Software Verification & Validation

Natthapong Jungteerapanich

Handout 4

Acknowledgements

- Some slides in this lecture are adapted from
  - Slides by **Mauro Pezzè and Michal Young** accompanying the textbook “ Software Testing and Verification”
  - Slides from **Gunnar Gotshalks**’s class at York University, Canada based on the book “Software Testing: A Craftman’s Approach”

- Recommended readings
  - Mauro Pezzè and Michal Young, “Software Testing and Verification”
  - Paul C. Jorgenson, “Software Testing: A Craftman’s Approach”
  - Robert V. Binder, “Testing Object-Oriented Systems”
  - NASA Technical Report, “A Practical Tutorial on Modified Condition/ Decision Coverage”

Testing logical expressions

- Imagine a control system for a steam boiler. The system reads the input consisting of the pressure, the volume, and the temperature of the boiler and performs suitable control functions.

- Consider the following requirement:
    - If **either the <u>pressure exceeds 500</u> and the <u>volume is below 10</u> or the <u>temperature exceeds 200</u>**, then halt the system.

- What would be suitable test cases?

Testing logical expressions

- Consider the following requirement:
    - If **either the <u>pressure exceeds 500</u> and the <u>volume is below 10</u> or the <u>temperature exceeds 200</u>**, then halt the system.

- Suppose we call the phrase in **bold** is called a "*decision*" and each underlined phrase is called a "*basic condition*".

- Hence, a decision is a Boolean expression where the basic conditions are the atomic propositions. The above decision can be rewritten symbolically as follows:

(p AND v) OR t

Decision Coverage

- **Decision Coverage Criterion (DC)**: each decision in the requirement is evaluated to true (T) in at least one test case and to false (F) in at least one test case.

- Degree of Decision Coverage:

$$C_{DC} = \frac{\text{\# truth values taken by all decisions}}{2 * \text{\# decisions}}$$

Basic Condition Coverage

- **Basic Condition Coverage Criterion (BCC)**: each basic condition in the program is evaluated to true (T) in at least one test case and to false (F) in at least one test case.

- Degree of Basic Condition Coverage:

$$C_{BCC} = \frac{\text{\# truth values taken by all basic conditions}}{2 * \text{\# basic conditions}}$$

- Also known as **Condition Coverage Criterion**

Condition/Decision Coverage

- Does decision coverage implies basic condition coverage?
- Does basic condition coverage implies decision coverage?
- **Basic Condition Coverage + Decision Coverage Criterion (**also called **Condition/Decision Coverage)**
  - each basic condition in the requirement is evaluated to true (T) in at least one test case and to false (F) in at least one test case;
  - each decision point in the requirement is evaluated to all possible outcomes (i.e. both T and F) during the test.

Compound Condition Coverage

- **Multiple Condition Coverage Criterion (**also called **Compound Condition Coverage):**
  - Cover all possible combination of the basic conditions within each Boolean expression

DH == 1 || DL == -1

All possible combinations

DH==1	DL==-1	DH==1 || DL==-1
T	T	T
F	T	T
T	F	T
F	F	F

All possible combinations with short-circuit evaluation

DH==1	DL==-1	DH==1 || DL==-1
T	-	T
F	T	T
F	F	F

Compound conditions: Exponential complexity

```
(((a || b) && c) || d) && e
```

Test Case	a	b	c	d	e
(1)	T	—	T	—	T
(2)	F	T	T	—	T
(3)	T	—	F	T	T
(4)	F	T	F	T	T
(5)	F	F	—	T	T
(6)	T	—	T	—	F
(7)	F	T	T	—	F
(8)	T	—	F	T	F
(9)	F	T	F	T	F
(10)	F	F	—	T	F
(11)	T	—	F	F	—
(12)	F	T	F	F	—
(13)	F	F	—	F	—

Modified condition/decision (MC/DC)

- The most critical (Level A) software for aircrafts and related control systems must satisfy a level of coverage called **modified condition/decision coverage** (MC/DC).
- It is used in the standard **DO-178B** to ensure that Level-A software is tested adequately.
- To satisfy the MC/DC coverage criterion, during testing all of the below must be true at least once:
  1. Each entry and exit point is invoked
  2. Each decision tries every possible outcome.
  3. Each basic condition in a decision takes on every possible outcome.
  4. Each basic condition in a decision is shown to independently affect the outcome of the decision.

