Unit 2

Black box testing techniques

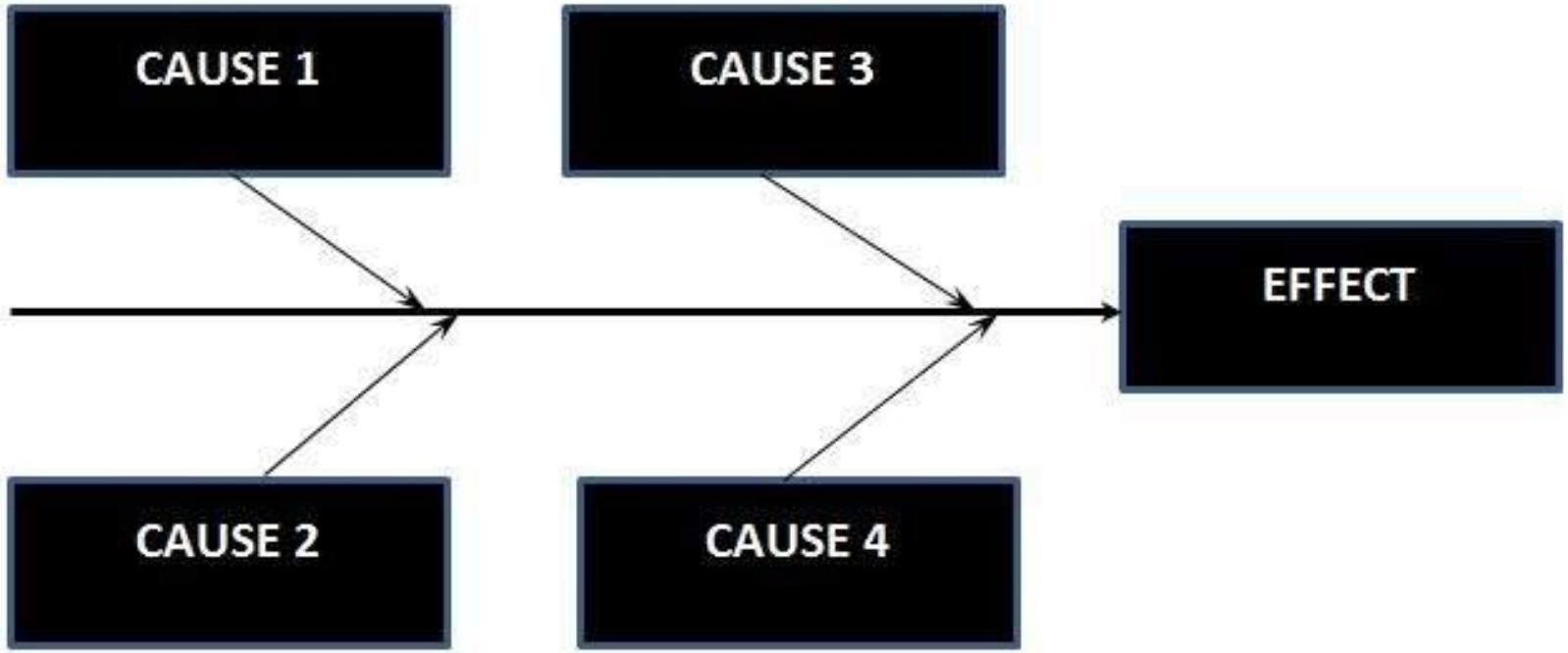
Black box testing

- cause-effect graphing
- error guessing
- decision table testing
- use case testing
- State transition testing

What is Cause-Effect Graph?

- Cause Effect Graph is a black box testing technique that graphically illustrates the relationship between a given outcome and all the factors that influence the outcome.
- It is also known as Ishikawa diagram as it was invented by Kaoru Ishikawa or fish bone diagram because of the way it looks.

Cause Effect - Flow Diagram



Circumstances - under which Cause-Effect Diagram used

- To Identify the possible root causes, the reasons for a specific effect, problem, or outcome.
- To Relate the interactions of the system among the factors affecting a particular process or effect.
- To Analyze the existing problems so that corrective action can be taken at the earliest

Benefits

- It Helps us to determine the root causes of a problem or quality using a structured approach.
- It Uses an orderly, easy-to-read format to diagram cause-and-effect relationships.
- It Indicates possible causes of variation in a process.
- It Identifies areas, where data should be collected for further study.
- It Encourages team participation and utilizes the team knowledge of the process.
- It Increases knowledge of the process by helping everyone to learn more about the factors at work and how they relate.

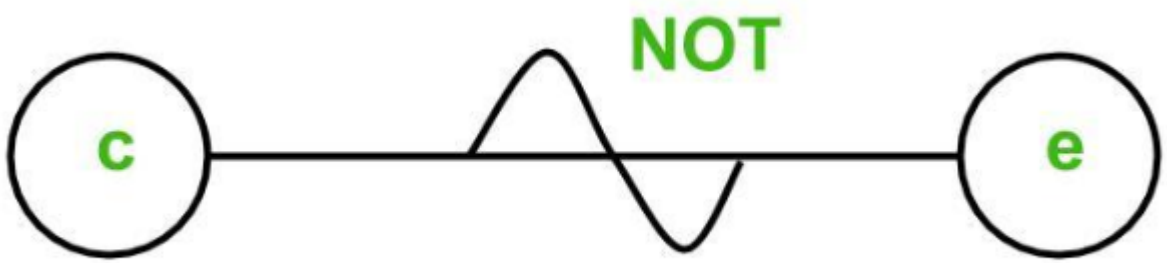
Steps for drawing cause-Effect Diagram

- Cause : Any distinct input condition or an equivalence class of input condition
- Effect: An output condition or system transformation (the effect that any input has on the state of system).
- Step 1: Identify the cause and effects from the specifications and assign unique numbers to each of them.
- Step 2: Draw a Boolean graph linking cause and effect.
- Step 3: Specify constraints on graph describing combinations of cause and/or effects that are impossible.
- Step 4: Convert the graph into limited entry decision table by tracing state conditions in the graph.

