

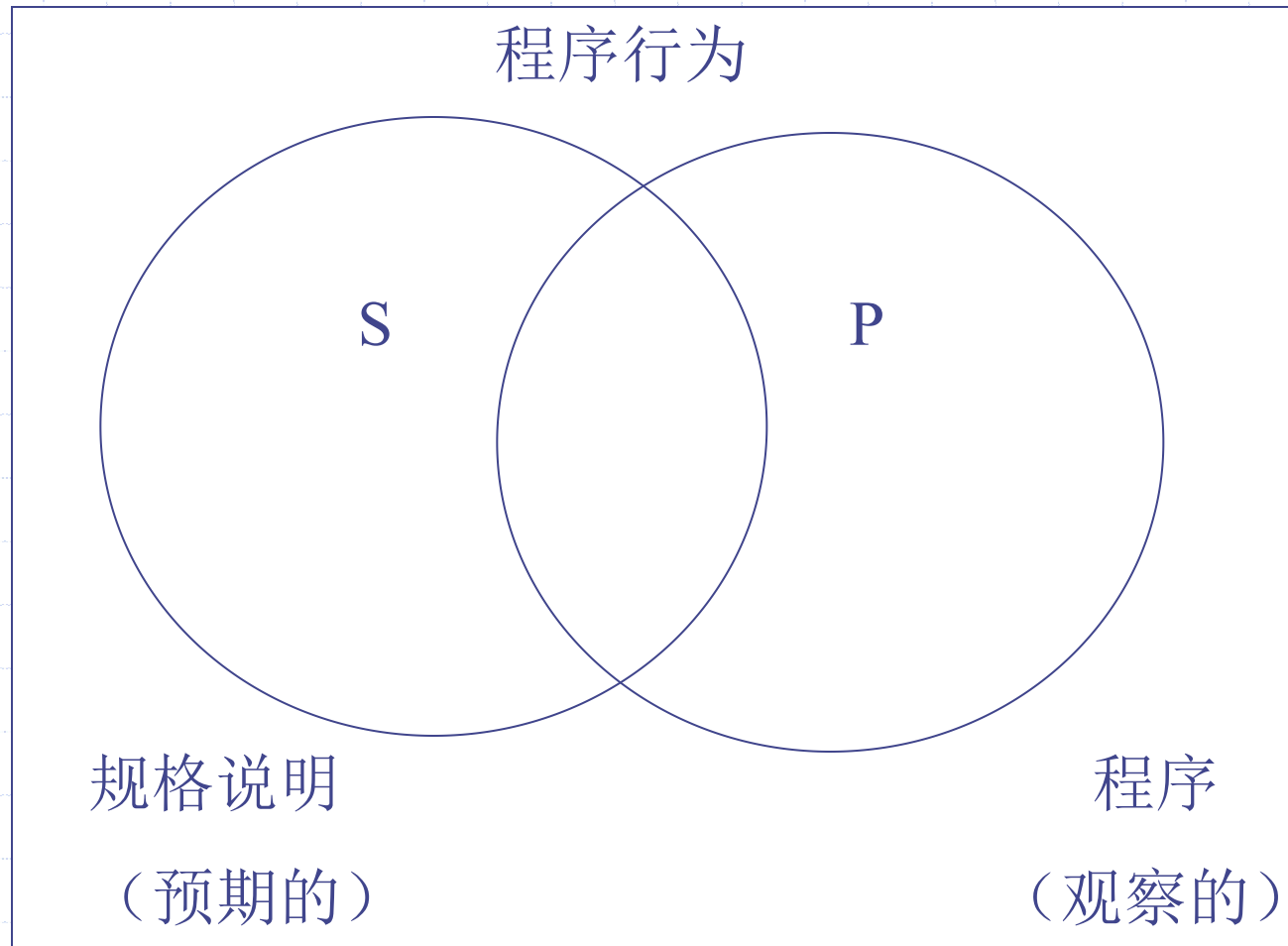


Software Testing Fundamental

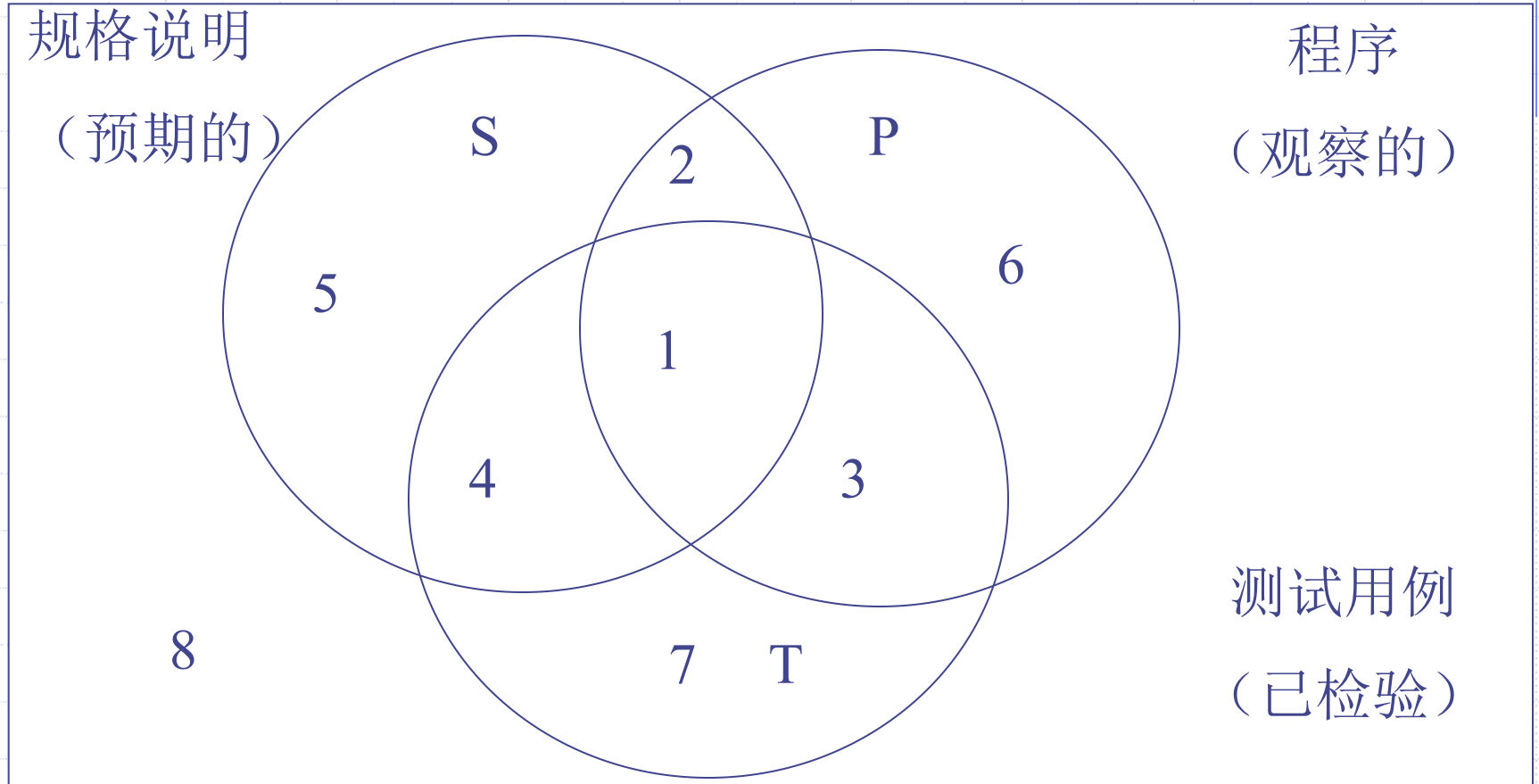
Outline

- ◆ Examples Used In Testing Discussions
- ◆ Discrete Mathematics for Testers
- ◆ Graph Theory for Testers

Specified and Programmed Behaviors



Specified, Programmed, and Tested Behaviors



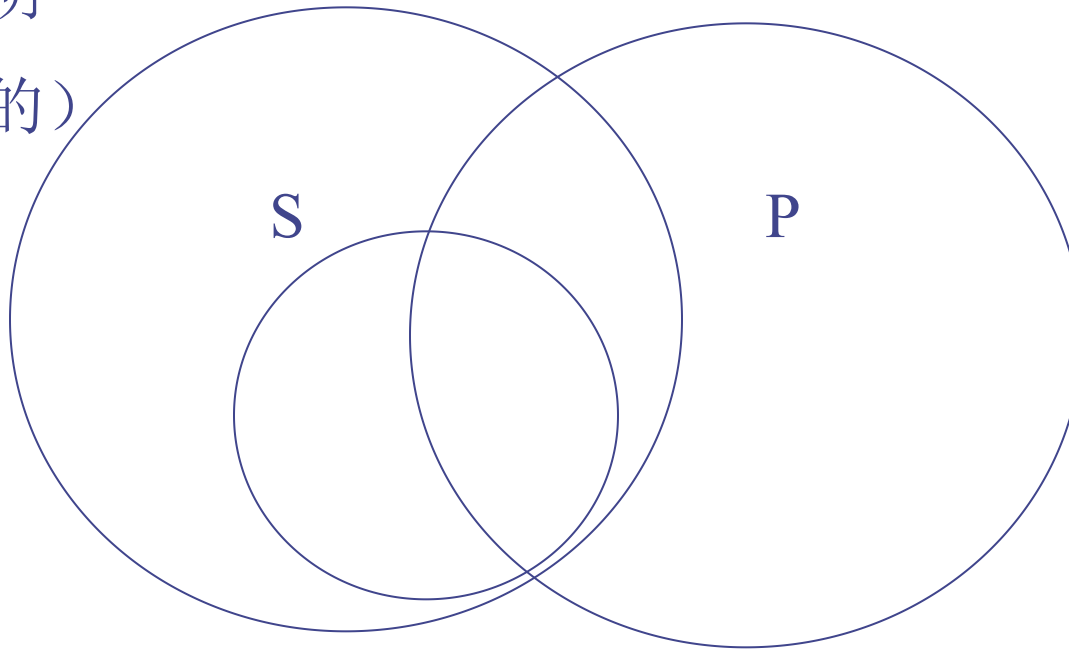
Test Case Design Strategies

- Functional Testing or Black-box testing (knowing the specified function a product is to perform and demonstrating correct operation based solely on its specification without regard for its internal logic)
- Structural Testing or White-box testing (knowing the internal workings of a product, tests are performed to check the workings of all independent logic paths)