(Excerpt from Wikipedia. Content based on NASA's "A Practical Tutorial on Modified Condition/Decision Coverage".)

Modified condition/decision (MC/DC)

- To check that each basic condition in a decision <u>independently affects</u> the outcome of the decision:
  - for each basic condition C, there must be two test cases in which the truth values of all evaluated basic conditions other than C are the same, and the decision as a whole evaluates to True for one of those test cases and False for the other.

Modified condition/decision (MC/DC)

- N+1 test cases for N basic conditions

```
(((a || b) && c) || d) && e
```

Test Case	a	b	c	d	e	outcome
(1)	T	--	T	--	T	T
(2)	F	T	T	--	T	T
(3)	T	--	F	T	T	T
(6)	T	--	T	--	F	F
(11)	T	--	F	F	--	F
(13)	F	F	--	F	--	F

Modified condition/decision (MC/DC)

- MC/DC is
  - Basic Condition/Decision Coverage (C/DC)
  - plus one additional condition:
    every condition in a decision must *independently affect* the decisions output
- It is subsumed by multiple condition coverage.
- The MC/DC criterion can be satisfied with N + 1 test cases, making it an attractive compromise between number of required test cases and thoroughness of the test.
- It is required by important quality standards in aviation, including RTCA/DO-178B, "Software Considerations in Airborne Systems and Equipment Certification," and its European equivalent EUROCAE ED-12B.

Decision Tables

	Rule 1	Rule 2	...	Rule p
Conditions				
• C1				
• C2				
• ...				
• Cm				
Actions				
• A1				
• A2				
• ...				
• An				

Constraints : .....

- **Decision tables** are often used to represent business rules based on set of conditions.
- C1 ... Cm are various **conditions**.
- A1 ... An are **actions** to be performed depending on combination of input conditions
- Each of the **rules** defines a combination of the value of each input condition and all the actions that should be performed.
- There could be additional **constraints** on possible combinations of values of the input conditions.

Decision Tables

- **Limited entry table :** Each condition has 2 possible values: T or F, 0 or 1, Yes or No, etc.

	Rule 1	Rule 2	Rule 3	Rule 4
C1: Has outstanding debts	T	F	F	F
C2: Is an individual customer	-	T	F	F
C3: Is a business customer	-	F	T	F
C4: Is an education institute	-	F	F	T
A1: Discount rate	0%	10%	10%	15%

Constraints: Exactly one of C2, C3, and C4 is true.

- We say that a rule applies in a certain situation if the value of each condition is as specified in the rule.
- A '**-**' entry (called **don't care**) in a condition of a rule means that that condition can be omitted when considering that rule. In other words, it means that such condition can take any value (e.g. T or F) as long as the constraints are satisfied.

Decision Tables

- **Extended entry table :** Each condition can have more than two possible values.

- Suppose a customer can be of one the 3 types: Individual, Business, or Education. The previous table can be rewritten as an extended entry table as follows:

	Rule 1	Rule 2	Rule 3	Rule 4
C1: Has outstanding debts	T	F	F	F
C2: Customer type	-	Individual	Business	Education
A1: Discount rate	0%	10%	10%	15%

Decision Tables

- Constraints can be integrated into the table as extra rules. This is useful when we would like to check whether every possible combination of conditions is covered by some rule.
- In the previous table, there's a constraint that exactly one of the conditions C2, C3, and C4 is true. We rewrite the table by adding a new action called "**Impossible**" which is true if and only if the combination of conditions in a rule violates the constraints.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
C1: Has outstanding debts	T	F	F	F	-	-	-	-
C2: Is an individual customer	-	T	F	F	T	T	-	F
C3: Is a business customer	-	F	T	F	T	-	T	F
C4: Is an education institute	-	F	F	T	-	T	T	F
Impossible					X	X	X	X
A1: Discount rate	0%	10%	10%	15%				

- Note: Since the conditions of Rules 5 – 8 do not satisfy the constraints, Rules 5 – 8 will never be applied in practice. They are only listed here to make the table complete.

