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

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				

Let' s try this for the Triangle problem

Triangle Decision Table

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			

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 problem of dependencies in the input domain
- Decision tables can highlight such dependencies
- Impossible dates can be clearly marked as a separate action
- Let’s try it…

NextDate Equivalence Classes

M1= {month | month has 30 days}
M2= {month | month has 31 days}
M3= {month | month is February}
D1= {day | $1 \leq$ day ≤ 28}
D2= {day | day = 29}
D3= {day | day = 30}
D4= {day | day=31}
Y1= {year | year = 1900 or 2100}
Y2= {year | year is a leap year}
Y3= {year | year is a common year}

NextDate DT (1st try - partial)

C1: month in M1?	T	T	T	T	T	T	T	T	T	T	T	T
C2: month in M2?												
C3: month in M3?												
C4: day in D1?	T	T	T									
C5: day in D2?				T	T	T						
C6: day in D3?							T	T	T			
C7: day in D4?										T	T	T
C8: year in Y1?	T			T			T			T		
C9: year in Y2?		T			T			T			T	
C10: year in Y3?			T			T			T			T
A1: Impossible										X	X	X
A2: Next Date	X	X	X	X	X	X	X	X	X			

NextDate DT (2nd try - part 1)

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								?

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

M1= {month | month has 30 days}
M2= {month | month has 31 days}
M3= {month | month is December}
M4= {month | month is February}
D1= {day | 1 ≤ day ≤ 27}
D2= {day | day = 28}
D3= {day | day = 29}
D4= {day | day = 30}
D5= {day | day=31}
Y1= {year | year is a leap year}
Y2= {year | year is a common year}

NextDate DT (3rd try - part 1)

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							

Test Case Design

- To identify test cases with decision tables, we interpret conditions as inputs, and actions as outputs.
- Sometimes conditions end up referring to equivalence classes of inputs, and actions refer to major functional processing portions of the item being tested.
- The rules are then interpreted as test cases.

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

- 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 - Issues

- 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
  - 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

- Decision tables do not scale up very well
  - May need to
    - Use extended entry decision tables
    - Algebraically simplify tables
- Decision tables can be iteratively refined
  - The first attempt may be far from satisfactory

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

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

- Review boolean algebra
  - **AB** = 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 = AB~C + AD
- 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
  - **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 variant creates a test candidate set
- Unique true point candidate sets in boiler example: {12} {9,11,15}

Negation variants

Candidate set number	Term negation	Variants containing this negation	Variants containing this negation where Z=0
2	ABC	14,15	14
3	A~B~C	8,9	8
4	~AB~C	4,5	4,5
6	A~D	8,10,12,14	8,10,14
7	~AD	1,3,5,7	1,3,5,7

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

Test suite

- Candidate sets
  12
  14
  8
  4,5
  9,11,15
  8,10,14
  1,3,5,7
- Minimum Test suite
  5
  8
  9
  12
  14