Decision table

Decision Table Based Testing

- **Decision table is based on logical relationships just as the truth table.**
- **Decision Table is a tool that helps us look at the combination of conditions**
  - **Completeness of conditions**
  - **Inconsistency of conditions**

Decision Table Based Testing

- Originally known as Cause and Effect Graphing
  - Done with a graphical technique that expressed AND-OR-NOT logic.
  - Causes and Effects were graphed like circuit components
  - Inputs to a circuit “caused” outputs (effects)
- Equivalent to forming a decision table in which:
  - inputs are conditions
  - outputs are actions
- Test every (possible) rule in the decision table.
- Recommended for logically complex situations.
- Excellent example of Model-Based Testing (MBT)

Content of a Decision Table

- Conditions
  - Binary in a Limited Entry Decision Table (LEDT)
  - Finite set in an Extended Entry Decision Table
  - Condition stub
  - Condition entries
- Actions
  - Also binary, either do or skip
  - The “impossible” action
- Rules
  - A rule consists of condition entries and action entries
  - A complete, non-redundant LEDT with n conditions has 2^n rules
  - Logically impossible combinations of conditions are “impossible rules”, denoted by an entry in the impossible action

Decision Table

- **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 make it easy 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:
  - <u>Structural testing</u> when applied to structure (i.e. control flow graph of an implementation).
  - <u>Functional testing</u> when applied to a specification.
- **Decision** tree is a two dimensional matrix. It is divided into four **parts**, condition stub, action stub, condition entry, and action entry

Steps

- **Step** 1: Identify the conditions and their values. ...
- **Step** 2: Identify Actions. ...
- **Step** 3: **Draw** the Condition and Actions. ...
- **Step** 4: Calculate Number of Rules. ...
- **Step** 5: Populate the Condition Alternatives. ...
- **Step** 6: Fill in the Action Entries. ...
- **Step** 7: Reduce the **Table** if Necessary

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

Condition stub

Action stub

	Rule1	Rule2	Rule3	Rule4
Condition1	Yes	Yes	No	No
Condition2	Yes	X	No	X
Condition3	No	Yes	No	X
Condition4	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 - 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				

Printer Troubleshooting

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.