Decision Tables

- How are conditions in a decision table interpreted with respect to a program?
  - Conditions are interpreted as
    - Inputs
    - Test characteristics or parameters
- How are actions in a decision table interpreted with respect to a program?
  - Actions are interpreted as
    - Expected output
    - Expected operations to be performed
- What is the relationship between decision tables and test cases?
  - Basically, a test case can be derived from each rule (omitting the rules which are impossible).

Decision Tables

Considerations when constructing a decision table:

- Are there conflicting rules?
- Are there redundant rules? Non-conflicting redundant rules are harmless, but you should check carefully whether they are mistakes.
- Are there missing rules?

	Rule 1	Rule 2	Rule 3
C1: Has debt	F	F	F
C2: student	T	-	F
C3: member	-	T	T
A1	10%	20%	20%

In this table,

- Rule 1 and Rule 2 conflict, because when C1 is F and both C2 and C3 are T, the actions prescribed by the two rules differ;
- Rule 3 is redundant as it's already implied by Rule 2;
- Since there're many combinations of conditions not covered by these three rules, there must be some missing rules. For example, there should be one or more rules which apply when C1 is T and a rule which applies when C2 and C3 are both F.

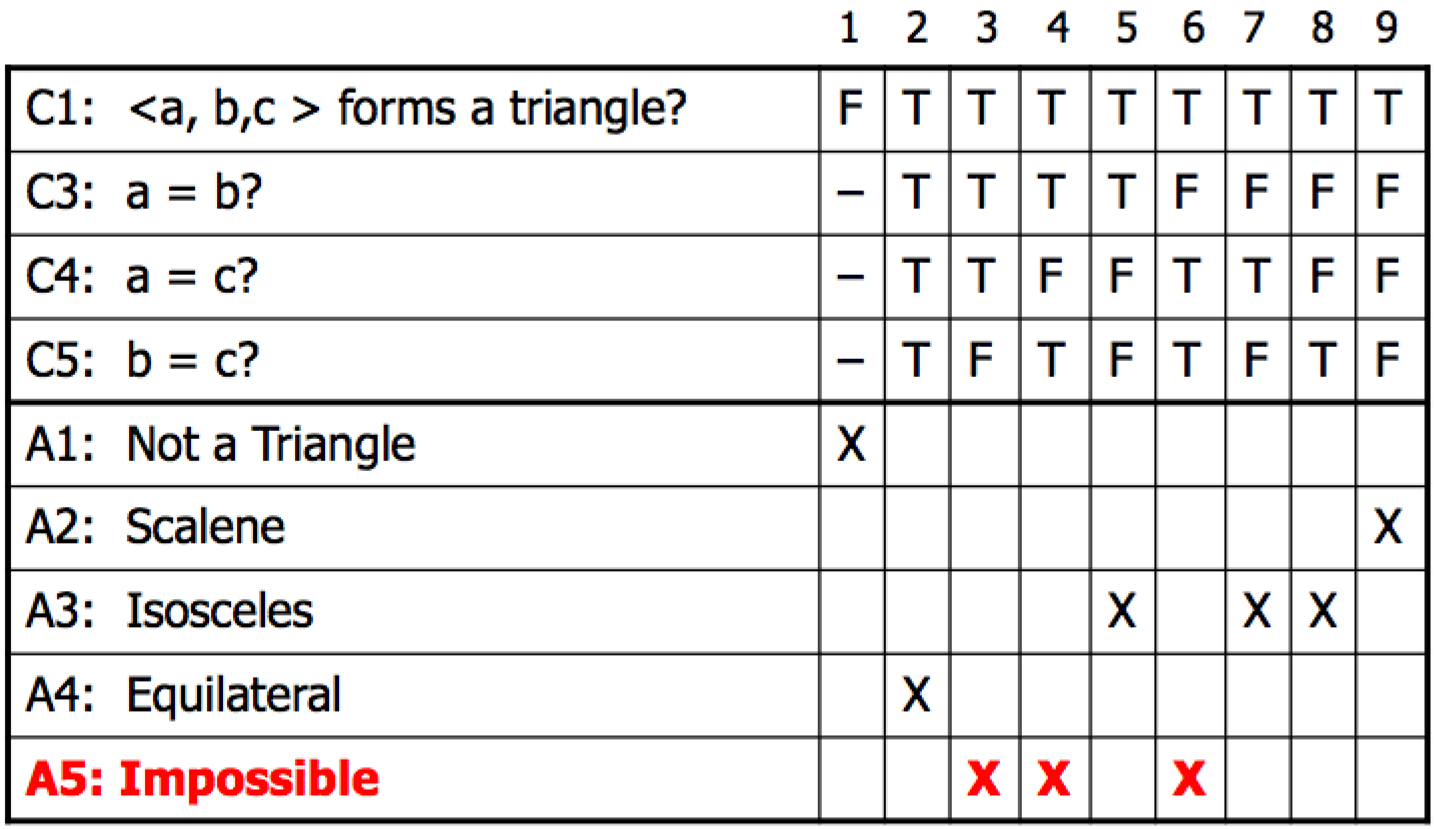
Decision Tables

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	Check 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

Decision Tables

- **Triangle Type Problem**: Given three positive integers, determine whether the integers are possibly the length of the three sides of a triangle or not and, if so, which triangle type it is.