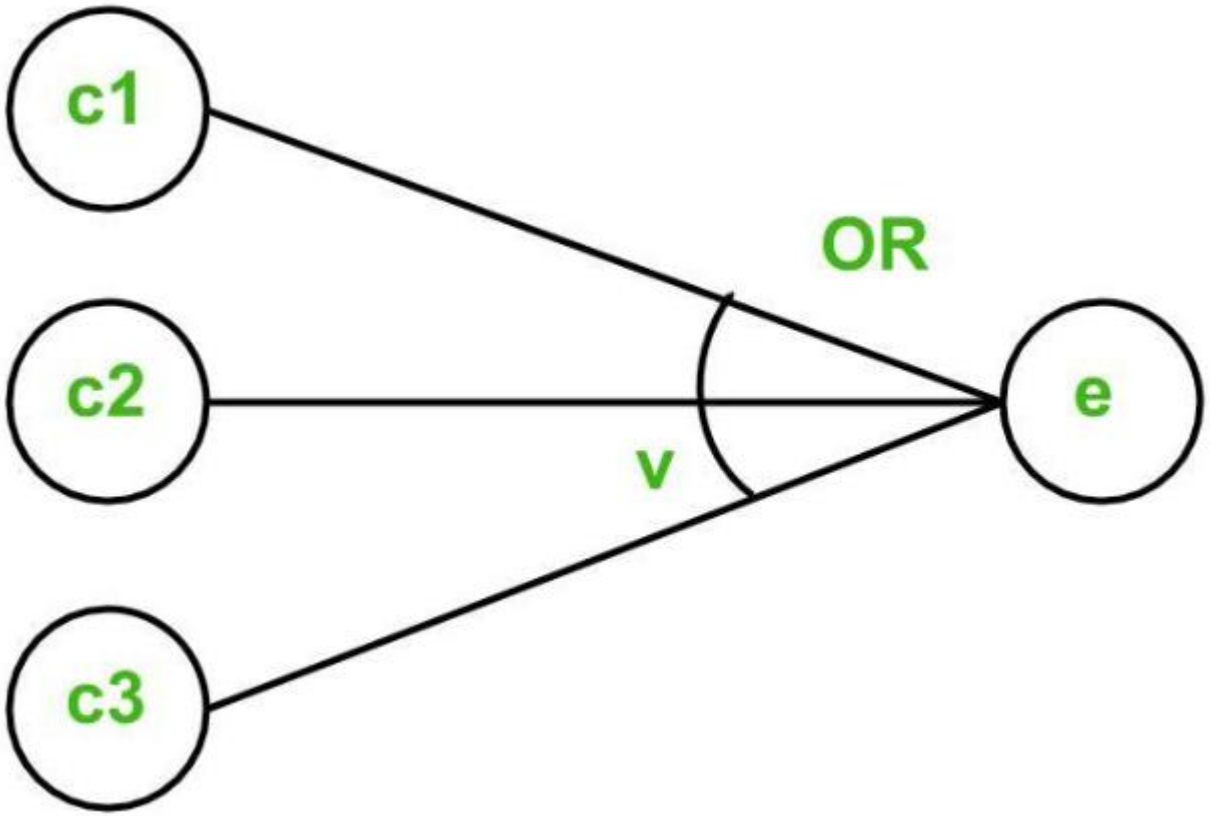
Cause effect notations

- **1. Identity function:**
- If c1 is true, e1 will also be true; otherwise e1 will be false
- If c1 is 1, e1 will also be 1; else e1 will be 0

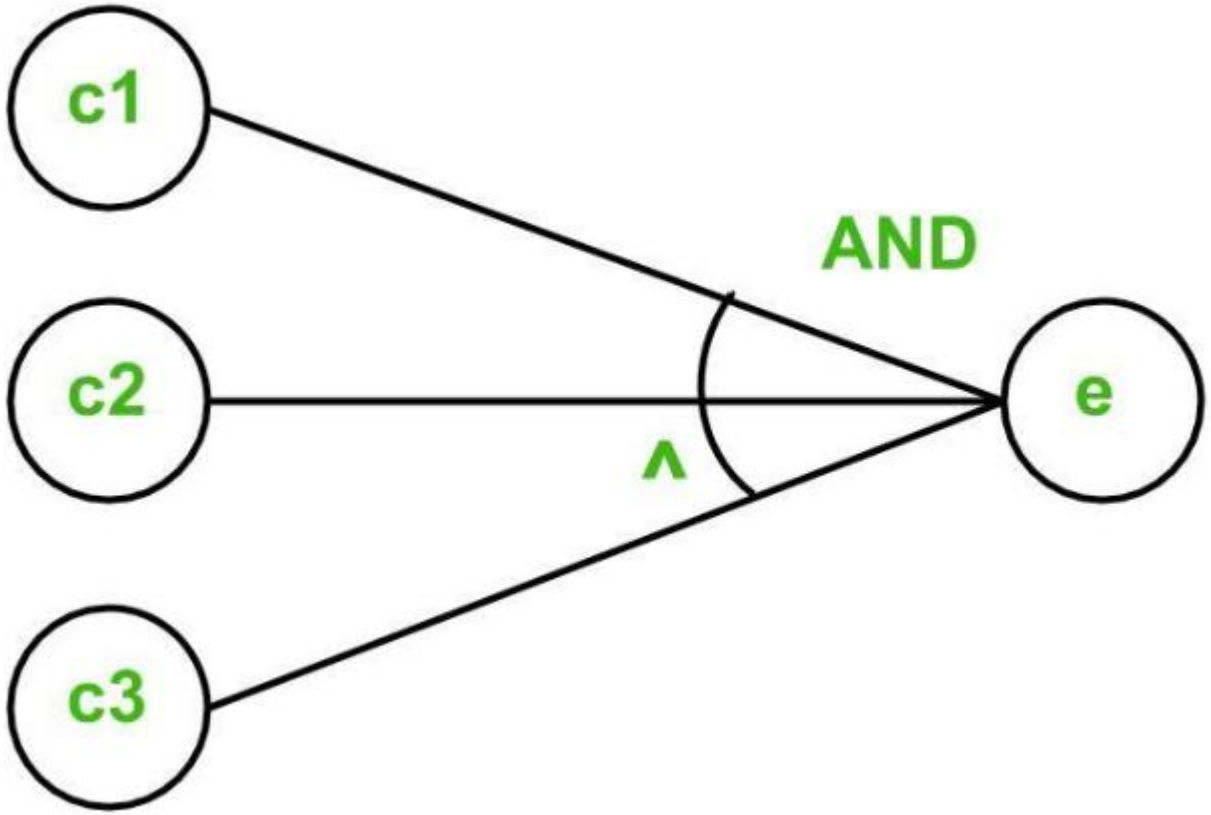
- **2. NOT Function**
- If c1 is 1 then e1 is 0; else e1 is 1
- If c1 exists, then e1 will not exist



- **3. OR Function:**
- Multiple cause with single effect
- If c1 OR c2 OR c3 is 1, then e1 is 1; else e1 is 0



- **4. AND Function:**
- Multi causes, single effect
- If both c1 AND c2 are 1, then e1 will be 1; else e1 will be 0.