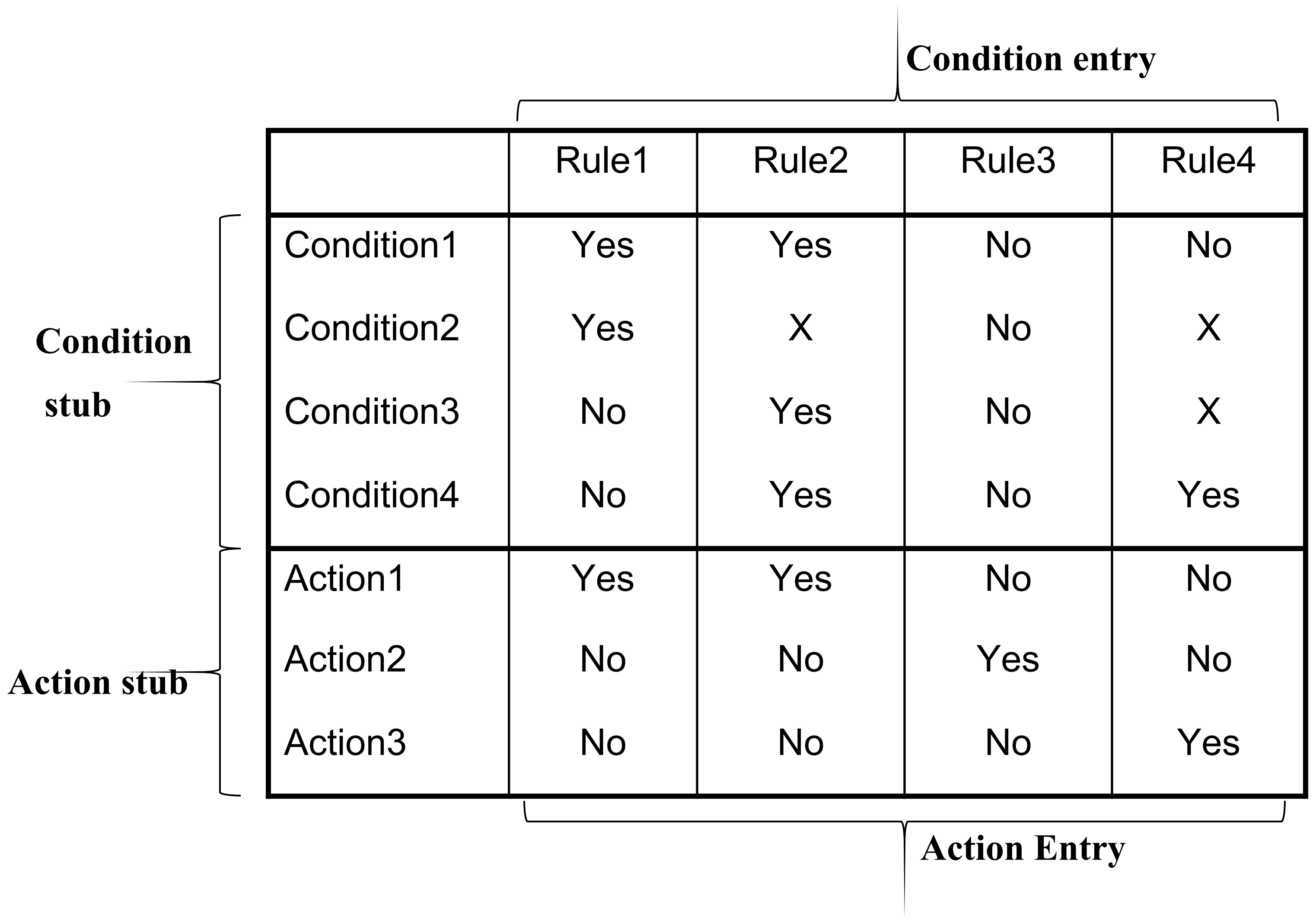
Decision Tables - Issues

- 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

- 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.

Definition


Condition entry
Condition stub
Action stub
Action Entry

	Rule1	Rule2	Rule3	Rule4
Condition1	Yes	Yes	No	No
Condition2	Yes	X	No	X
Condition3	No	Yes	No	X
Condition4	No	Yes	No	Yes
Action1	Yes	Yes	No	No
Action2	No	No	Yes	No
Action3	No	No	No	Yes

Example 1- Email

Email	False	True	False	True
Password	False	False	True	True
Expected Result	Error: Please enter email	Error: Please enter password	Error: Enter Email	Login processed

2*2=4

email	Blank	Blank	Blank	Invalid	Invalid	Invalid	Valid	Valid	Valid
psw	Blank	Invalid	Valid	Blank	Invalid	Valid	Blank	Invalid	Valid
Exp o/p	Error: Enter email	Error: Enter email	Error: Enter email	Error: Enter valid email	Error: Login Failed	Error: Enter valid email	Error; Enter psw	Error: Login Failed	------
Show o/p	Login Page	Login Page	Login Page	Login Page	Login Page	Login Page	Login Page	Login Page	Home Page

No of combinations:
3 power 2=9

Example 2- Library

User registered	False	False	False	False	True	True	True	True
No dues	False	False	True	True	False	False	True	True
Under borrow limit	False	True	False	True	False	True	False	True
Borrow book	No	No	No	No	No	No	No	Yes

No of test cases = 2*2*2 = 8

Simplified Library Example by following simplification step

User registered	False	False	True	True	True
No dues	False	True	False	True	True
Under borrow limit	- (Don't Care)	- (Don't Care)	- (Don't Care)	False	True
Borrow book	No	No	No	No	Yes

$\longrightarrow$

User registered	False	True	True	True
No dues	False	False	True	True
Under borrow limit	-	- (Don't Care)	False	True
Borrow book	-	No	No	Yes

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											
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 ≤ day ≤ 28}
D2= {day | day = 29}
D3= {day | day = 30}
D4= {day | day=31}
Y1= {year | year = 2000}
Y2= {year | year is a common year)
Y3= {year | year is a non century leap year}

Note year Y2 is a set of years between 1812 and 2012 evenly divisible by 4 excluding the year 2000

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 (2nd 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 ≤ 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}

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

Components of a Decision Table

rules

“binary” conditions

values of conditions

	R1	R2	R3	R4	R5	R6	R7	R8
C1	T	T	T	T	F	F	F	F
C2	T	T	F	F	T	T	F	F
C3	T	F	T	F	T	F	T	F
a1	x			x	x			x
a2	x							x
a3		x				x		
a4			x	x			x	x
a5	x			x				

actions

actions taken

Read a Decision Table by columns of rules : (e.g. R1 from reqs. or design says when all conditions are T, then actions a1, a2, and a5 should occur)

Example (continued)

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	T
a1	X	X		X		—
a2	X				X	
a3		X				
a4			X	X		X

- The condition entries in rules 3 and 4, and rules 7 and 8 have the same actions. The "—" means ...
  - "Don't Care" (as in circuit analysis),
  - Irrelevant, or
  - not applicable, n/a

Example (continued)

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

- Rule counting
  - A rule with no don’t care entries counts as 1
  - Each don’t care entry in a rule doubles the rule count
  - For a table with **n** limited entry conditions, the sum of the rule counts should be 2^n.