Functional Testing (Black Box Testing, BBT)

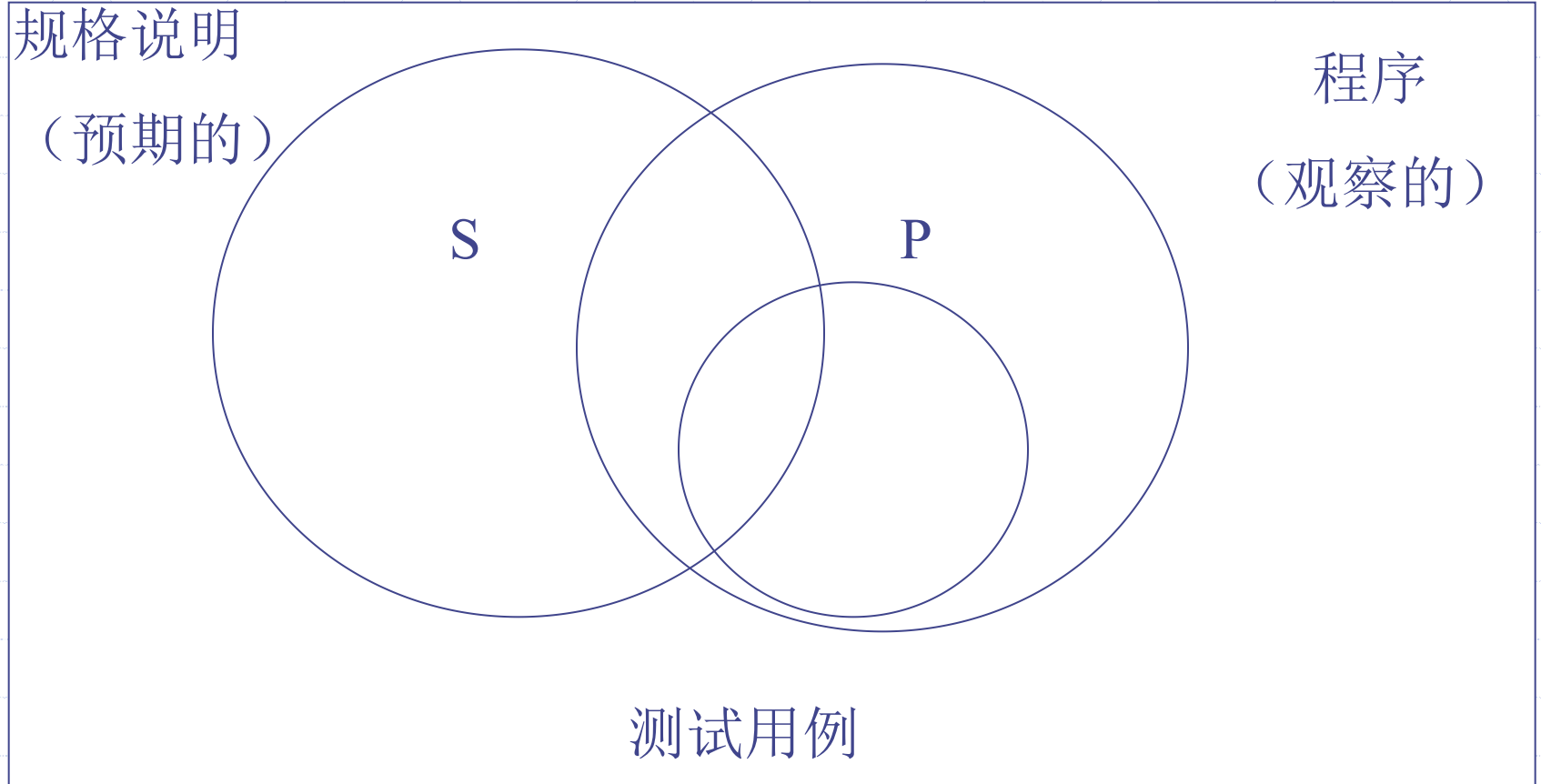
规格说明
(预期的)

程序
(观察的)



