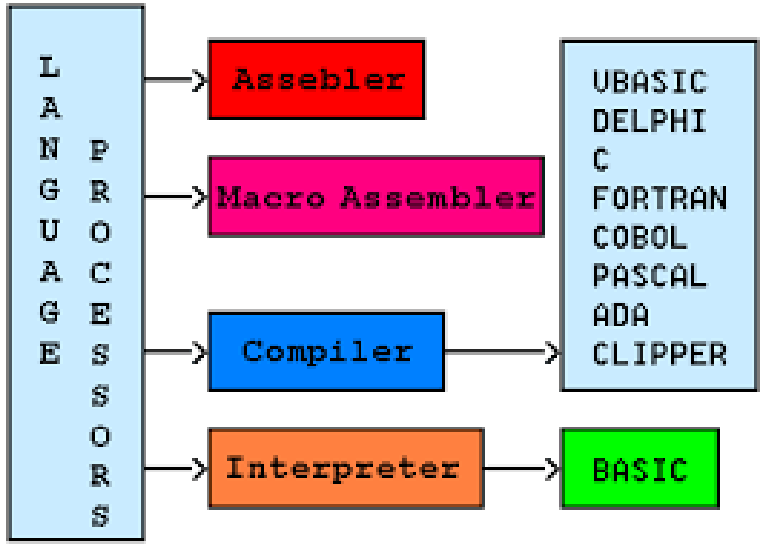
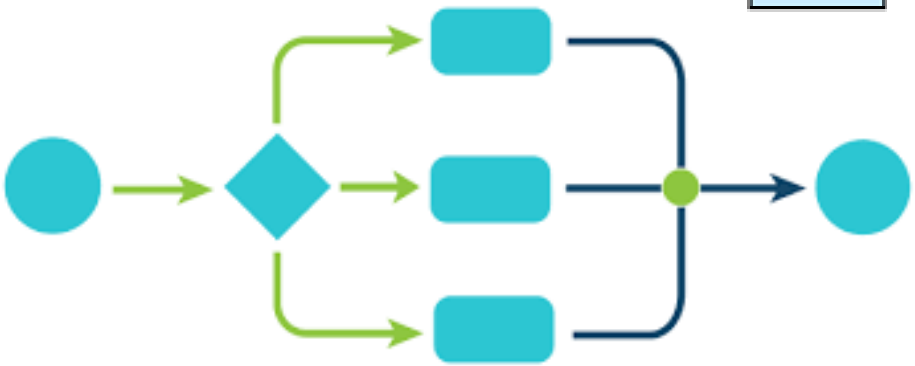
State Based Testing

Where it is useful?

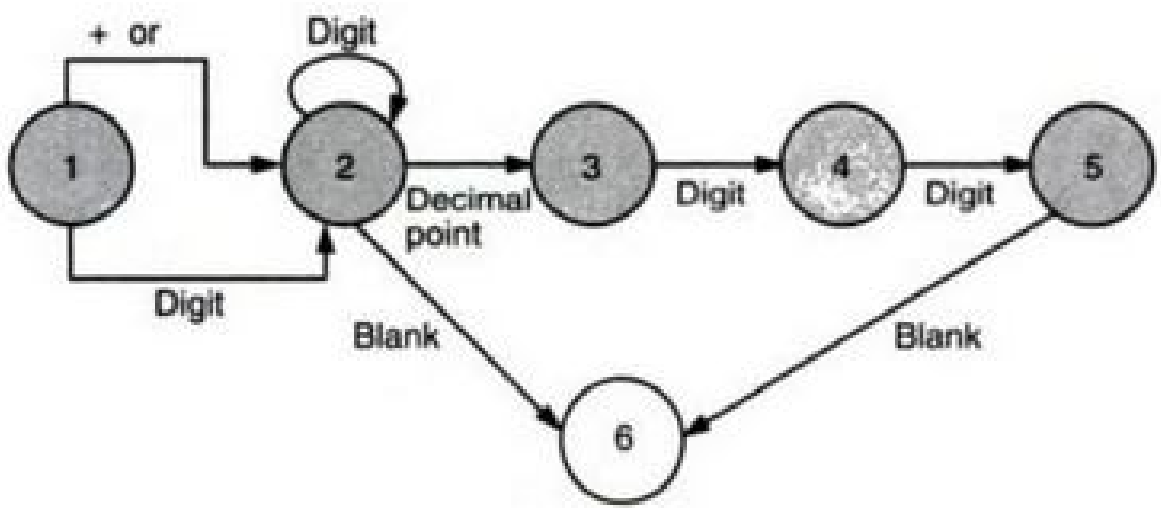
- Useful at situations
  - Language Processor
  - Workflow Modeling
  - Dataflow modeling