Problematic Decision Tables

- For LEDTs, simple rule counting helps identify decision tables that are ...
  - incomplete (rule count $< 2^n$) ,
  - redundant (rule count $> 2^n$) , or
  - inconsistent
    - (rule count $> 2^n$) AND
    - At least two rules have identical condition entries but different action entries.
- Redundancy and inconsistency are more likely with algebraically simplified tables that have been “maintained”.

A Redundant DT

Conditions	***1 – 4***	***5***	***6***	***7***	***8***	***9***
c1	T	F	F	F	F	T
c2	—	T	T	F	F	F
c3	—	T	F	T	F	F
a1	X	X	X	—	—	X
a2	—	X	X	X	—	—
a3	X	—	X	X	X	X

- Rule 9 is redundant with Rules 1 – 4 (technically, with what was rule 4)
- But the action entries are identical (No harm, no foul?)

Month Equivalence Classes

(to be used in the next example)

- M1 ={x : x is a 30-day month}
- M2 ={x : x is a 31-day month}
- M3 ={x : x is February}

Last Day of Month Decision Table

Conditions	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***16***
c1. M1?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
c2. M2?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
c3. M3?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
c4. leap year?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
a1. last day = 30							x	x								
a2. last day = 31											x	x				
a3. last day = 28														x		
a4. last day = 29													x			
a5. impossible	x	x	x	x	x	x			x	x					x	x

- Rule pairs 1 and 2, 3 and 4, 5 and 6, 9 and 10 don’t need c4, so they could be combined, BUT
- Impossible because c1, c2, and c3 are mutually exclusive.

Extended Entry Decision Tables

- When conditions are mutually exclusive, exactly one must be true.
- Extended entry decision tables typically (but not necessarily) have mutually exclusive conditions.
- The “extended” part is because a condition stub is an incomplete statement that is completed by the condition entry.
- (See the revised Last Day of Month EEDT)

Revised Last Day of Month DT

c1. 30-day month?	M1	M1	—	—	—	—
c2. 31-day month?	—	—	M2	M2	—	—
c3. February?	—	—	—	—	M3	M3
c4. leap year?	T	F	T	F	T	F
a1. last day = 30	x	x				
a2. last day = 31			x	x		
a3. last day = 28						x
a4. last day = 29					x	

- When conditions are mutually exclusive, exactly one must be true.
- This can be further simplified.

Triangle Program Decision Table

c1: a, b, c form a triangle?	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								
a2: Scalene									X
a3: Isosceles					X		X	X	
a4: Equilateral		X							
a5: Impossible			X	X		X			

Expanding c1...

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 Problem Example ("short" form)

Pick input set, <a, b, c>, for each of the columns, or rules, below

Assume a, b and c are all between 1 and 200

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
1. a < b + c	F	T	T	T	T	T	T	T	T	T	T
2. b < a + c	-	F	T	T	T	T	T	T	T	T	T
3. c < a + b	-	-	F	T	T	T	T	T	T	T	T
4. a = b	-	-	-	T	T	T	T	F	F	F	F
5. a = c	-	-	-	T	T	F	F	T	T	F	F
6. b = c	-	-	-	T	F	T	F	T	F	T	F
1. Not triangle	X	X	X								
2. Scalene											X
3. Isosceles							X		X	X	
4. Equilateral				X							
5. "impossible"					X	X		X			

equivalent or all necessary

Note the Impossible cases

Req or Design should NOT have these cases

Concept of Rule Count

Rule Counting

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
Rule count	32	16	8	1	1	1	1	1	1	1	1
a1: Not a triangle	x	x	x								
a2: Scalene											x
a3: Isosceles						x			x	x	
a4: Equilateral				x							
a5: Impossible					x	x		x			