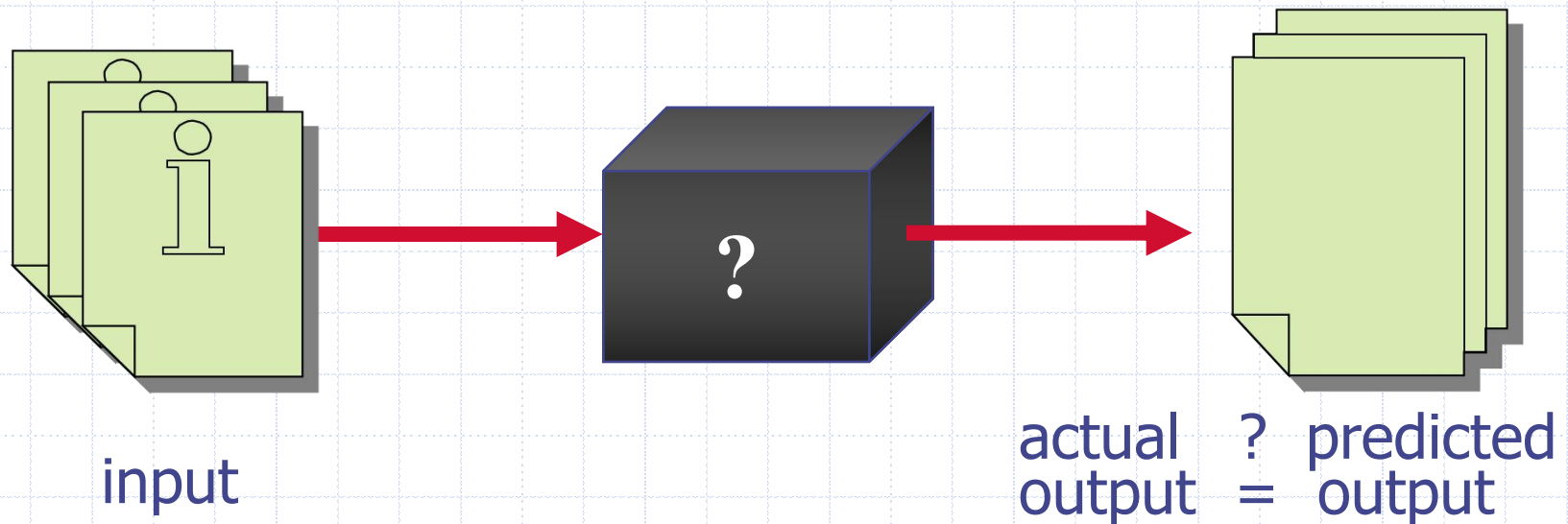
测试用例

Structural Testing (White Box Testing, WBT)

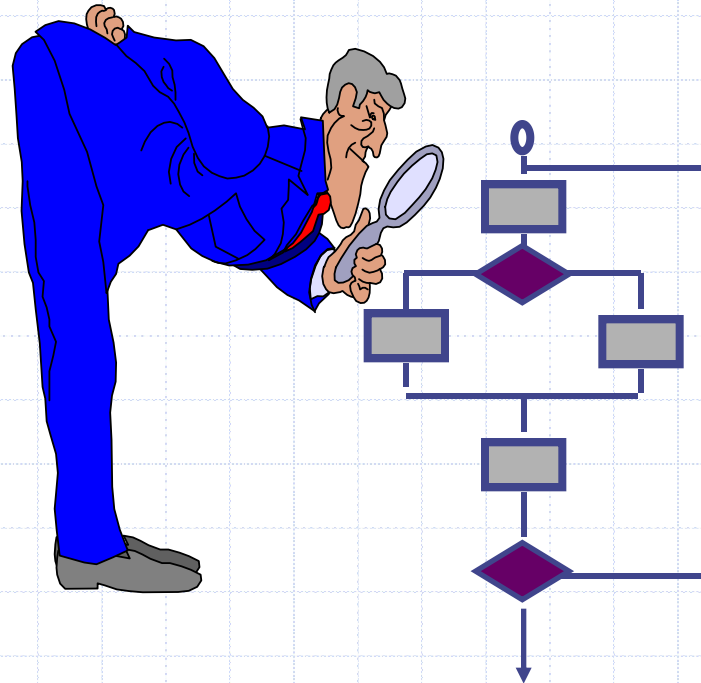


Functional Testing

- ◆ Derive external conditions that fully exercise all functional requirements



Structural Testing



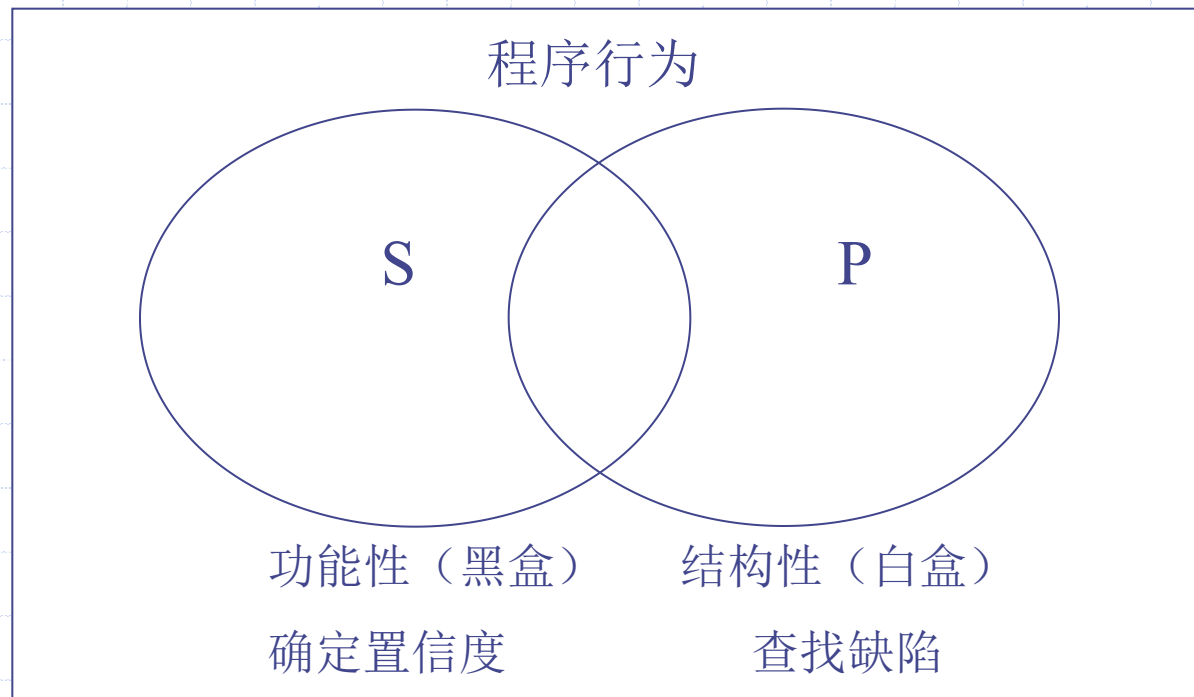
... our goal is to ensure that all statements and conditions have been executed at least once ...