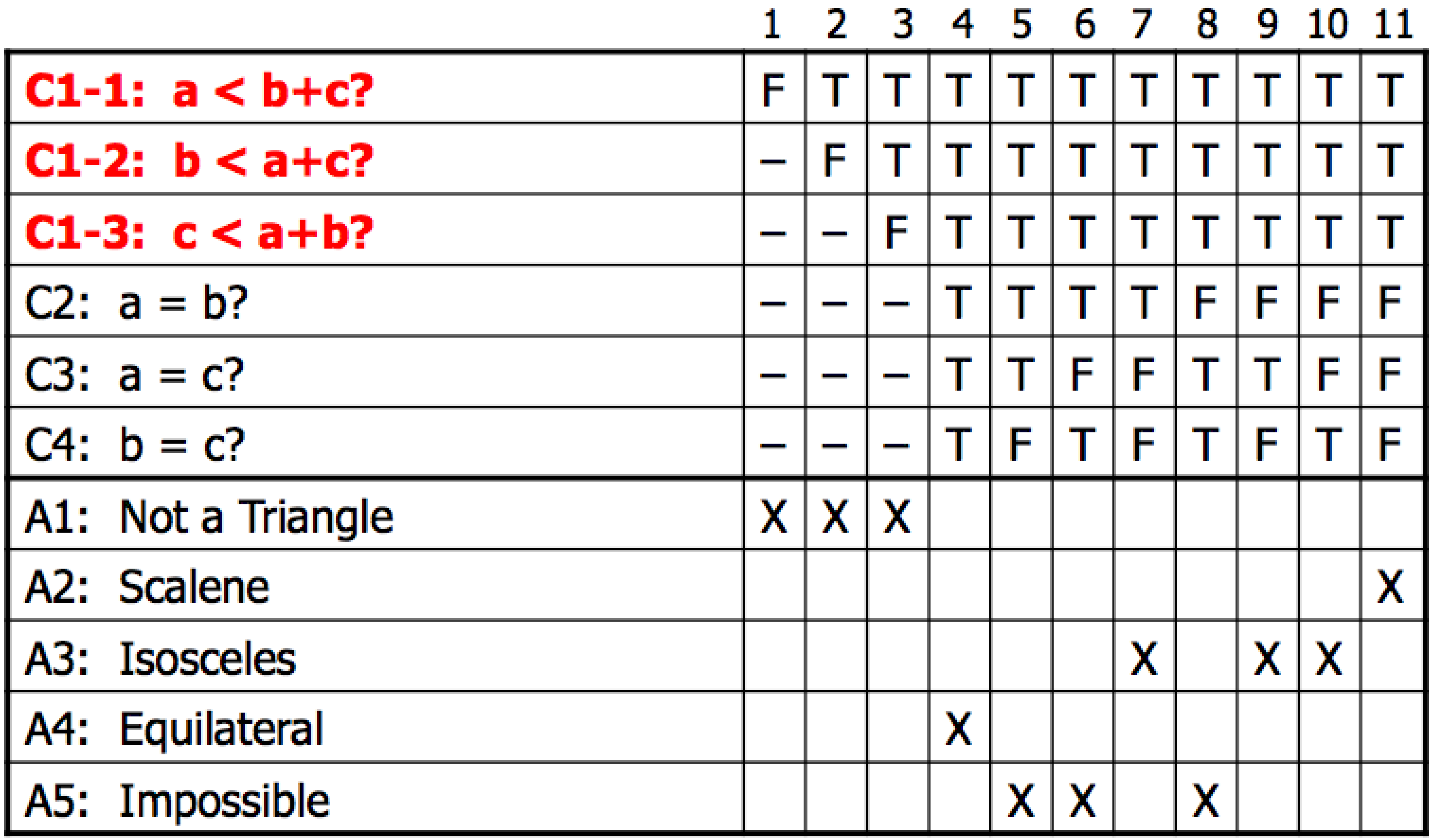


	1	2	3	4	5	6	7	8	9
C1: <a, b,c > forms a triangle?	F	T	T	T	T	T	T	T	T
C3: a = b?	–	T	T	T	T	F	F	F	F
C4: a = c?	–	T	T	F	F	T	T	F	F
C5: b = c?	–	T	F	T	F	T	F	T	F
A1: Not a Triangle	X								
A2: Scalene									X
A3: Isosceles					X		X	X	
A4: Equilateral		X							
A5: Impossible			**X**	**X**		**X**			