Example

Q. The characters in a column 1 must be A or B. The character in column 2 must be a digit. If these hold then file update is possible. If the character in column 1 is incorrect, message x is issued. If character in column 2 is not a digit, message y is issued.

SOLUTION:

Causes are:

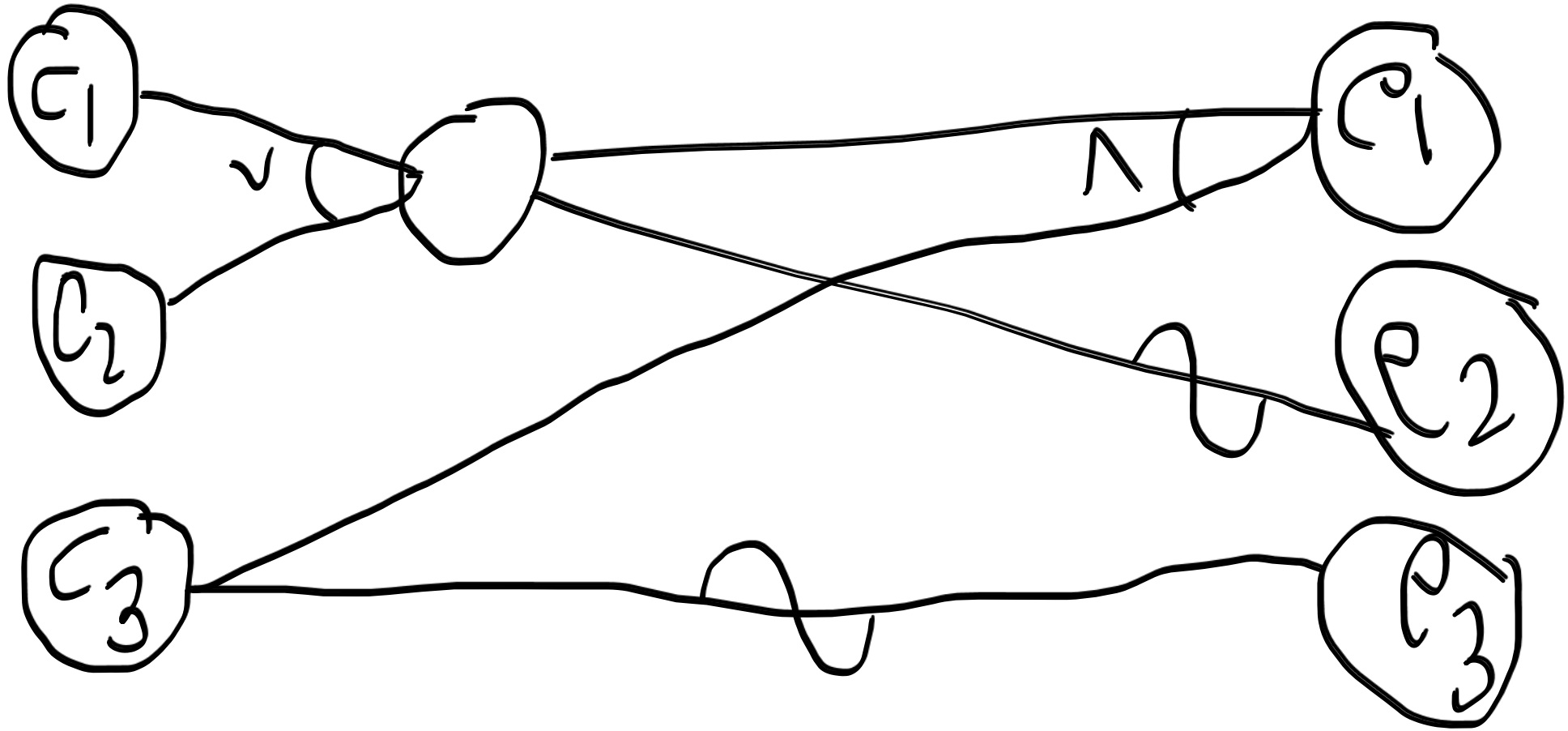
C1: character in column 1 is A

C2: character in column 1 is B

C3: character in column 2 is a digit

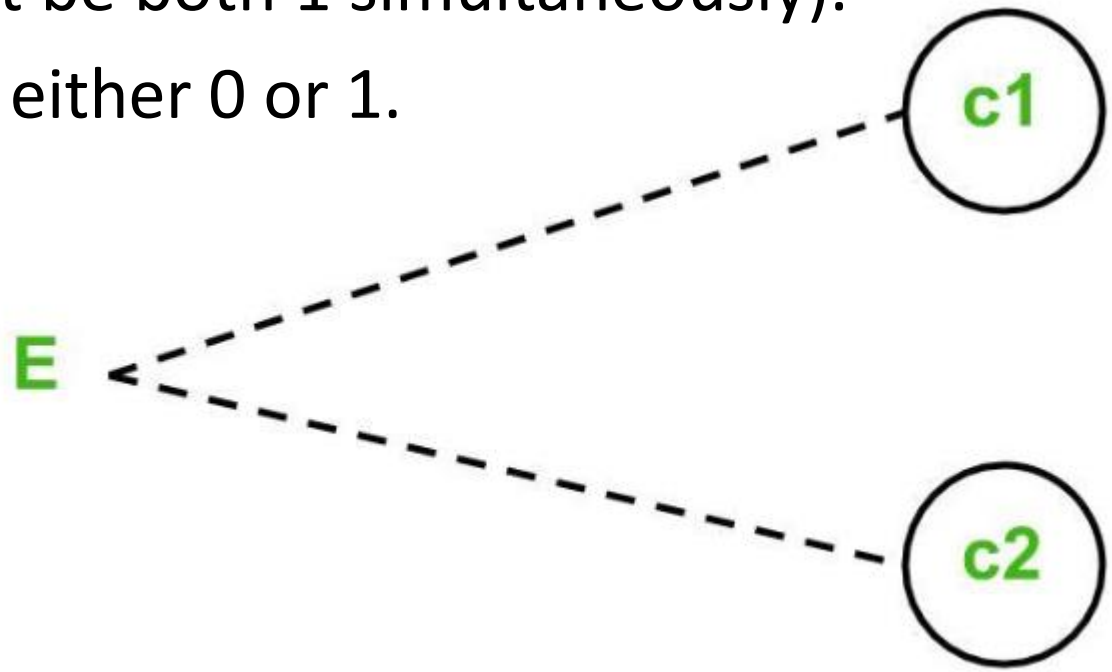
- **Effects are:**
- e1: update made -> (c1 V c2) ^ c3
- The update will be made if column 1 holds A or B and column 2 holds a digit. Therefore C1 OR C2 AND C3
- e2: message x is issued -> C1 ^C2
- The message x will be issued if Column 1 holds neither A or B. Therefore C1 not AND C2 not
- e3: message y is issued -> C3
- The message y will be issues if column 2 doesn't hold a digit. Therefore, C not.

