

23.What is Decision Table

A **decision table** is a tabular method for representing and analyzing decision logic. It includes:

- **Condition Stub:** Lists the input conditions.
- **Condition Entries:** Shows true (YES), false (NO), or immaterial (I) for each condition under each rule.
- **Action Stub:** Lists the possible actions.
- **Action Entries:** Shows whether the action is taken (YES) or not (NO) for each rule.

Uses:

- Systematically represent complex decision logic.
- Serve as a basis for designing test cases.
- Help verify completeness and consistency.

Example

	RULE 1	RULE 2	RULE 3	RULE 4
CONDITION 1	YES	YES	NO	NO
CONDITION 2	YES	I	NO	I
CONDITION 3	NO	YES	NO	I
CONDITION 4	NO	YES	NO	YES
ACTION 1	YES	YES	NO	NO
ACTION 2	NO	NO	YES	NO
ACTION 3	NO	NO	NO	YES

Unit-4 -→ 8 marks

31. Can you explain the process of determining number of states in a state graph?

STATE GRAPH

The behaviour of the system is represented in graphical form which is known as "State Graph". The state graph is used for functional testing of the system to identify different bugs. The bugs include,

- (1) State Bugs.
- (2) Transition Bugs.

Elements : The state graphs include many elements such as,

- (1) States.
- (2) Inputs.
- (3) Transitions.
- (4) Outputs.

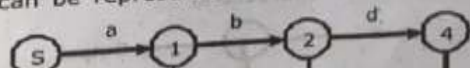
State, State Graphs and Transition Testing [Unit - VII]

- (1) **States :** A state in a state graph is the node which represents the characteristics and behaviour of the system. State graph has set of states and they are identified by characters or numbers.
- (2) **Inputs :** The states in the state graph contains input values to represent the changes to the states. They can be denoted by numbers or characters.
- (3) **Transitions :** The link that joins two states is known as transition. When some input is applied to the node, the state changes and the process of changing of state is known as transition.
- (4) **Outputs :** After processing an input, the result is produced which is known as output. The input and output of the state graph are separated by "slash"(/).

Process : The overall process of state graph could be the input variables are taken and they are processed from one state to another and the result of the process is produced.

Example : Consider the example of automatic teller machine (ATM) that consists of inputs, transitions and outputs.

- (1) In the first state, after inserting the card we enter pin number.
 - (2) The second state is that it verifies the pin number and if it is correct, it is processed to next step i.e., it asks for balance enquiry or money withdrawal. If it is incorrect, the state is repeated to enter the correct pin again.
 - (3) If we choose balance enquiry, the receipt is printed and if money withdrawal is chosen, it asks for money and then gives out money and receipt is printed.
 - (4) The process ends by taking the receipt and ATM card.
- The whole process can be represented in the form of State diagram as in Fig. 7.1.



The whole process can be represented as follows:

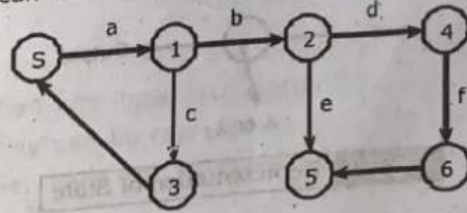


Fig. 7.1 Example of State Graph

The states and inputs and transitions in the state graph can be explained as,

- S → Start State
- a → Enter Pin Number
- 1 → Verification of pin number
- c → Pin number is wrong

- 3 → re-enter pin
- b → pin number is ok
- 2 → Screen displays two options balance enquiry, money withdrawal
- e → if "balance enquiry" is chosen
- 5 → Print receipt
- d → if "money withdrawal" is chosen
- 4 → enter amount
- f → amount is entered
- 6 → received money
receipt is printed.

32. How to identify equivalent states and how to merge them in representing state graph? Explain.

✓ What is a State Graph?

A **state graph** (or state transition diagram) represents:

- **States** of the system (nodes)
 - **Transitions** between states (edges) triggered by events or inputs.
-

🔗 What are Equivalent States?

