Decision Table

Decision Tables - General

Decision tables are a precise yet compact way to model complicated logic. Decision tables, like if-then-else and switch-case statements, associate conditions with actions to perform.

But, unlike the control structures found in traditional programming languages, decision tables can associate many independent conditions with several actions in an elegant way.

Decision Tables - Usage

- Decision tables make it easier to observe that all possible conditions are accounted for.

- Decision tables can be used for:
  - Specifying complex program logic
  - Generating test cases (Also known as *logic-based testing)*

- *Logic-based testing* is considered as:
  - structural testing when applied to structure (i.e. control flow graph of an implementation).
  - functional testing when applied to a specification.

Decision Tables - Structure

Conditions - *(Condition stub)*	Condition Alternatives – *(Condition Entry)*
Actions – *(Action Stub)*	Action Entries

- Each condition corresponds to a variable, relation or predicate
- Possible values for conditions are listed among the condition alternatives
  - Boolean values (True / False) – Limited Entry Decision Tables
  - Several values – Extended Entry Decision Tables
  - Don’t care value

- Each action is a procedure or operation to perform
- The entries specify whether (or in what order) the action is to be performed

- To express the program logic we can use a limited-entry decision table consisting of 4 areas called the *condition stub*, *condition entry*, *action stub* and the *action entry*:

Condition entry

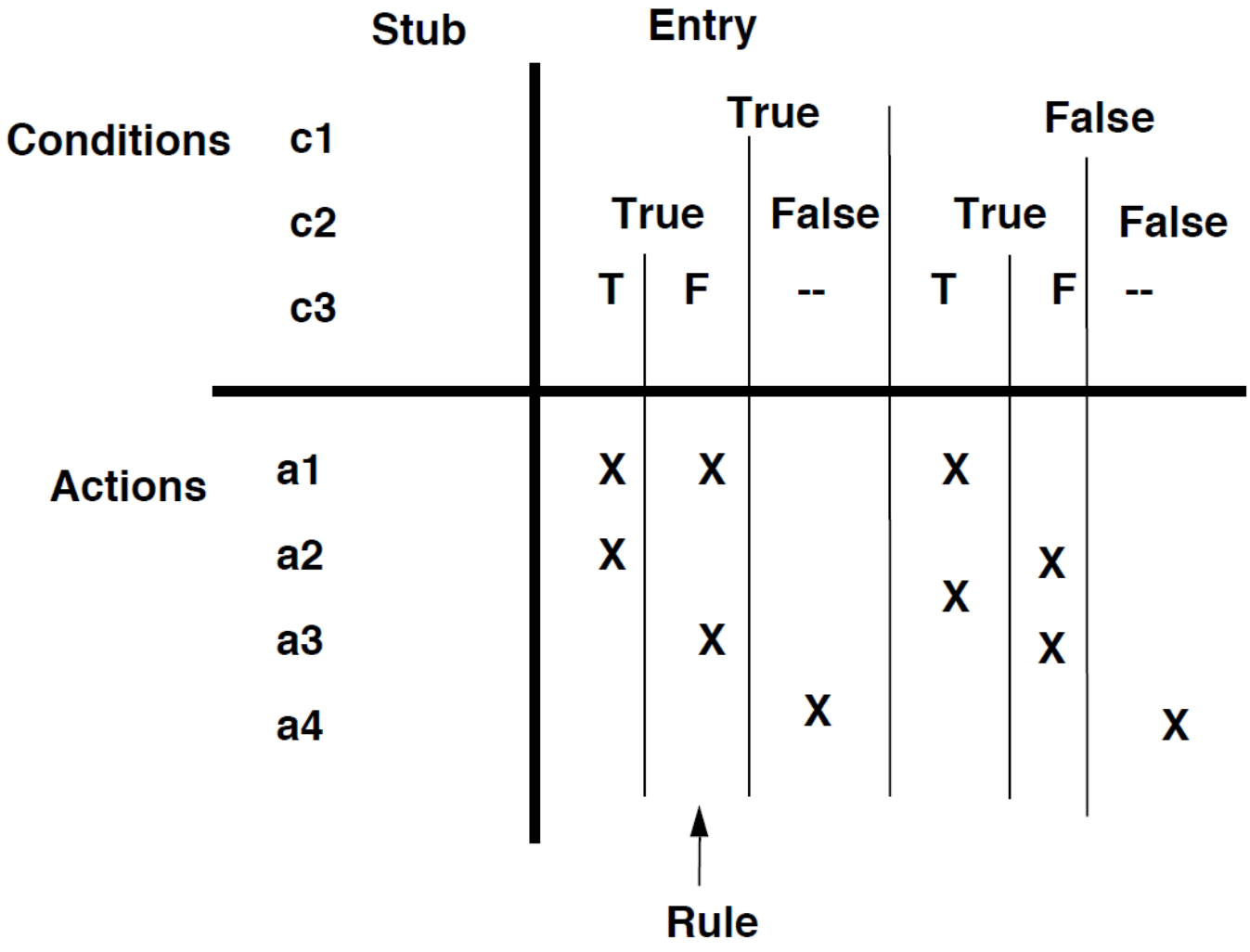
	Rule1	Rule2	Rule3	Rule4
Condition stub Condition 1	Yes	Yes	No	No
Condition 2	Yes	X	No	X
Condition 3	No	Yes	No	X
Action stub Condition 4	No	Yes	No	Yes
Action1	Yes	Yes	No	No
Action2	No	No	Yes	No
Action3	No	No	No	Yes