Cause effect graph



Impossible constraints in the Cause effect graph

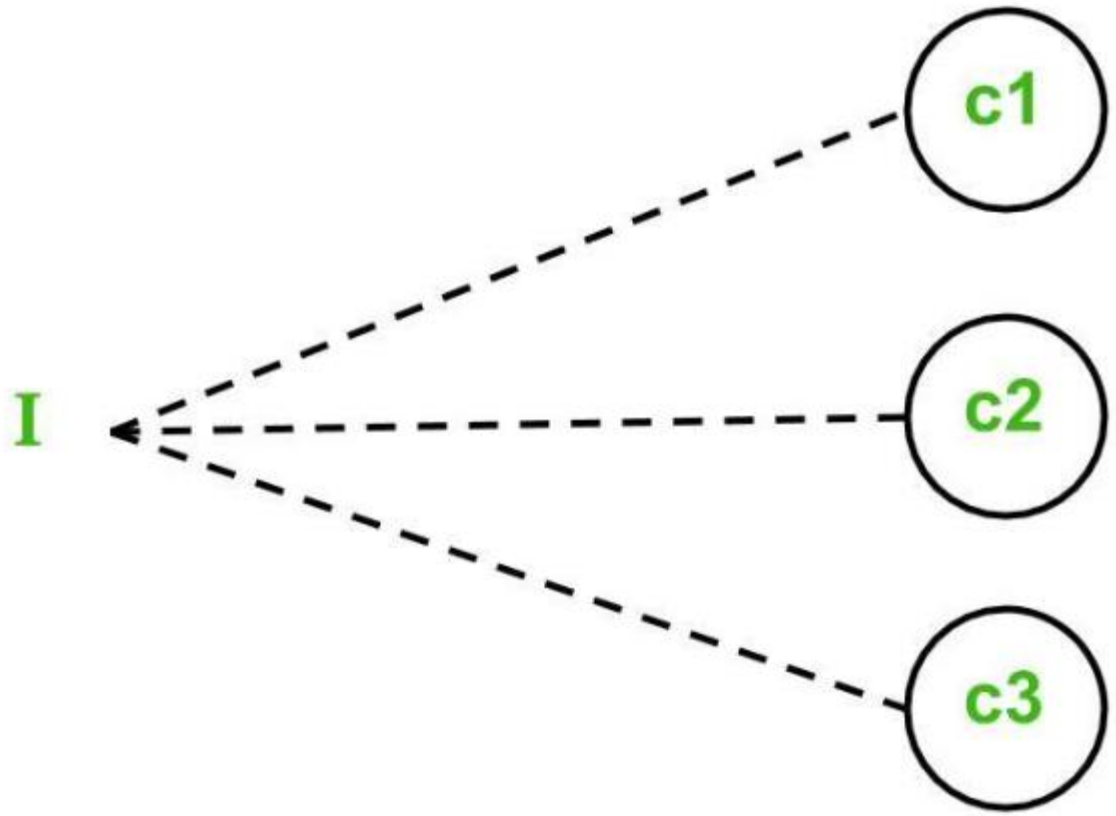
- To represent the impossible combination of causes few notations are:
- **1. E constraint:** It must always be true that at most one of c1 or c2 can be 1 (c1 and c2 cannot be both 1 simultaneously).
- Here, the causes could be either 0 or 1.
- E = exclusive



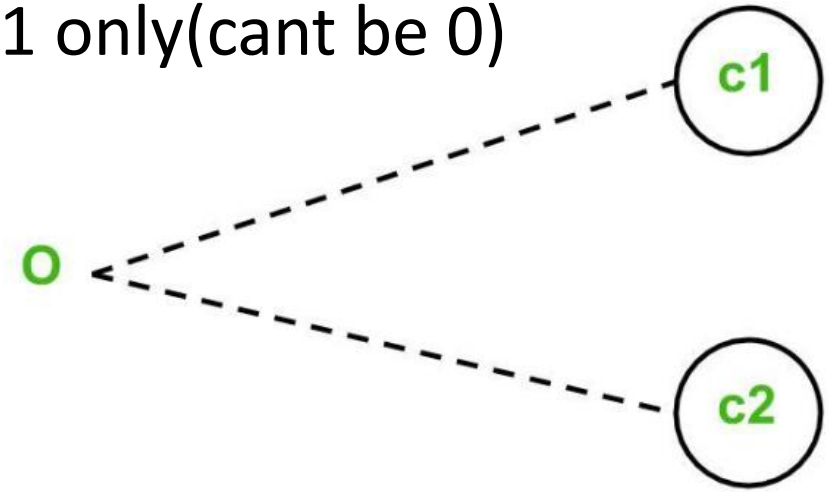
2. I constraint:

Atleast one of c1, c2 and c3 must always be 1 (c1,c2, and c3 cannot be all 0 simultaneously)