Two states are said to be **equivalent** if:

- They respond to the **same inputs** in the **same way**, and
- Lead to **equivalent next states** under the same conditions.

☞ In other words, the behavior of the system is **identical** regardless of which state you are in.

🔍 How to Identify Equivalent States?

1. **Analyze Input-Output Behavior:**
 - Check how each state responds to every input.
 - Compare outputs for those inputs.
2. **Check Transition Paths:**
 - Compare the transitions (next states) from both states for the same input.
 - Ensure the next states are also equivalent.
3. **Use Partitioning Method (if needed):**
 - Initially group all states.
 - Iteratively split groups based on behavior differences until stable groups remain.

🔗 How to Merge Equivalent States?

Once two or more equivalent states are identified:

1. **Create a new single state** representing all equivalent ones.
2. **Redirect all incoming transitions** from the original states to the new merged state.
3. **Update outgoing transitions** so they originate from the merged state.
4. **Delete redundant original states.**

Example: ATM Pin Verification System

System Description:

- A user has 3 attempts to enter the correct PIN.
 - If the correct PIN is entered, go to **Access Granted** state.
 - After 3 incorrect attempts, go to **Card Blocked** state.
-

☐ States:

- **S0**: Start
 - **S1**: 1st incorrect attempt
 - **S2**: 2nd incorrect attempt
 - **S3**: 3rd incorrect attempt → Blocked
 - **S4**: Correct PIN → Access Granted
-

☐ Transitions:

- From **S0**, wrong PIN → **S1**
 - From **S1**, wrong PIN → **S2**
 - From **S2**, wrong PIN → **S3** (Card Blocked)
 - From **S0**, **S1**, or **S2**, correct PIN → **S4**
-

✓ Identifying Equivalent States:

Observe that:

- From **S0**, **S1**, and **S2**:
 - On correct PIN → **S4**
 - On incorrect PIN → next incorrect attempt state

The **output behavior is different**:

- From **S0**, an incorrect PIN leads to **S1**.
- From **S1**, an incorrect PIN leads to **S2**.
- From **S2**, it leads to **S3** (Blocked).

☞ These states are **not equivalent** because they lead to **different next states** on the same input.

🔄 But Now Consider:

Let's say you have two states:

- **S5** and **S6**, both representing "User is idle and authenticated"
- From both, on action "**Check Balance**" → **Balance State**
- On **Logout** → **S0**

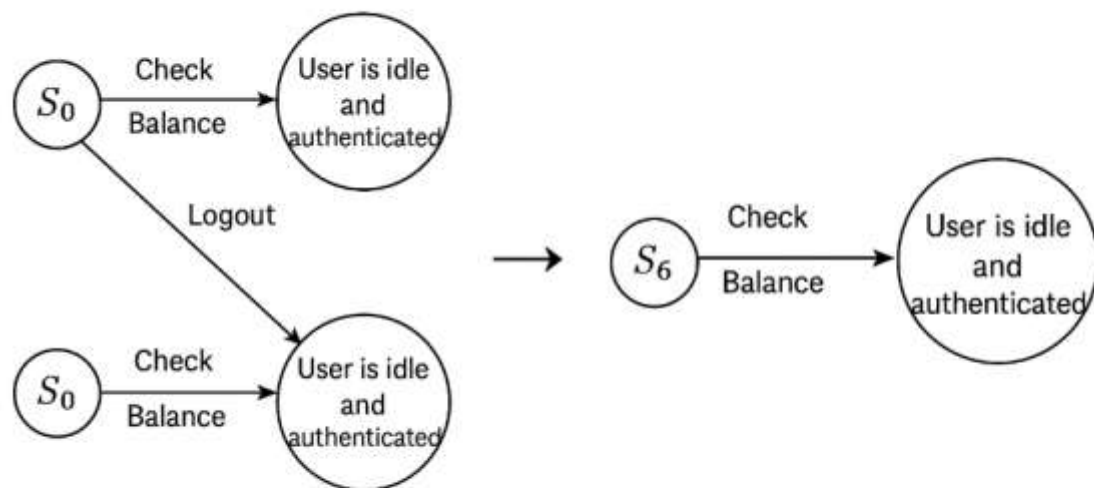
If all inputs in **S5** and **S6** result in the same outputs and transitions, then:

✓ **S5** \equiv **S6** (Equivalent)

🔄 Merging:

- Replace S5 and S6 with a new state **S5_6**
- Redirect all transitions to/from S5 and S6 to/from **S5_6**

Equivalent States: $S_5 = S_6$



Merging of Equivalent

33 Explain in detail about good and bad state graphs

Good State Graphs : The good and bad state graphs are identified or judged based on what kinds of graphs are used in test designing. There are some principles of test design in case of state graphs. They are,

- (1) The number of states in the state graph must be same as the product of possible number of factors that make up the state. i.e., the number of permutations of all values and all properties/attributes of the system.
- (2) For every unique combination of state and input, there should be only one transition specified.
- (3) For every transition, there should be atleast one sensible output.
- (4) Each and every state there should be a sequence of inputs that moves the system to the starting state.

Other than the above stated ones there are some other criterions to be considered for good graphs. They are,

- (1) The number of input codes for a state graph must be atleast two because only few kinds of state graphs can be constructed with one input code.
- (2) A good state graph has at least two input symbols. With one symbol only a limited number of useful graphs are possible.

Bad State Graphs : The state graphs which cannot be good and which do not follow any principles of good state graphs are known as bad state graphs.

The state graphs which are not reachable and which are buggy are bad state graphs.

Examples : There are some cases to show bad state graphs. They are, shown in Fig. 7.7.

(1)

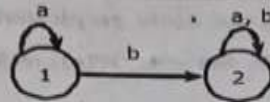


Fig. 7.7 Example 1

In the above figure, the state 2 once reached would never be left. As a result the starting state cannot be achieved. It violates the principle of good state graph.

(2)

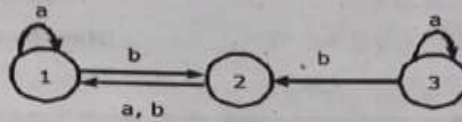


Fig. 7.8 Example 2

The state '3' cannot be entered at all as there is no path from 2 to 3. This leads to unreachable paths. Hence bad or improper graph.

(3)

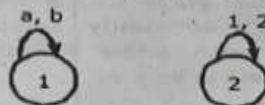


Fig. 7.9 Example 3

Here both the states 1 and 2 are not reachable as there is no path between them.

them.

(4)

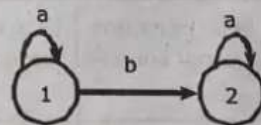


Fig. 7.10 Example 4

There is no transition for input 'b' at state '2'. If input 'b' is received at state '2' the path is undefined.

(5)

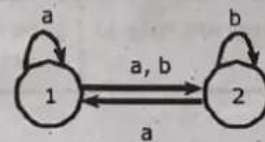


Fig. 7.11 Example 5

There are two different paths for the same input on same state. The system will be in confusion to consider the path when input 'a' is given at state 1.

The above discussed are some of the improper or bad state graphs.

Table 7.2 Comparison between Good and Bad State Graphs

S.No.	Good State Graph	Bad State Graph
(1)	If the elements of the state graph i.e., inputs, transitions, outputs are clearly specified and they are easily understandable, then they are said to be good state graphs.	If the elements of the state graph are not specified clearly then they are said to be bad state graphs.
(2)	In a good state graph, every state should have a sequence of inputs that drives the system to the starting state.	In a bad state graph, the sequence of inputs does not lead to the initial state.
(3)	There should be only one transition specified for one state and input combination.	More than one transitions are specified for every state and input combination.
(4)	There would be only one output for every transition because unique output gives clarity otherwise it leads to confusion.	There might be none or more than one output for every transition which results in improper state graph.
(5)	Here bugs are less and they are easy to find.	Here bugs are more and they are very difficult to find.