Functional vs. Structural

- ◆ planned without the intimate knowledge of the program design or its implementation.
- ◆ based on the specification of the program interfaces (I.e. procedure and function headers).
- ◆ need to specify the program inputs and the expected program outputs.
- ◆ planned with the intimate knowledge of the program detailed design (e.g. PDL) or its implementation
- ◆ Test each aspect of the program logic
- ◆ The required test inputs and expected outputs need to be constructed in such a way as to satisfy the expected program coverage

Functional vs. Structural

- ❖ If all specified behaviors have not been implemented, structural test cases will never be able to recognize this.
- ❖ Conversely, if the program implements behaviors that have not been specified, this will never be revealed by functional test cases.



Questions

- ◆ What should be the objective of testing?
- ◆ Describe the differences between black-box testing and white-box testing.
- ◆ True or false?
 - Functional test is better than structural test

Outline

- ◆ Examples Used In Testing Discussions
- ◆ Discrete Mathematics for Testers
- ◆ Graph Theory for Testers

Examples Used In This Course

- ◆ Examples used in unit testing methods
 - The Triangle Problem
 - The NextDate Function
 - The Commission Problem
- ◆ Examples used in integration and system testing methods
 - The SATM System
- ◆ Examples used in object-oriented testing methods
 - The NextDate Function
 - The Currency Converter
 - Saturn Windshield Wiper Controller

Example 1: The Triangle Problem

◆ Problem Statement

- Input: 3 integers (sides of a triangle)
- Output: Type of Triangle (Equilateral, Isosceles, Scalene or NotATriangle)

The Triangle Problem

- ◆ The Triangle Property: The sum of any pair of sides must be strictly greater than the third side
 - a, b, c : integer sides
 - If $((a < b + c \ \&\& \ b < a + c \ \&\& \ c < a + b) == \text{true})$ then Triangle else NotATriangle
- ◆ Equilateral triangle: $a = b = c$
- ◆ Isosceles triangle: $a = b \neq c \ || \ a = c \neq b \ || \ b = c \neq a$
- ◆ Scalene triangle: $a \neq b \ \&\& \ a \neq c \ \&\& \ b \neq c$
- ◆ Complexity is due to relationships between inputs and correct outputs

Example 2: The NextDate Function

◆ Problem Statement

- Input: 3 integers (month, day, year)
- Output: The date of the day after the input date
- Input Constraints: $1 \leq \text{month} \leq 12$, $1 \leq \text{day} \leq 31$ and $1812 \leq \text{year} \leq 2012$

The NextDate Function

- ◆ Complexity is due to logical relationships among the input variables
 - Complexity of the input domain
 - The rule that distinguishes common years from leap years
- ◆ Gregorian Calendar rule:
 - A year is a leap year if is divisible by 4, unless it is a century year
 - Century years are leap years only if they are multiples of 400
- ◆ NextDate function is an example of Zipf's Law
 - 80% of the activity occurs in 20% of the space