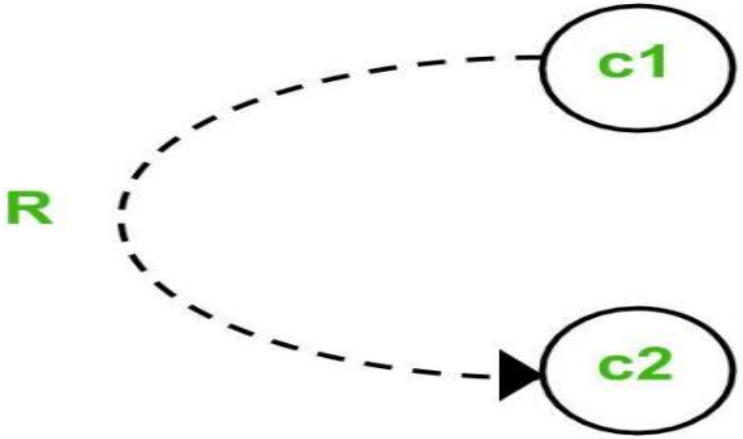
I = Inclusive



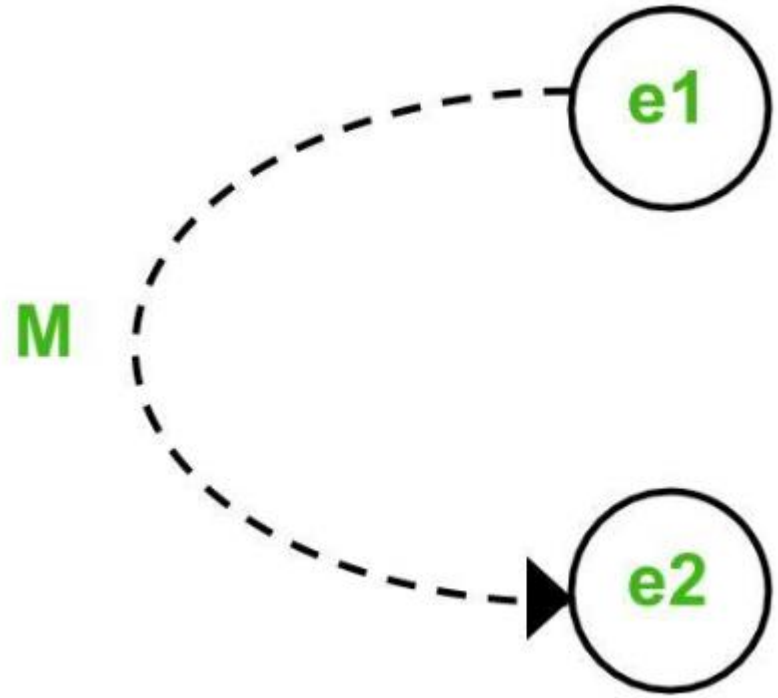
- **3. O Constraint:** One and only one out of C1 and C2 must be 1.
- Here, the cause would be 1 only(cant be 0)
- O= One and only one



4. R Constraint: States that for c1 to be 1, c2 must be 1. (it is impossible for c1 to be 1 and c2 to be 0) R = Requires

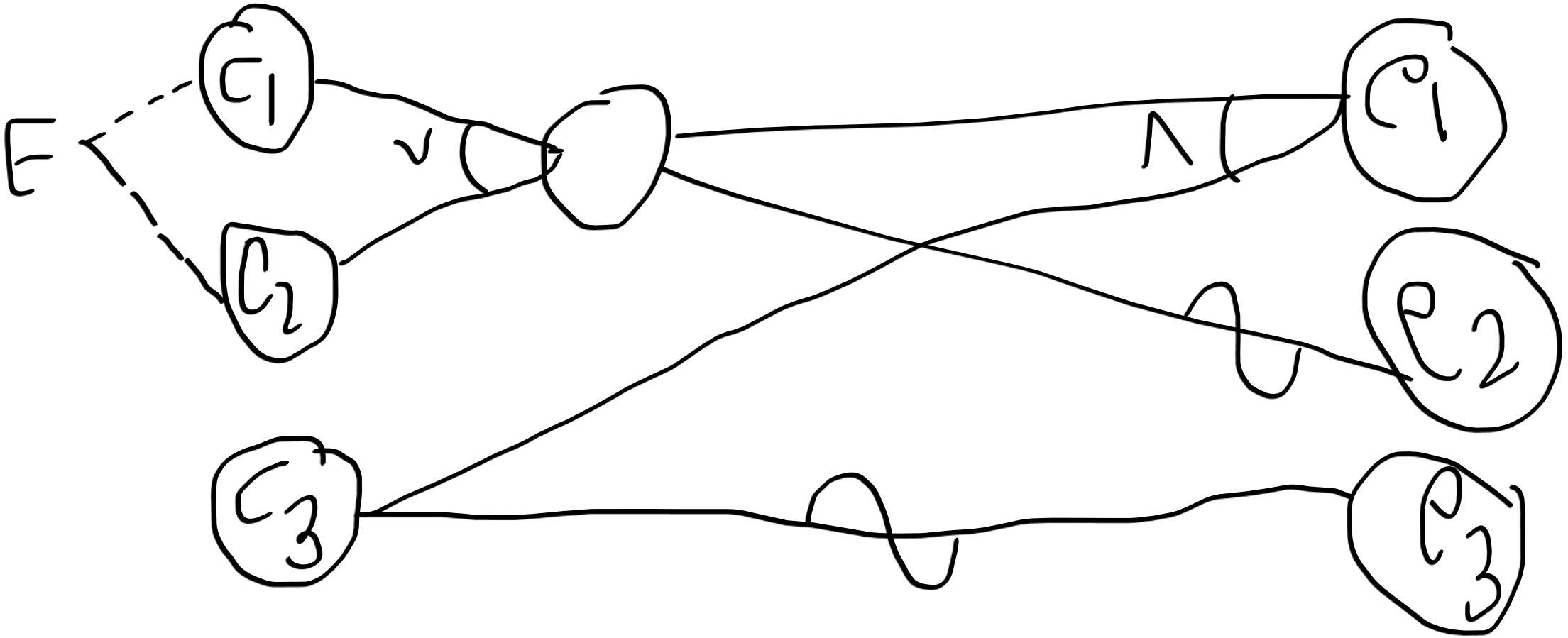


- **5. M Constraint:** States that if effect e1 is 1, effect e2 is forced to be 0.
- M= Mask



- In the previous example, the constraint was exclusive (E), as the column 1 can hold either A or B. Therefore, the constraint exclusive is applied at C1 and C2.