Action added by a tester showing impossible rules

Decision Tables

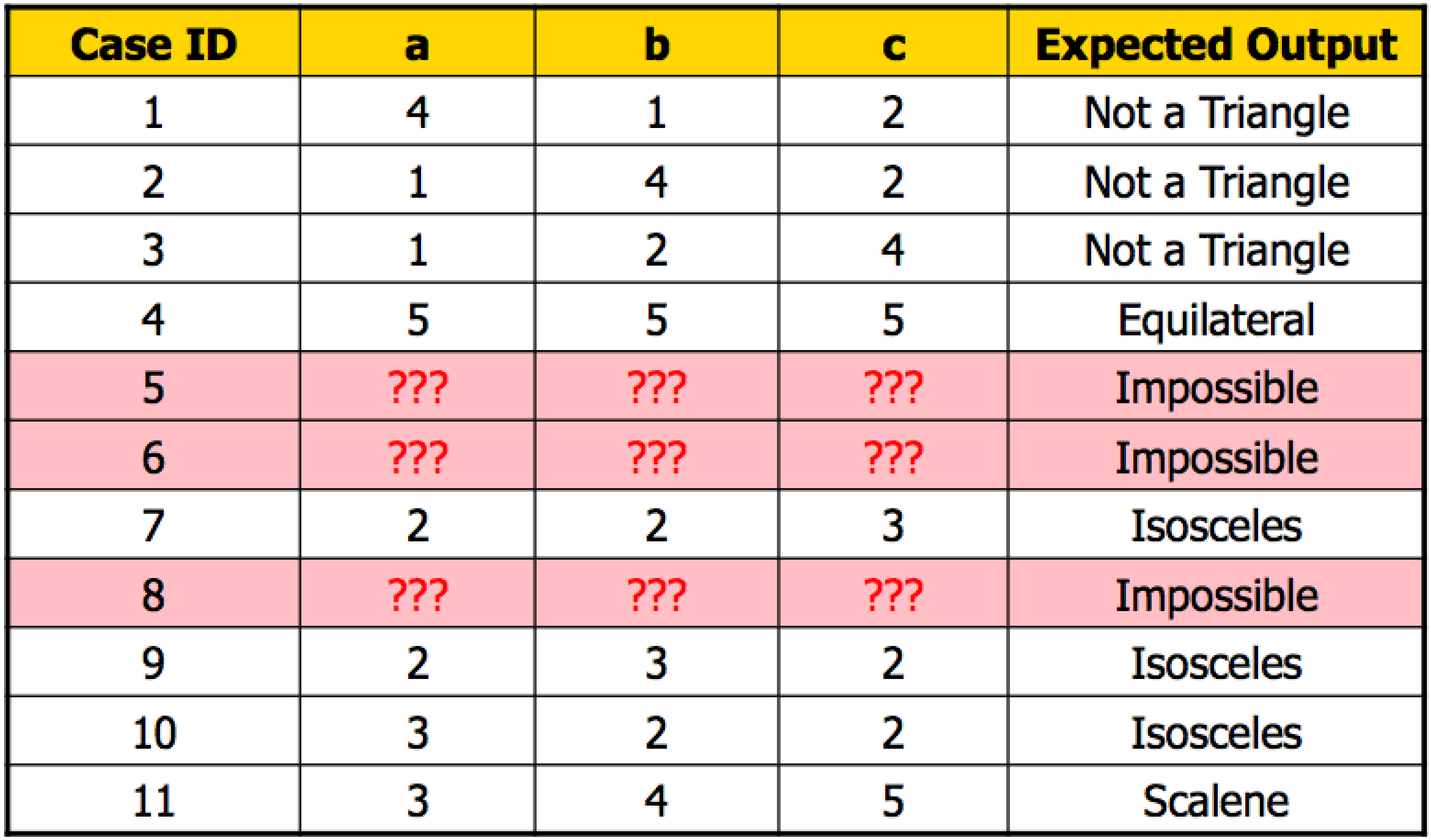
- A more refined decision table



	1	2	3	4	5	6	7	8	9	10	11
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			

Decision Tables

- Extracting test cases from each rule, omitting the rules that are impossible.



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

Decision Tables

- Recall the NextDate problem:
  - Given a date during 1/1/1812 – 31/12/2012, the program should output the next date.

- The **NextDate** problem illustrates the problem of dependencies in the input
  - Decision tables can highlight such dependencies.
  - Impossible dates can be clearly marked as separate rules.