Corresponding 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

Advantages/Disadvantages of Decision Table

- Advantages: (check completeness & consistency)
  1. Allow us to start with a "complete" view, with no consideration of dependence
  2. Allow us to look at and consider "dependence," "impossible," and "not relevant" situations and eliminate some test cases.
  3. Allow us to detect potential *error in our Specifications*
- Disadvantages:
  1. Need to decide (or know) what conditions are relevant for testing - - - this may require Domain knowledge
     - e.g. need to know leap year for "next date" problem in the book
  2. Scaling up can be massive: *2^n rules for n conditions - - - that is if the conditions are binary and gets worse if the values are more than binary*

Decision table for triangle problem

- Conditions
  - A=B?
  - A=C?
  - B=C?
- Actions
  - Scalene
  - Isosceles
  - Equilateral

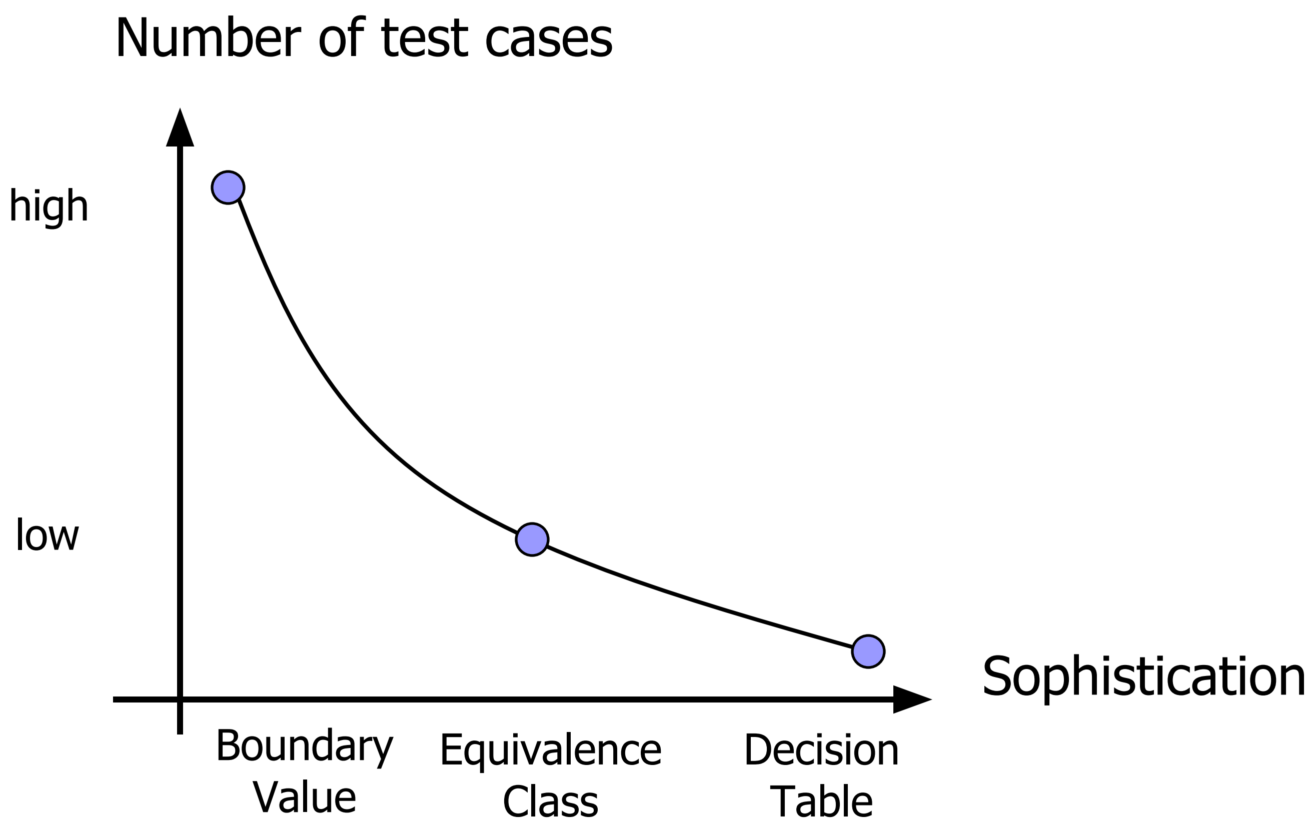
CASE STUDY- AIRLINE RESERVATION SYSTEM

- On-line flight reservation system (OFRS) is a software solution provided by "Always-We-Connect" travel agent for booking air tickets globally by Air India. The customer has to specify intended date of journey, from & to city names, max number of flight changes acceptable, class (business/economy), max waiting period acceptable and number of seats required. System shall not accept reservation less than 3 days from date of journey and more than 90 days gap from date of journey. Design adequate test cases to test this system. System will display whether reservations are available are not to customer based on conditions given by customer. If the seats are available in different class the same must be highlighted in the display