Cause effect graph with constraint



Example:

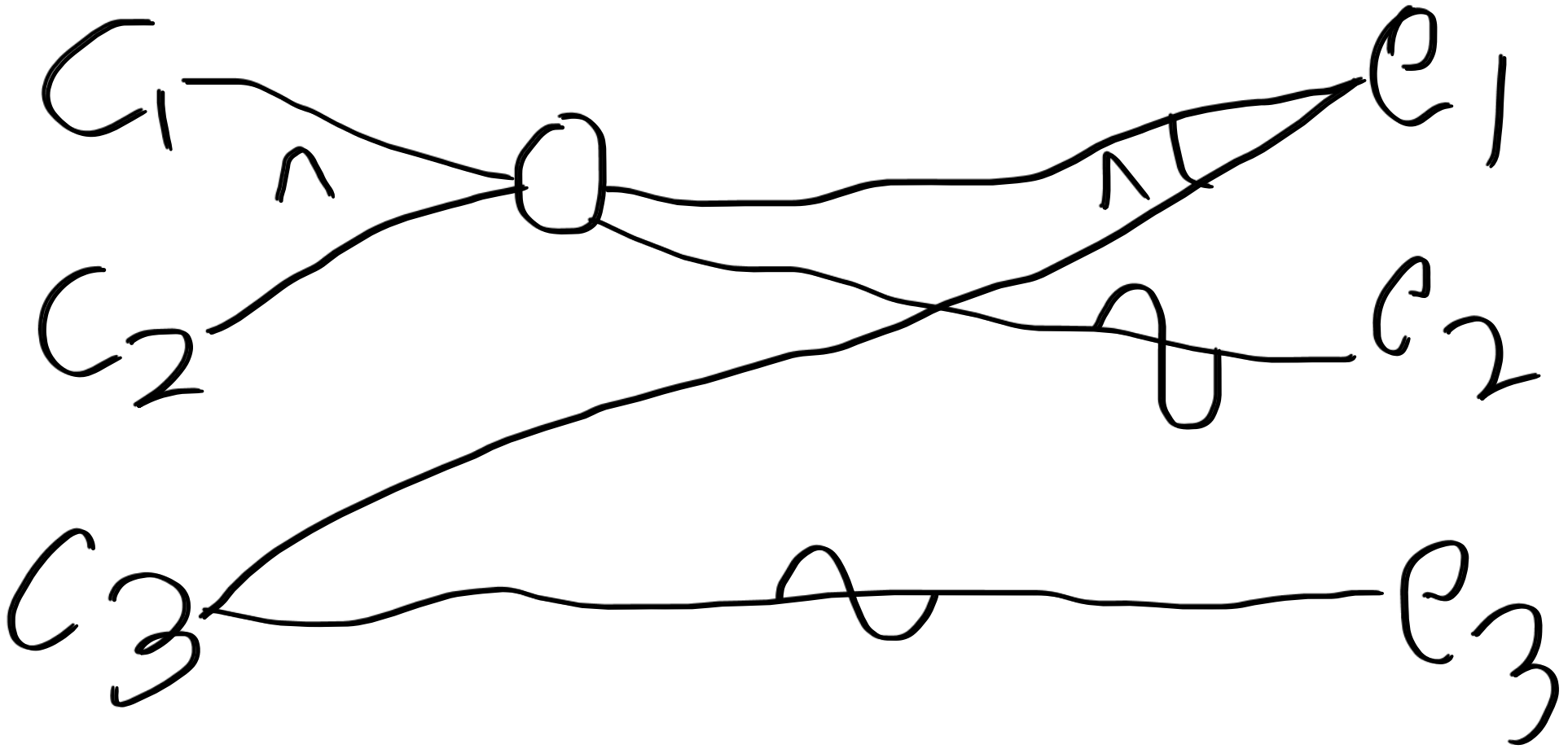
Q. If parameter X accepts numbers 1 and 0, and parameter Y accepts any alphabetical characters, the output will be ACCEPT PARAMETER. Otherwise, if parameter X is incorrect, message A is displayed, and if Parameter Y is incorrect, message B is displayed.

Solution

- C1 : character of parameter X is 1
- C2: character of parameter X is 0
- C3: character of parameter Y is any alphabet

- E1: ACCEPT PARAMETER
- E2: display message A
- E3: display message B

Solution Cause effect graph



Error Guessing Technique

- The test case design technique ensures that all the possible values that are both positive and negative are required for the testing purposes. In software testing, we have three different test case design techniques which are as follows:
- Error Guessing
- Equivalence Partitioning
- Boundary Value Analysis[BVA]

- The implementation of this technique depends on the experience of the tester or analyst having prior experience with similar applications. It requires only well-experienced testers with quick error guessing technique. This technique is used to find errors that may not be easily captured by formal black box testing techniques, and that is the reason, it is done after all formal techniques.
- The scope of the error guessing technique entirely depends on the tester and type of experience in the previous testing involvements because it does not follow any method and guidelines. Test cases are prepared by the analyst to identify conditions. The conditions are prepared by identifying most error probable areas and then test cases are designed for them.

The main purpose of this technique is to identify common errors at any level of testing by exercising the following tasks:

- Enter blank space into the text fields.
- Null pointer exception.
- Enter invalid parameters.
- Divide by zero.
- Use maximum limit of files to be uploaded.
- Check buttons without entering values.

Examples of Error guessing method

- A function of the application requires a mobile number which must be of 10 characters. Now, below are the techniques that can be applied to guess error in the mobile number field:
- What will be the result, if the entered character is other than a number?
- What will be the result, if entered characters are less than 10 digits?
- What will be the result, if the mobile field is left blank?

After implementing these techniques, if the output is similar to the expected result, the function is considered to be bug-free, but if the output is not similar to the expected result, so it is sent to the development team to fix the defects.

The benefits of error guessing technique are as follows:

- It is a good approach to find the challenging parts of the software.
- It is beneficial when we will use this technique with the grouping of other formal testing techniques.
- It is used to enhance the formal test design techniques.

Following are the drawbacks of error guessing technique:

- The error guessing technique is person-oriented rather than process-oriented because it depends on the person's thinking.
- If we use this technique, we may not achieve the minimum test coverage.
- With the help of this, we may not cover all the input or boundary values.
- With this, we cannot give the surety of the product quality.

Decision Table Testing

A decision table is the tabular representation of several input values, cases, rules, and test conditions. The Decision table is a highly effective tool utilized for both requirements management and complex software testing. Through this table, we can check and verify all possible combinations of testing conditions. The testers can quickly identify any skipped needs by reviewing the True(T) and False(F) values assigned for these conditions.

Decision Table Example 1

- In this example, we see how to create the decision table for a login screen that asks for UserId and Password.
- The condition here is that the user will be redirected to the homepage if he enters the correct user name and password, and an error message will be displayed if the input is wrong.
- **Legend:**
- T- Correct username or password
- F- Wrong username or password
- E- Error message is displayed.
- H – Home screen is displayed.