Example

- Validates a number according to the rules
  - Starts with optional sign
  - optional sign followed by any number of digits
  - Digits can be optionally followed by a decimal point, represented by a period
  - Two digits after decimal point
  - Whether or not it has decimal, terminated by blank (applies for both decimal or not a decimal)



- Start from State 1
- State 1→ state 2, it may be +/-/digit
- Any alphabet at state 2 →error
- Continue until it reaches the final state state 6

State Transition Table

Current state	Input	Next state
1	Digit	2
1	+	2
1	–	2
2	Digit	2
2	Blank	6
2	Decimal point	3
3	Digit	4
4	Digit	5
5	Blank	6

Language Processor

- Identify the grammar scenario. Represent as state diagram
- Design test cases to each Valid state input combinations
- Design test cases to most common Invalid combinations of state inputs

Leave Application by an employee

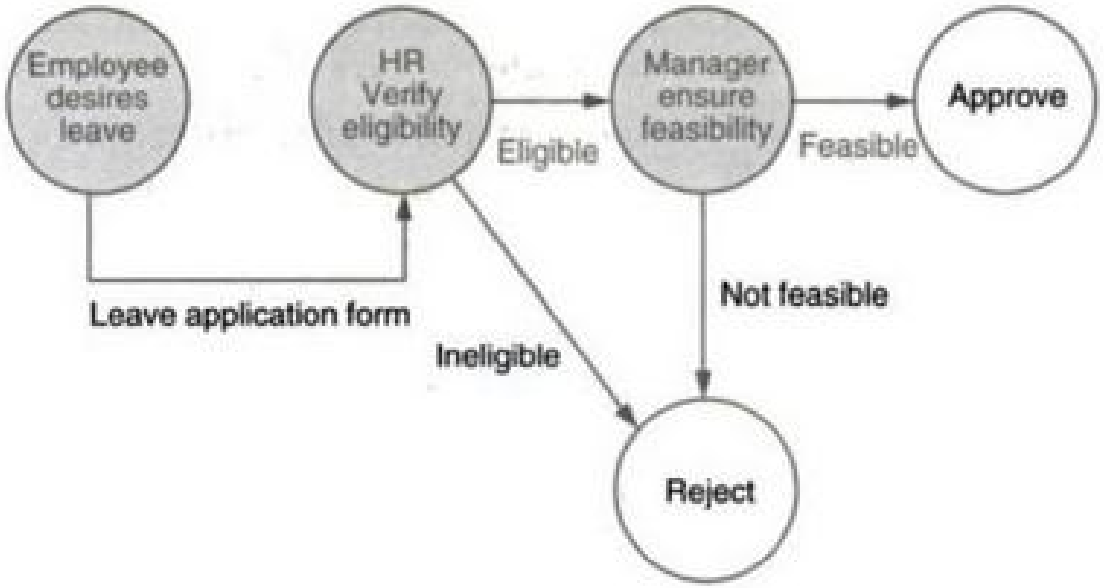
The employee fills up a leave application, giving his or her employee ID, and start date and end date of leave required.

This information then goes to an automated system which validates that the employee is eligible for the requisite number of days of leave. If not, the application is rejected; if the eligibility exists, then the control flow passes on to the next step below.

This information goes to the employee's manager who validates that it is okay for the employee to go on leave during that time

Having satisfied himself/herself with the feasibility of leave, the manager gives the final approval or rejection of the leave application.