Action Entry

- We can specify *default rules* to indicate the action to be taken when none of the other rules apply.
- When using decision tables as a test tool, default rules and their associated predicates must be explicitly provided.

	Rule5	Rule6	Rule7	Rule8
Condition1	X	No	Yes	Yes
Condition2	X	Yes	X	No
Condition3	Yes	X	No	No
Condition4	No	No	Yes	X
Default action	Yes	Yes	Yes	Yes

Decision Table Terminology



Decision Table - Example

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				

Decision Table Example

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Decision Table Development Methodology

1. Determine conditions and values
2. Determine maximum number of rules
3. Determine actions
4. Encode possible rules
5. Encode the appropriate actions for each rule
6. Verify the policy
7. Simplify the rules (reduce if possible the number of columns)

Decision Tables - Usage

- The use of the decision-table model is applicable when :
  - 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.

- 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 using the tables, ensure:
  - rules must be complete
    - every combination of predicate truth values plus default cases are explicit in the decision table
  - rules must be consistent
    - every combination of predicate truth values results in only one action or set of actions

Test Case Design

- Once the specification has been verified, the objective is to demonstrate that the implementation provides the correct action for all combinations of predicate values:
  - if there are *k* rules over *n* binary predicates, then there are at least *k* cases and at most 2^n cases to consider.
- Base test design on unexpanded rules or on the expanded rules with 2^n tests
  - find the input vector to force each case.

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.

Decision Table for the Triangle Problem

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
Actions	How many Xs? 11										
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 for the Triangle Problem

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

Decision Table for NextDate (First Attempt)

- Let us consider the following equivalence classes:

M1= {month | month has 30 days}
M2= {month | month has 31 days}
M3= {month | month is February}
D1= {day | $1 \leq day \leq 28$}
D2= {day | day = 29}
D3= {day | day = 30}
D4= {day | day=31}
Y1= {year | year = 1900}
Y2= {year | $1812 \leq year \leq 2012$ AND $year \neq 1900$ AND (0 = year mod 4}
Y3= {year | $1812 \leq year \leq 2012$ AND $0 \neq$ year mod 4}

Decision Table for NextDate (1)

Conditions	1	2	3	4	5	6	7	8
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	-	-	-	-	-	-	-	-
Rule count	3	3	3	3	3	3	3	3
Actions								
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								?

Decision Table for NextDate (2)

Conditions	9	10	11	12	13	14	15	16
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	-	-
Rule count	1	1	1	1	1	1	3	3
Actions								
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								

Decision Table for NextDate (Second Attempt)

- Let us consider the following 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 | $1 \leq day \leq 27$}
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}

Decision Table for NextDate (1)

Conditions	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
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	-	-	-	-	-	-	-	-	-	-
Actions										
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										

Decision Table for NextDate (2)

Conditions	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
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	-	-
Actions												
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							

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)
- Decision tables do not scale up very well
- Decision tables can be iteratively refined