Decision Table

By
RESHMI MAULIK

DECISION TABLE

Decision tables are used to model complicated programming logic. They can make it easy to see that all possible combinations of conditions have been considered.

- Decision table provides a handy and compact way to represent complex business logic.
- In a decision table, business logic is well divided into conditions, actions (decisions) and rules for representing the various components that form the business logic.
- The tables are composed of 4 parts: conditions, actions, condition alternatives (each column is a rule), and actions for the rules.

Definition

- Decision tables are used to lay out in tabular form all possible situations which a business decision may encounter.
- A decision table lists causes and effects in a matrix. Each column represents a unique combination.
- Purpose is to structure logic

		Combinations							
Causes	**Values**	1	2	3	4	5	6	7	8
Cause 1	*Y, N*	Y	Y	Y	Y	N	N	N	N
Cause 2	*Y, N*	Y	Y	N	N	Y	Y	N	N
Cause 3	*Y, N*	Y	N	Y	N	Y	N	Y	N
Effects									
Effect 1		X			X				X
Effect 2			X				X		X

Cause = condition

Effect = action = expected results

What is a decision table ?

- Table representing complete set of conditional expressions
- where expressions are mutually exclusive in a predefined area

Why use decision tables ?

Powerful visualisation

- Compact and structured presentation

Preventing errors is easier

- Avoid incompleteness and inconsistency

Modular knowledge organisation

- Group related rules into single table
- Combine tables to achieve decision

Let's take an example scenario for an ATM where a decision table would be of use.

- A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted.

A decision table makes the same requirements clearer to understand

Conditions	**R1**	**R2**	**R3**
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

In a decision table, **conditions are usually expressed as true (T) or false (F)**

- Above table contains three different business rules, and one of them is the "withdrawal is granted if the requested amount is covered by the balance."
- It is normal to create at least one test case per column, which results in full coverage of all business rules.

Step 1 – Analyze the requirement and create the first column

Requirement:

"Withdrawal is granted if requested amount is covered by the balance or if the customer is granted credit to cover the withdrawal amount".

Express conditions and resulting actions in a list so that they are either TRUE or FALSE.

In this case there are two conditions, "withdrawal amount ≤ balance" and "credit granted".

There is one result, the withdrawal is granted.

Conditions
Withdrawal Amount <= Balance
Credit granted
Actions
Withdrawal granted

Step 2 : add Colum

- Calculate how many columns are needed in the table. The number of columns depends on the number of conditions and the number of alternatives for each condition.
- If there are two conditions and each condition can be either true or false, you need 4 columns. If there are three conditions there will be 8 columns and so on.
- Mathematically, the number of columns is $2^{\text{conditions}}$.
- In this case $2^2 = 4$ columns.

Number of columns that is needed:

Number of Conditions	Number of Columns
1	2
2	4
3	8
4	16
5	32

Now is the time to fill in the T (TRUE) and F (FALSE) for the conditions

- How do you do that? The simplest is to say that it should look like this:
- Row 1: TF
- Row 2: TTFF
- Row 3: TTTTFFFF
- For each row, there is twice as many T and F as the previous line.
- Repeat the pattern above from left to right for the entire row.
- In other words, for a table with 8 columns, the first row will read TFTFTFTF, the second row will read TTFFTTFF and the third row will read TTTTFFFF.

Step 3: Reduce the table

- Mark insignificant values with “-”.
- If the requested amount is less than or equal to the account balance it does not matter if credit is granted.
- In the next step, you can delete the columns that have become identical.

Conditions				
Withdrawal Amount <= Balance	T	F	T	F
Credit granted	-	T	-	F
Actions				
Withdrawal granted				

Conditions				
Withdrawal Amount <= Balance	T	F	T	F
Credit granted	T	T	F	F
Actions				
Withdrawal granted				

Check for invalid combinations

- Invalid combinations are those that cannot happen, for example, that someone is both an infant and senior. Mark them somehow, e.g. with "X". In this example, there are no invalid combinations.
- Finish by removing duplicate columns.
- In this case, the first and third column are equal, therefore one of them is removed.

Step 4: determine action

- Enter actions for each column in the table.
- You will be able to find this information in the requirement.
- Name the columns (the rules).
- They may be named R1/Rule 1, R2/Rule 2 and so on, but you can also give them more descriptive names

Conditions			
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Step 5: Write test cases

- Write test cases based on the table. At least one test case per column gives full coverage of all business rules.
- Test case for R1: balance = 200, requested withdrawal = 200. Expected result: withdrawal granted.
- Test case for R2: balance = 100, requested withdrawal = 200, credit granted. Expected result: withdrawal granted.
- Test case for R3: balance = 100, requested withdrawal = 200, no credit. Expected Result: withdrawal denied.

- **Advantage of using decision tables**

  is that they **make it possible to detect combinations of conditions that would otherwise not have been found**

- **A disadvantage of the technique**

  is that **a decision table is not equivalent to complete test cases** containing step-by-step instructions of what to do in what order

  However, if we have a lot of combinations, it may not be possible or sensible to test every combination.