Unit-5

40. Explain Graph Matrices and applications

The video player shows a slide titled "GRAPH MATRICES AND APPLICATIONS" with a subtitle "MOTIVATIONAL OVERVIEW". The slide lists several points:

- The pictorial graphs are represented using another notation is know as Graph Matrices.
- The concept of graph matrices have been evolved inorder to solve the problems of pictorial graphs.
- Graph matrices are used for proving the theorems.
- The graph theory contains basic tool kit of three algorithms.
- ❖ Matrix Multiplication
- ❖ Partitioning Algorithm
- ❖ Matrix Determinant

The video player interface includes a search bar with the text "graph matrices and applications in stm", a play button, a progress bar at 1:41 / 13:09, and a video thumbnail of a woman. The channel name "CSE SeekerRs" with 2.72K subscribers is visible, along with a "Subscribe" button and a "Share" button.

The video player shows a slide titled "APPLICATIONS OF GRAPH MATRICES". The slide lists several applications:

- Matrices are used when the programmer codes or Encrypts a message.
- Geologists make use of matrices for seismic surveys.
- Matrices are also used in Computer Animation.
- Matrices are used with computing in Architecture.
- Math reports are recorded using matrices in Engineering
- Matrices are used to calculate gross domestic product in econ
- goods can be produced more efficiently.

The video player interface includes a search bar with the text "graph matrices and applications in stm", a play button, a progress bar at 3:01 / 13:09, and a video thumbnail of a woman. The channel name "CSE SeekerRs" with 2.72K subscribers is visible, along with a "Subscribe" button and a "Share" button.


18:17 49%


graph matrices and applications in stm


MATRIX OF A GRAPH


- The another notation for representing graphs is known as "Graph Matrix"
- Each and every node in the graph is represented as a single row and a single column.

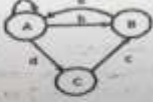
Examples : Some examples of graph and their matrices are given below.

(a)  $\begin{bmatrix} 0 \end{bmatrix}$

(b)  $\begin{bmatrix} 1 \end{bmatrix}$

(c)  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

(d)  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

(e)  $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

5:03 / 13:04 - Graph Matrix

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:17 49%

graph matrices and applications in stm

SIMPLE WEIGHT AND CONNECTION MATRIX

➤ The simplest weight to represent whether the connection exists or does not exist between two nodes is the binary weights ie., it is represented as "1" if link exists otherwise it is represented as "0".

➤ The Arithmetic rules for binary weights are,

$1+1=1$	$1 \times 1=1$
$1+0=1$	$1 \times 0=1$
$0+0=0$	$0 \times 0=0$

5:48 / 13:09 - Graph Matrix

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:18 49%

graph matrices and applications in stm

PRINCIPLES

- The size of the matrix i.e., number of rows x number of columns should be equal to the number of nodes.
- The direct link or connection between any node to any other node can be put in the matrix.
- The entry at the intersection of the row and column is the weight of the link, that will connect two nodes in represented direction.
- If there are more than one link between two nodes which are known as parallel links, then the entry of those links in the matrix would be sum of all the link w

6:48 / 13:09 - Graph Matrix

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:18 48%

graph matrices and applications in stm

If the matrix is represented by the weights which are defined as above are known as Connected Matrix, i.e., representing a matrix with 0's and 1's is known as Connected Matrix.

Example : The example of a connected matrix can be,

Fig. 8.2 Example Graph

7:03 / 13:09 - Flow Graph

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:18 48%

graph matrices and applications in stm

The general matrix i.e., relational matrix and connection matrix of the Fig. 8.2 can be represented as,

	A	B	C	D	E
A			a		
B					
C		d		b	
D		c			f
E		g	e		h

	A	B	C	D	E
A	0	0	1	0	0
B	0	0	0	0	0
C	0	1	0	1	0
D	0	1	0	0	1
E	0	1	1	0	1

(a) Relation Matrix (b) Connection Matrix

Fig. 8.3 Matrix Representations

7:53 / 13:09 - Flow Graph >

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers [Subscribe](#) 346 [Share](#)