Example


Employee desires leave
HR Verify eligibility
Manager ensure feasibility
Approve
Eligible
Feasible
Leave application form
Ineligible
Not feasible
Reject

Cause – Effect Graphing

Cause and Effect Graphing

- Used to combine conditions and derive an effective set of test cases that may disclose inconsistencies in a specification.
- The specification must be transformed into a graph.
- The graph itself must be expressed in a graphical language.
- Developing the graph, especially for a complex module with many combinations of inputs, is difficult and time consuming.
- The graph must be converted to a decision table that the tester uses to develop test cases.

Steps in developing test cases with a cause-and-effect graph

- The tester must decompose the specification of a complex software component into lower-level units.
- For each specification unit, the tester needs to identify causes and their effects.
- A cause is distinct input condition or an equivalence class of input conditions. An effect is an output condition or a system transformation. Putting together a table of causes and effects helps the tester to record the necessary details. The logical relationships between the causes and effects should be determined. It is useful to express these in the form of a set of rules.

- From the cause-and-effect information, a Boolean cause-and-effect graph is created. Nodes in the graph are causes and effects. Causes are placed on the left side of the graph and effects on the right. Logical relationships are expressed using standard logical operators such as AND, OR, and NOT, and are associated with arcs.
- The graph may be annotated with constraints that describe combinations of causes and/or effects that are not possible due to environmental or syntactic constraints.
- The graph is then converted to a decision table.
- The columns in the decision table are transformed into test cases

Example

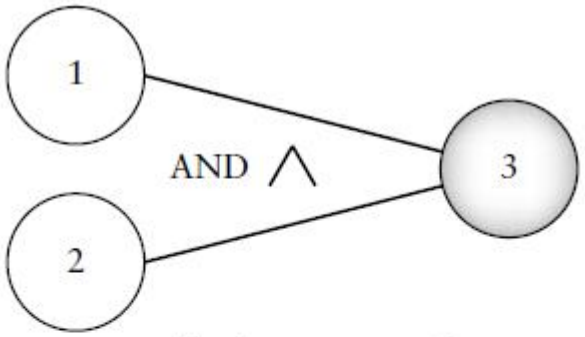
- Suppose we have a specification for a module that allows a user to perform a search for a character in an existing string. The specification states that the user must input the length of the string and the character to search for. If the string length is out-of-range an error message will appear. If the character appears in the string, its position will be reported. If the character is not in the string the message “not found” will be output.

Example: Cause and effect

- The input conditions, or causes are as follows:
  - C1: Positive integer from 1 to 80
  - C2: Character to search for is in string
- The output conditions, or effects are:
  - E1: Integer out of range
  - E2: Position of character in string
  - E3: Character not found

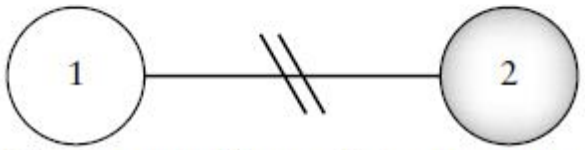
Example: Logical Relationships

- The rules or relationships can be described as follows:
  - If C1 and C2, then E2.
  - If C1 and not C2, then E3.
  - If not C1, then E1.



Effect 3 occurs if both causes 1 and 2 are present.

Effect 2 occurs if cause 1 occurs.



Effect 2 occurs if cause 1 does not occur.

Example: Decision Table

- A decision table will have a row for each cause and each effect.
- The entries are a reflection of the rules and the entities in the cause and effect graph.
  - “1” for a cause or effect that is present
  - “0” represents the absence of a cause or effect
  - “—” indicates a “don’t care” value.

Example: abcde

Inputs	Length	Character to search for	Outputs
T1	5	c	3
T2	5	w	Not found
T3	90		Integer out of range

	T1	T2	T3
C1	1	1	0
C2	1	0	—
E1	0	0	1
E2	1	0	0
E3	0	1	0

Advantages

- Development of the rules and the graph from the specification allows a thorough inspection of the specification. Any omissions, inaccuracies, or inconsistencies are likely to be detected.
- Exercising combinations of test data that may not be considered using other black box testing techniques.

Problem

- Developing a graph and decision table when there are many causes and effects to consider.

Thank You