Decision Tables

First 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) }`

Decision Tables

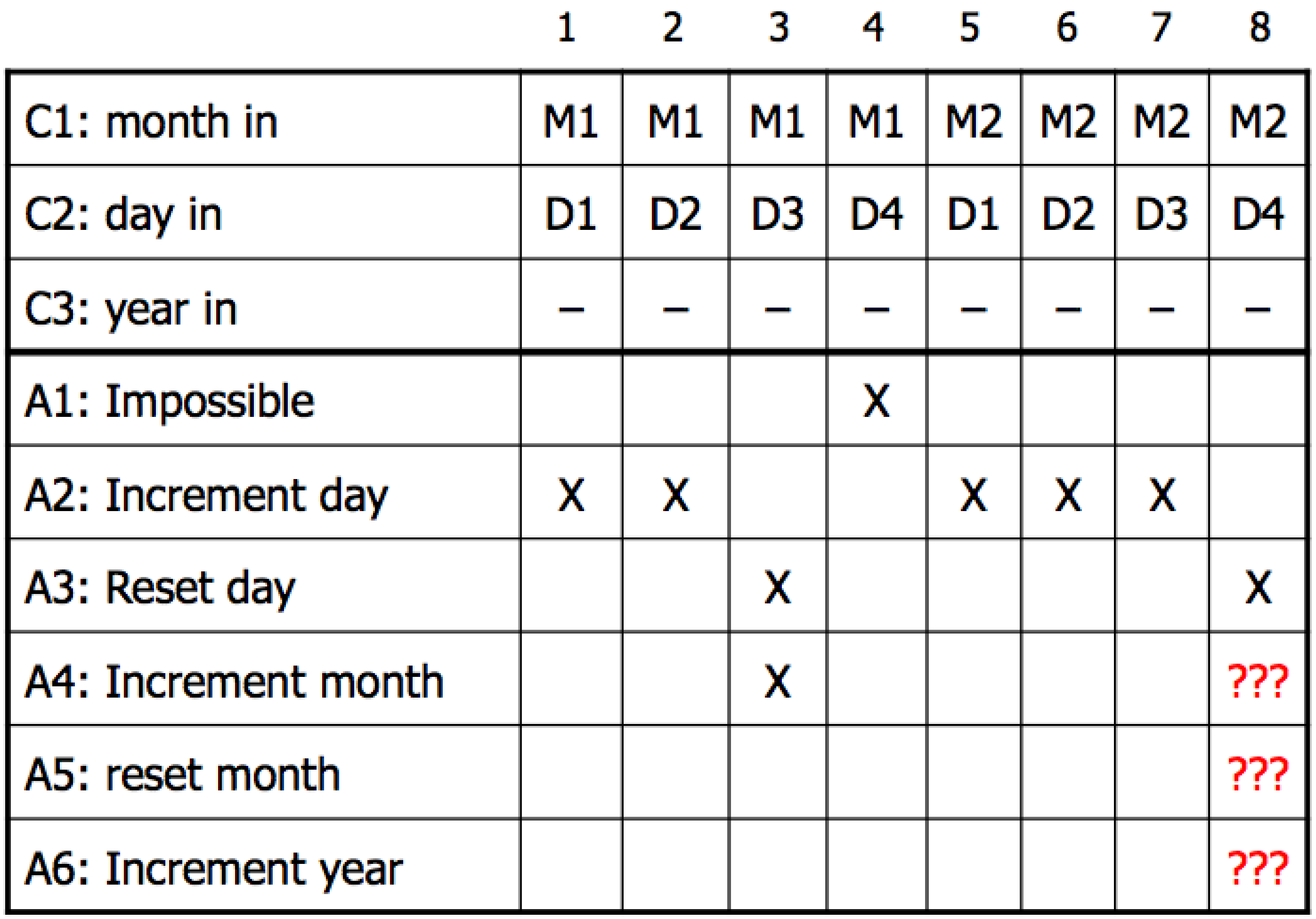
C1: month in M1?	T	T	T	T	T	T	T	T				
C2: month in M2?									T	T	T	T
C3: month in M3?												
C4: day in D1?	T	T							T	T		
C5: day in D2?			T	T							T	T
C6: day in D3?					T	T						
C7: day in D4?							T	T				
C8: year in Y1?	T		T		T		T		T		T	
C9: year in Y2?		T		T		T		T		T		T
A1: Impossible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

Decision Tables

Second 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) ∧`
  `year ≠  2000 }`
- `Y3 = {year : 1812 .. 2012 | common_year (year) }`

Decision Tables



	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	–	–	–	–	–	–	–	–
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

Decision Tables

	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	–	–
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								