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:18 48%

graph matrices and applications in stm

- The links of the figure are represented as "1" and if there is no link, it is represented as "0".
- The entries in each row would represent the outlinks from the corresponding node and entries in each column represents the inlinks from the corresponding node.
- The different nodes such as Branch node, Junction Node, loop node in connection graph can be identified as,

8:18 / 13:09 - Flow Graph >

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers [Subscribe](#) 346 [Share](#)

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:19 48%

graph matrices and applications in stm

Junction Node

- If the column contains two or more 1's then it is a Junction node.
- The columns B,C,E in the above figure are known as "Junction nodes".

Loop Node

- If there is an entry "1" in the diagonal of corresponding node then that node is a loop node.
- The node E in Figure is known as "Loop Node"

9:45 / 13:05 • Flow Graph >

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

Subscribe 346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

18:19 48%

graph matrices and applications in stm

Branch Node

- If the row contains two or more 1's then it is a branch node. The rows C,D,E in the above figure are branch node.
- The number of decisions at that particular node can be calculated by using $n-1$ formula.

So for node

$$C \rightarrow 2 - 1 = 1$$

$$D \rightarrow 2 - 1 = 1$$

$$E \rightarrow 2 - 1 = 1$$

9:13 / 13:05 • Flow Graph >

SOFTWARE TESTING || UNIT - 5 || GRAPH MATRICES AND ITS APPLICATIONS || CLASS - 4 ||

CSE SeekerS 2.72K subscribers

Subscribe 346 Share

ST || UNIT - 5 || POWER OF MATRIX, PARTITION...

42. Write the usage and applications of Win runner tools in software testing

✔ Usage and Applications of WinRunner in Software Testing

WinRunner is an automated functional GUI testing tool developed by **Mercury Interactive** (now part of Micro Focus). It is widely used to perform **regression, functional, and data-driven testing** of software applications.

🔧 Usage of WinRunner in Software Testing

1. **Automated Functional Testing**
 - Automates GUI interactions like mouse clicks, keyboard inputs, form submissions, etc.
 - Helps in testing how the application behaves under various input conditions.
 2. **Regression Testing**
 - Re-runs recorded test cases to ensure that recent code changes haven't broken existing functionality.
 3. **Data-Driven Testing**
 - Allows separation of test logic and test data.
 - Test scripts can run multiple times with different data sets (from Excel, databases, etc.).
 4. **Test Script Language (TSL)**
 - Uses a scripting language called TSL (Test Script Language) for customizing test cases, inserting checkpoints, loops, and conditions.
 5. **GUI Mapping and Testing**
 - Identifies and stores all GUI objects in an application in a **GUI map**.
 - Scripts use this map to interact with UI components during tests.
 6. **Synchronization**
 - Ensures test scripts wait for the application to be ready (like loading a page or a button becoming enabled).
 7. **Error Handling**
 - Supports recovery scenarios to handle unexpected errors or popups during test execution.
-

★ Applications of WinRunner

1. **Banking and Financial Systems**
 - Automates repetitive test cases for transactional applications to ensure consistency and accuracy.
2. **ERP and CRM Testing**
 - Tests SAP, Oracle, and PeopleSoft applications for UI consistency and functional correctness.
3. **E-commerce Applications**

- Validates workflows such as user login, cart operations, and payment processes.
- 4. **Desktop and Web-based Applications**
 - Used in testing both standalone applications and browser-based interfaces.
- 5. **Cross-version Compatibility Testing**
 - Ensures that older features continue working with newer releases (especially important in legacy systems).
- 6. **Product-based Companies**
 - Used to regularly test software builds to maintain stability and reduce manual testing overhead.

⊗ **Note:**

- **WinRunner is now obsolete** and has been officially replaced by **HP UFT (Unified Functional Testing)**, previously known as **QuickTest Professional (QTP)**.
- However, its concepts still form the foundation of modern GUI automation tools.