each candidate set
- Can be done by inspection
- Random selection is also used
- Near False Points exercise combinations of don't care values
- 6% of all possible tests are created
- 98% of simulated bugs can be found

Selecting test cases – 2

Test Candidate Set

Variant	1	2	3	4	5	6	7	Test case?
0								
1							X	
2								
3							X	
4				X				
5				X			X	X M
6								
7							X	
8			X			X		X M
9					X			M
10						X		
11					X			X .
12	X							X M
13								
14		X				X		X M
15					X			X .

Test suite

- Candidate sets

  12

  14

  8

  4, 5

  9, 11, 15

  8, 10, 14

  1, 3, 5, 7

- Minimum Test suite variants

  5 candidate sets 4 & 7

  8 candidate sets 3 & 6

  9 candidate set 5

  12 candidate set 1

  14 candidate set 2