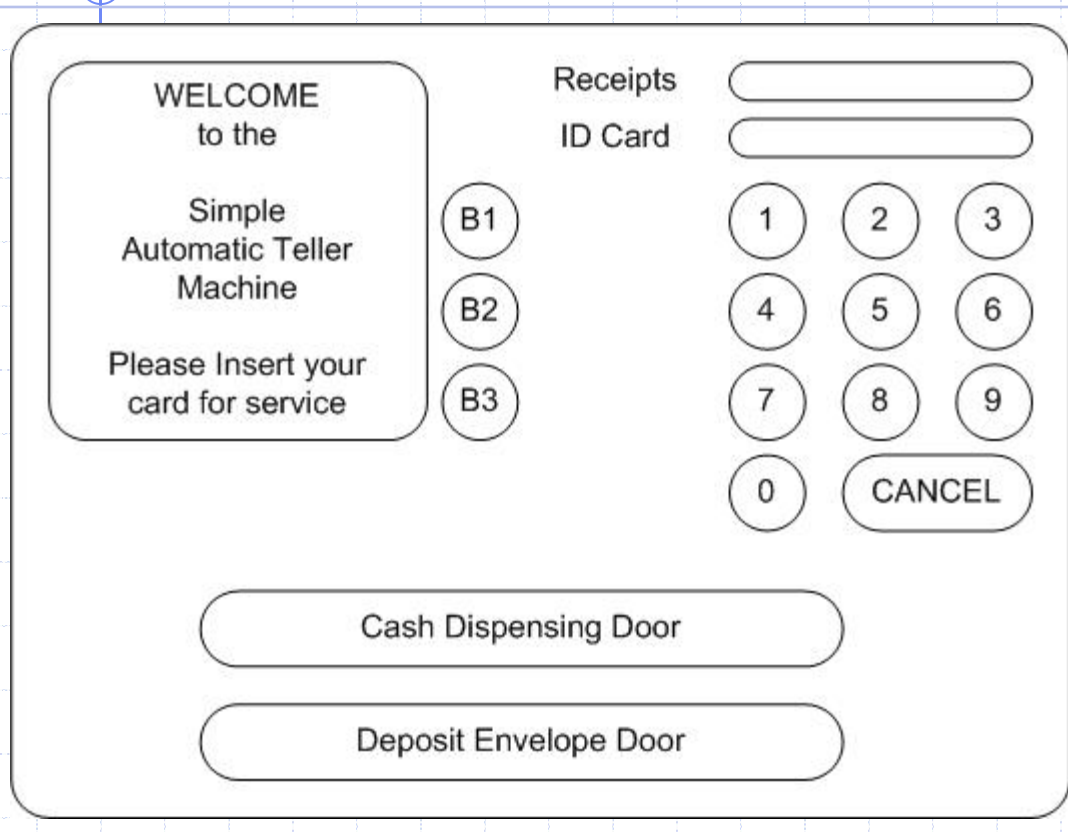
Example 3: The Commission Problem

- ◆ Rifle salespersons in the Arizona Territory sold rifle locks, stocks, and barrels made by a gunsmith in Missouri
- ◆ Lock = \$45.00, stock = \$30.00, barrel = \$25.00
- ◆ Each salesperson had to sell at least one complete rifle per month
- ◆ The most one salesperson could sell in a month was 70 locks, 80 stocks, and 90 barrels
- ◆ Each salesperson sent a telegram to the Missouri company with the total order for each town (s)he visits
- ◆ $1 \leq \text{towns visited} \leq 10$, per month
- ◆ Commission: 10% on sales up to \$1000, 15% on the next \$800, and 20% on any sales in excess of \$1800

The Commission Problem

- ◆ The company had 4 salespersons
- ◆ Monthly datafile: name followed by one line for each telegram order (locks, stocks, barrels)
- ◆ The datafile ended with -1 in the locks position
- ◆ Monthly sales report: name, total number of locks, stocks, and barrels sold, the total dollar sales, the commission
- ◆ It is a more typical example of commercial computing

Example 4: The SATM System



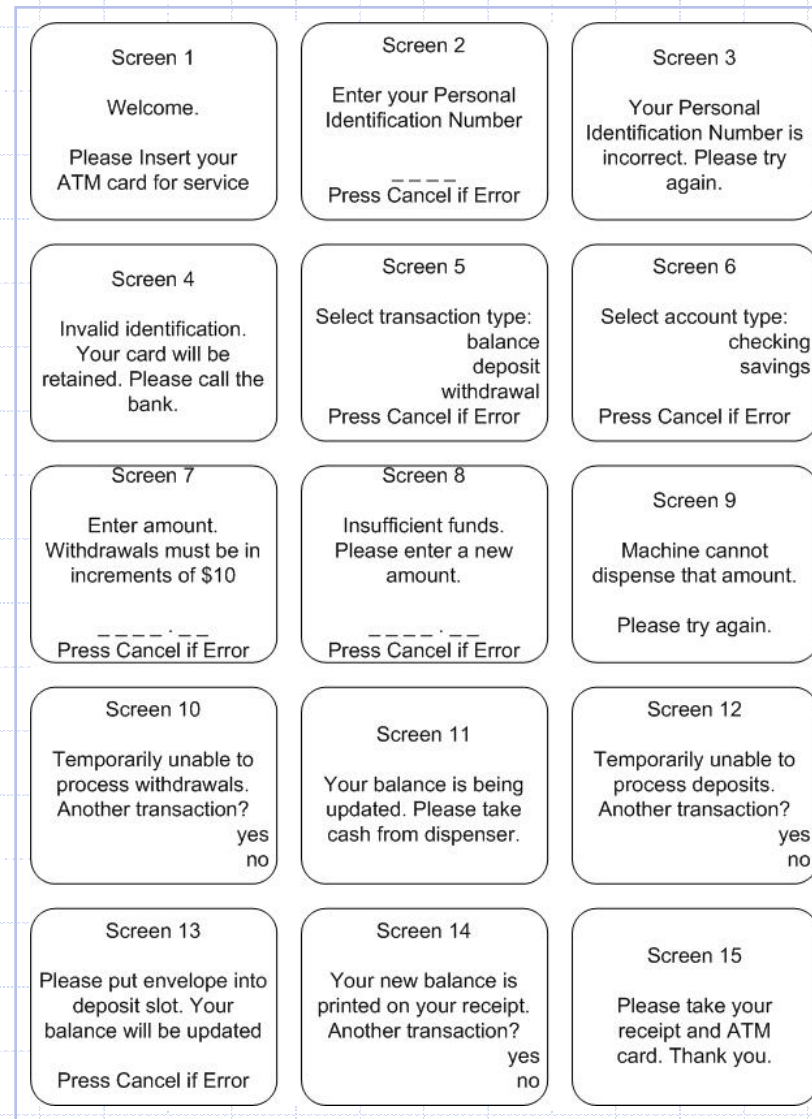
The diagram illustrates the SATM Terminal interface. It features a central display area with the following elements:

- Welcome Message:** "WELCOME to the Simple Automatic Teller Machine".
- Instruction:** "Please Insert your card for service".
- Card Slots:** Two slots labeled "Receipts" and "ID Card".
- Buttons:** A numeric keypad (0-9) and a "CANCEL" button. To the left of the keypad are three circular buttons labeled "B1", "B2", and "B3".
- Dispensing/Deposit Doors:** Two large buttons at the bottom labeled "Cash Dispensing Door" and "Deposit Envelope Door".

The SATM Terminal

- ◆ Transaction Types: deposits, withdrawals, balance inquiries
- ◆ Account Types: checking, savings

The SATM System



The SATM System

- ◆ When a bank customer arrives at an SATM station, screen 1 is displayed. The bank customer accesses the SATM system with a plastic card encoded with a Personal Account Number (PAN), which is a key to an internal customer account file, containing, among other things, the customer's name and account information. If the customer's PAN matches the information in the customer account file, the system presents screen 2 to the customer. If the customer's PAN is not found, screen 4 is displayed, and the card is kept.
- ◆ At screen 2, the customer is prompted to enter his/her Personal Identification Number (PIN). If the PIN is correct (i.e., matches the information in the customer account file), the system displays screen 5; otherwise, screen 3 is displayed. The customer has three chances to get the PIN correct; after three failures, screen 4 is displayed, and the card is kept.

The SATM System

- ◆ On entry to screen 5, the system adds two pieces of information to the customer's account file: the current date, and an increment to the number of ATM sessions. The customer selects the desired transaction from the options shown on screen 5; then the system immediately displays screen 6, where the customer chooses the account to which the selected transaction will be applied.
- ◆ If **balance** is requested, the system checks the local ATM file for any unposted transactions, and reconciles these with the beginning balance for that day from the customer account file. Screen 14 is then displayed.
- ◆ If **deposit** is requested, the status of the Deposit Envelope slot is determined from a field in the Terminal Control File. If no problem is known, the system displays screen 7 to get the transaction amount. If there is a problem with the deposit envelope slot, the system displays screen 12. Once the deposit amount has been entered, the system displays screen 13, accepts the deposit envelope, and processes the deposit. The deposit amount is entered as an unposted amount in the local ATM file, and the count of deposits per month is incremented. Both of these (and other information) are processed by the Master ATM (centralized) system once per day. The system then displays screen 14.

The SATM System

- ◆ If **withdrawal** is requested, the system checks the status (jammed or free) of the withdrawal chute in the Terminal Control File. If jammed, screen 10 is displayed, otherwise, screen 7 is displayed so the customer can enter the withdrawal amount. Once the withdrawal amount is entered, the system checks the Terminal Status File to see if it has enough money to dispense. If it does not, screen 9 is displayed; otherwise the withdrawal is processed. The system checks the customer balance (as described in the Balance request transaction), and if there are insufficient funds, screen 8 is displayed. If the account balance is sufficient, screen 11 is displayed, and the money is dispensed. The withdrawal amount is written to the unposted local ATM file, and the count of withdrawals per month is incremented. The balance is printed on the transaction receipt as it is for a balance request transaction. After the cash has been removed, the system displays screen 14.
- ◆ When the No button is pressed in screens 10, 12, or 14, the system presents screen 15 and returns the customer's ATM card. Once the card is removed from the card slot, screen 1 is displayed. When the Yes button is pressed in screens 10, 12, or 14, the system presents screen 5 so the customer can select additional transactions.

The SATM System

- ◆ There is a surprising amount of information “buried” in the system description
 - e.g. the terminal only contains \$10 bills
- ◆ Very precise textual definition
- ◆ Deliberately simple example

Example 5. The Currency Converter

- ◆ The currency conversion program is another event-driven program that emphasizes code associated with a GUI.

The image shows a window titled "Currency converter". Inside the window, there are two text input fields. The first is labeled "US dollar amount" and the second is labeled "Equivalent in ...". Below these fields is a list of radio buttons with the following labels: "Brazil", "Canada", "European community", and "Japan". To the right of the radio buttons are three rounded rectangular buttons labeled "Compute", "Clear", and "Quit".