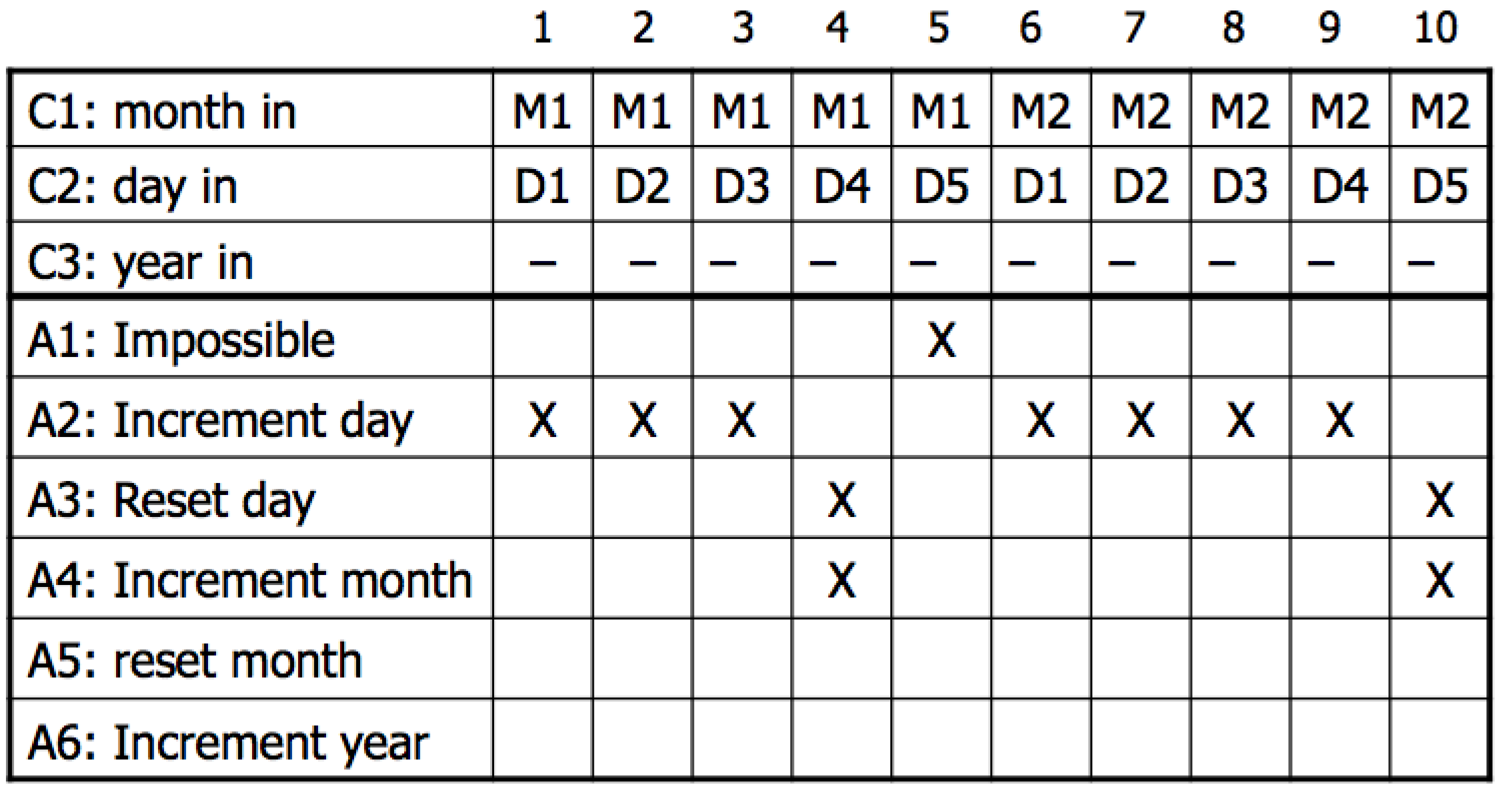
Decision Tables

Third 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) }`

Decision Tables

A 22 rule table



	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	–	–	–	–	–	–	–	–	–	–
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 Tables

	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	–	–
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							

Decision Tables

- **Basic Condition Coverage Criterion**: For each rule, there is at least one test case in which the rule applies.

  Note: Each don’t care entry ‘-’ can be chosen to be any value as long as the constraints are not violated.

- **Multiple Condition Coverage Criterion:** For each possible combination of conditions not violating the constraints, there is at least one test case satisfying such combination of conditions.

  Note: For a limited entry table with N conditions, there could be up to 2^N possible combinations of conditions. So up to 2^N test cases may be needed to satisfy compound condition coverage.

Decision Tables

Exercise:

- An online shopping website sends a selection of product brochures (Sporting Goods, DIY Goods, Clothing, Children Items, and Electronic Items) based on the customer’s age, gender (male or female), and marital status (married or single) according to the following rules:
  1. All male customers will be sent a brochure of “Sporting Goods”.
  2. All male customers of age between 30 and 50 will be sent a brochure of “DIY Goods”.
  3. All female customer will be sent a brochure of “Clothing”.
  4. All married customers of age below 40 will be sent a brochure of “Children Items”.
  5. All customer of age below 40 will be sent a brochure of “Electronic Items ”.

- Draw a decision table for the rules above.