Decision table

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

Decision Table Interpretation

- Case 1: Username and Password both are wrong, and the user is shown an error message.
- Case 2: Username is correct, but the password is wrong, and the user is shown an error message,
- Case 3: The username is wrong, but the password is correct, and the user is shown an error message.
- Case 4: Username and password both are correct, and the user is taken to the homepage.

Decision Table Example 2

In this example, we consider the decision table and test scenarios for an Upload screen. There is a dialogue box that will ask the user to upload a photo with the following conditions:

- The file must be in the .jpg format.
- The file size must be less than 32kb.
- The image resolution must be 137*177.

If any one of the above conditions fails, the system will display the corresponding error messages about the issue. If all conditions are satisfied, the photo will be uploaded successfully.

SOLUTION EXAMPLE 2

Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.JPG	.JPG	.JPG	.JPG	Not.JPG	Not.JPG	Not.JPG	Not.JPG
Size	< 32 kb	< 32 kb	>= 32 kb	>=32 kb	< 32 kb	< 32 kb	>= 32 kb	>= 32 kb
Resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded successfully	Error message due to resolution mismatch	Error message due to size mismatch	Error message due to size and resolution mismatch	Error message due to format mismatch	Error message due to format and resolution mismatch	Error message due to format and size mismatch	Error message due to format, size and resolution mismatch

Use Case Testing

Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.

- Use Cases capture the interactions between 'actors' and the 'system'.
- 'Actors' represents user and their interactions that each user takes part into.
- Test cases based on use cases and are referred as scenarios.
- Capability to identify gaps in the system which would not be found by testing individual components in isolation.
- Very effective in defining the scope of acceptance tests.

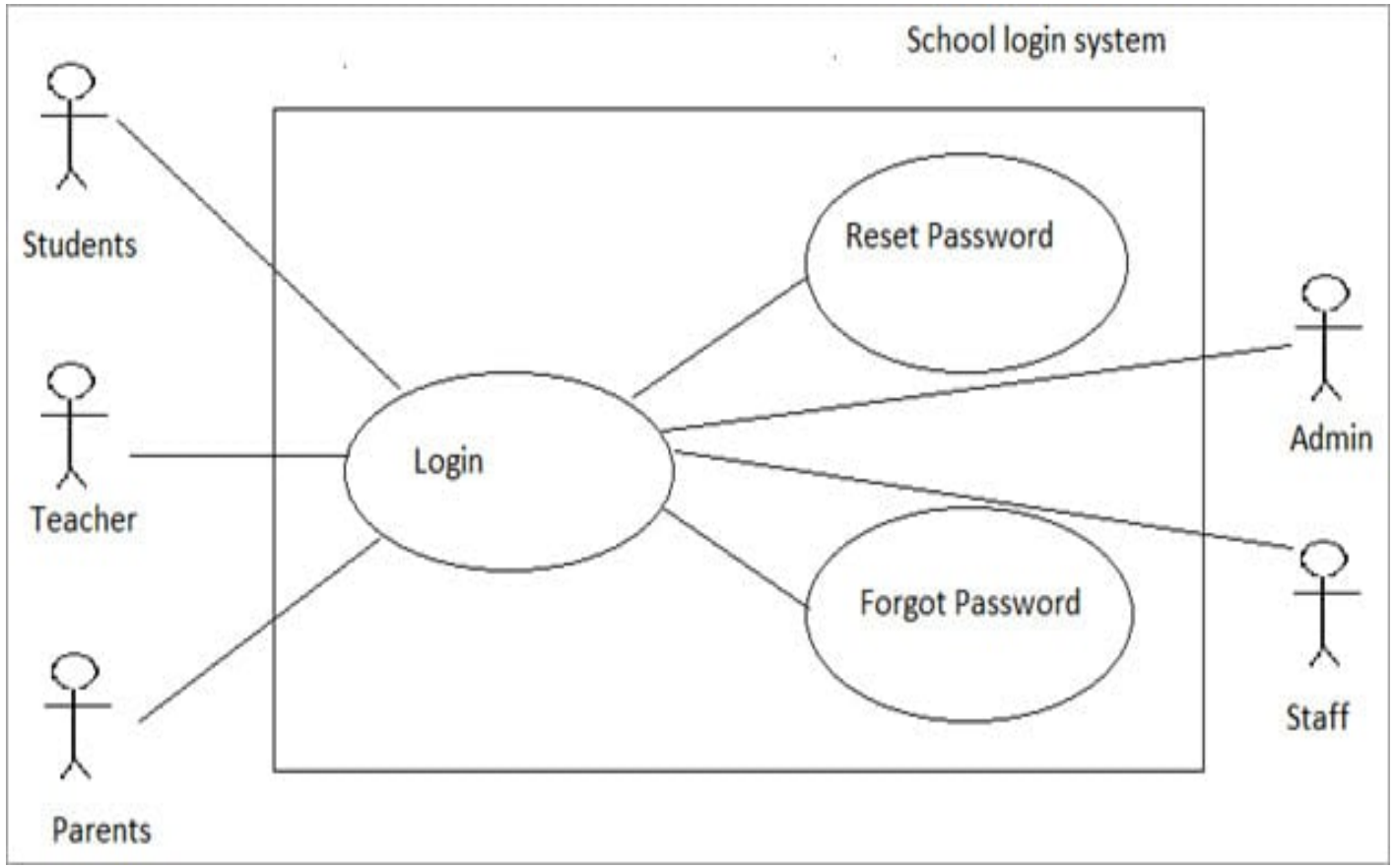
Use Case Example 1

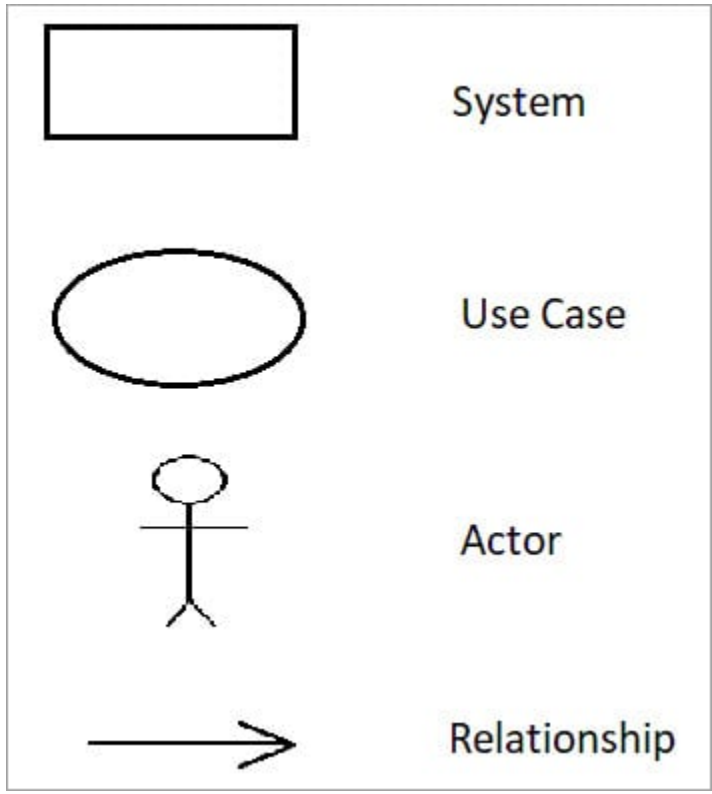
- Create a ‘Login’ to a ‘School Management System’