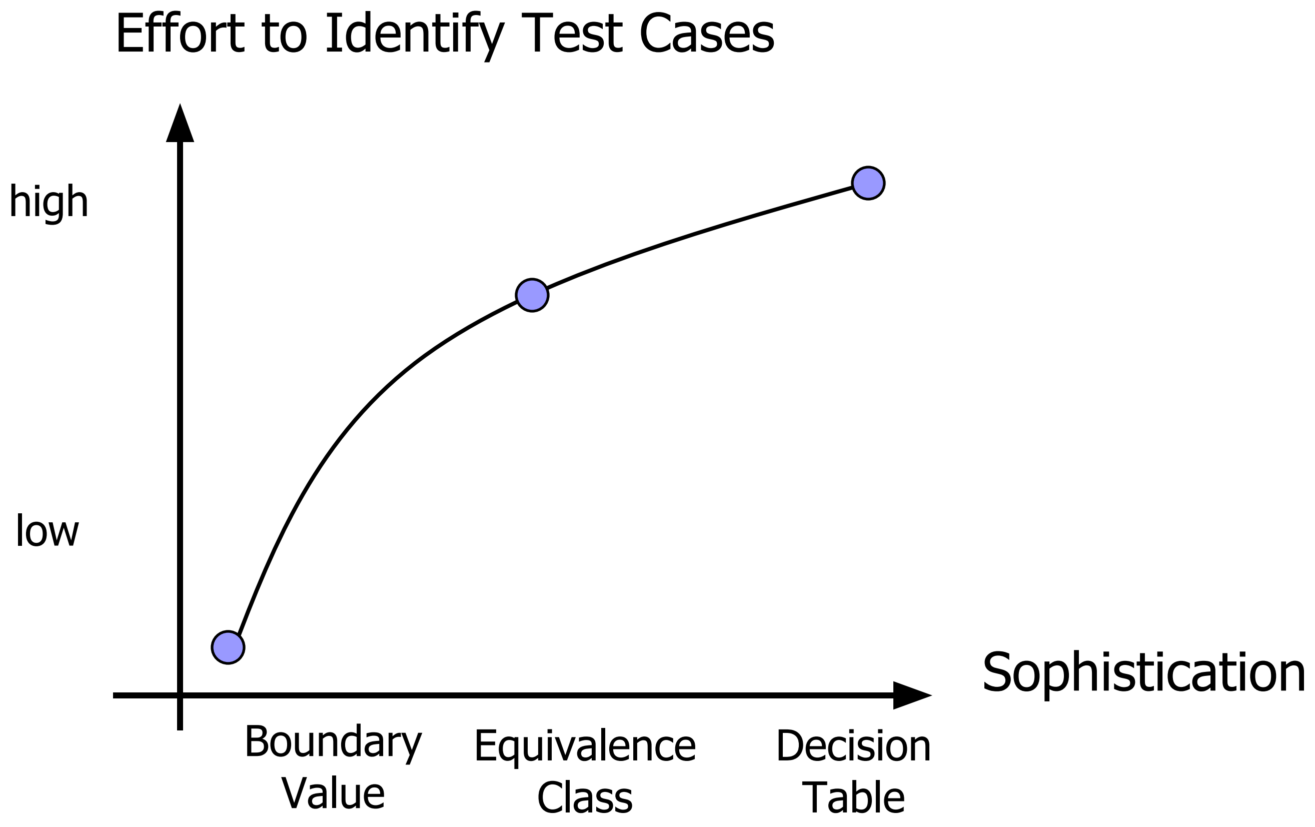
Example: Enquiry of tickets availability

- (Assumption: From and to city are entered correctly by user, the system will get all possible connections from database with relevant details)
  - Number of flight changes required must be less than max number of flight changes acceptable to customer
  - Total waiting period not to exceed max acceptable waiting period specified
  - Number of seats available must be greater or equal to number of seats requested
  - Reservations shall be allowed min 4-days in advance to a max of 90 days in advance from today
  - The class must be same as requested by customer if the seats are available in different class the same must be highlighted in the display

Summary


Number of test cases
high
low
Sophistication
Boundary Value
Equivalence Class
Decision Table

Summary-continued


Effort to Identify Test Cases
high
low
Sophistication
Boundary Value
Equivalence Class
Decision Table