Use Case Name	Login
Use case Description	A user login to System to access the functionality of the system.
Actors	Parents, Students, Teacher, Admin
Pre-Condition	System must be connected to the network.
Post -Condition	After a successful login a notification mail is sent to the User mail id

Main Scenarios	Serial No	Steps
Actors/Users	1	Enter username Enter Password
	2	Validate Username and Password
	3	Allow access to System
Extensions	1a	Invalid Username System shows an error message
	2b	Invalid Password System shows an error message
	3c	Invalid Password for 4 times Application closed

Use case diagram of the example 1





Use case Testing example 2

- Consider The ATM PIN example. We show successful and unsuccessful scenarios. In this diagram we can see the interactions between the A (actor – in this case it is a human being) and S (system). From step 1 to step 5 that is success scenario it shows that the card and pin both got validated and allows Actor to access the account. But in extensions there can be three other cases that is 2a, 4a, 4b which is shown in the diagram below.
- For use case testing, we would have a test of the success scenario and one testing for each extension. In this example, we may give extension 4b a higher priority than 4a from a security point of view.

Example 2 contd.

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit

State Transition Testing

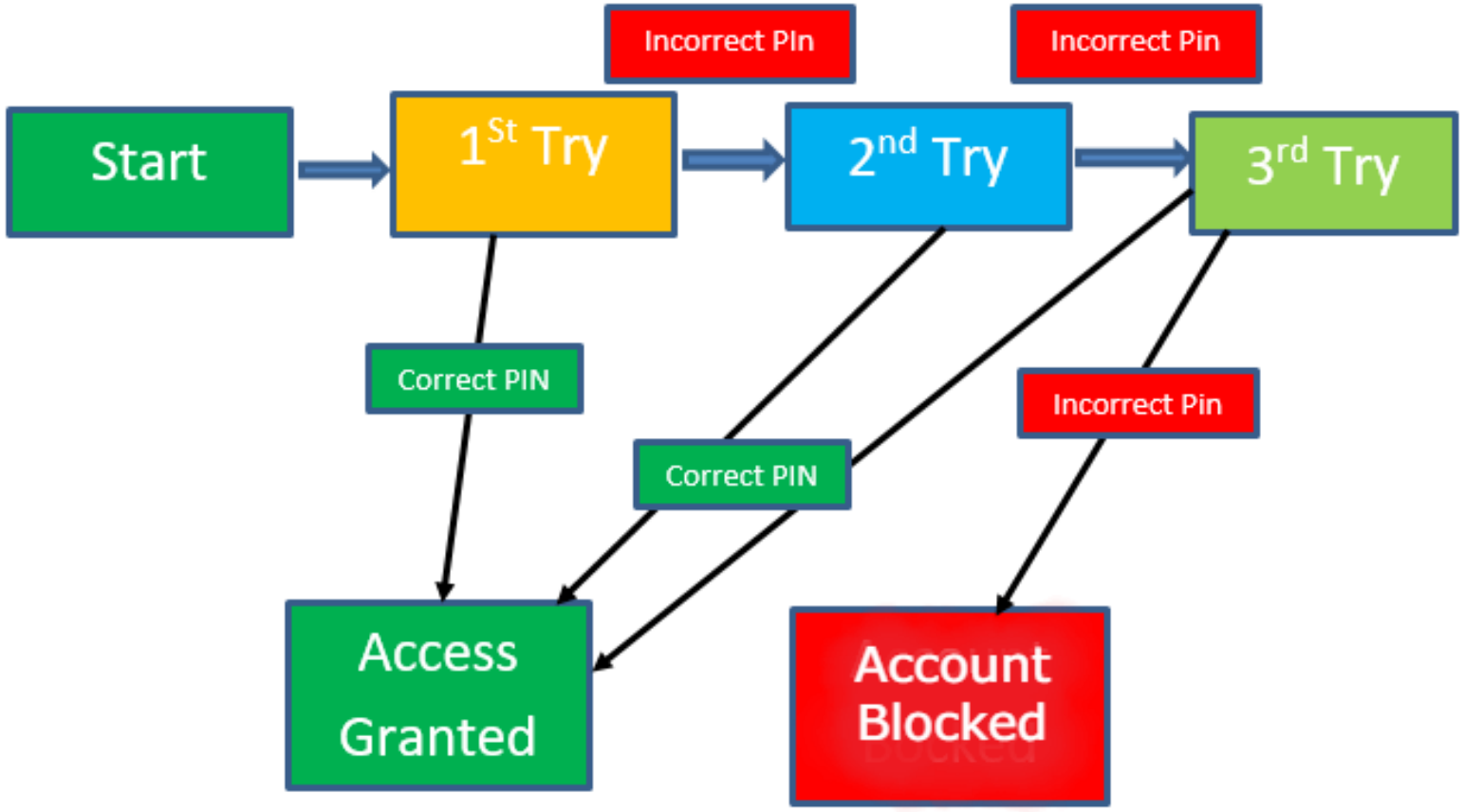
State Transition Testing is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test(AUT). State transition testing helps to analyze behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behavior.

Example

- Let’s consider an ATM system function where if the user enters the invalid password three times the account will be locked.
- In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

State transition diagram

In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.



State Transition Table

	Correct PIN	Incorrect PIN
S1) Start	**S5**	**S2**
S2) 1st attempt	**S5**	**S3**
S3) 2nd attempt	**S5**	**S4**
S4) 3rd attempt	**S5**	**S6**
S5) Access Granted	**–**	**–**
S6) Account blocked	**–**	**–**