- ◆ This example nicely illustrates a description with UML and an object-oriented implementation

Example 6: Saturn Windshield Wiper Controller

- ◆ The windshield wiper on some Saturn automobiles is controlled by a lever with a dial.
- ◆ The lever has four positions: OFF, INT (for intermittent), LOW, and HIGH;
- ◆ and the dial has three positions, numbered simply 1, 2, and 3. The dial positions indicate three intermittent speeds, and the dial position is relevant only when the lever is at the INT position

c1. Lever	OFF	INT	INT	INT	LOW	HIGH
c2. Dial	n/a	1	2	3	n/a	n/a
a1. Wiper	0	4	6	12	30	60

Outline

- ◆ Basics of Software Testing
- ◆ Examples Used In Testing Discussions
- ◆ Discrete Mathematics for Testers
- ◆ Graph Theory for Testers

Discrete Mathematics

- ◆ The mathematical topics to be presented are tools for the tester
- ◆ They lead to rigor, precision and efficiency in testing
- ◆ Discrete mathematics is mostly relevant to Functional Testing
- ◆ Graph Theory is mostly relevant to Structural Testing

Set Theory

- ◆ A set is a collection or family or group or “bunch” of things called its elements
- ◆ In formal mathematics, “set” is an undefined term
- ◆ We usually use capital letters to refer to sets: A , B , Y , etc.
- ◆ If x is an element of a set A , we write $x \in A$

Set Theory - Notation

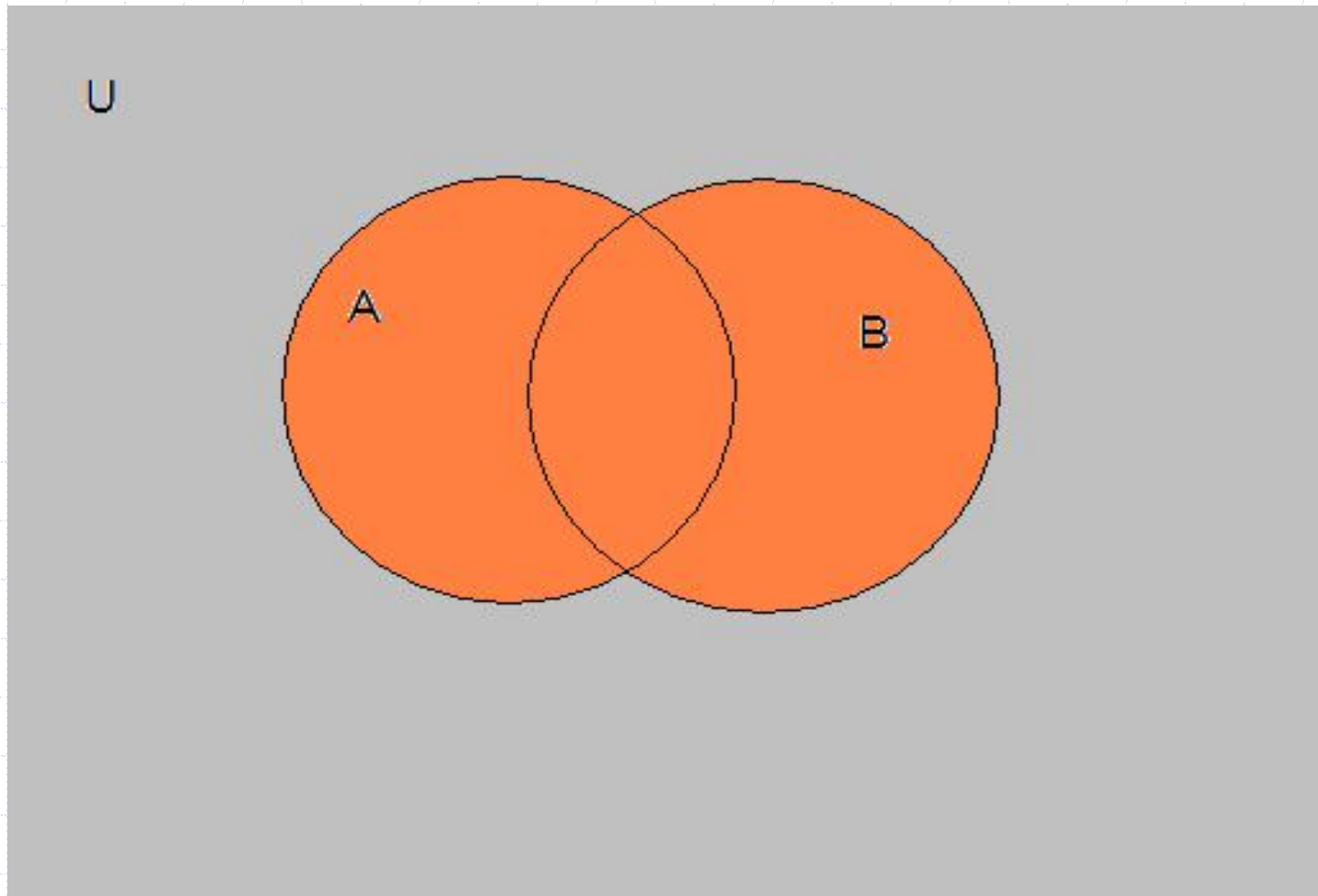
There are several ways to express sets

- ◆ For finite sets, we can just list the elements in braces: $A = \{1, 2, 3\}$ or $B = \{2, 4, 6, \dots, 20\}$
- ◆ For infinite sets, we can use a list as above: $C = \{1, 3, 5, 7, \dots\}$ as long as how to continue is unambiguous
- ◆ Or, we use set-builder notation:
 $D = \{x : x \text{ is an integer, } 11 \leq x \leq 20\} = \{11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$
- ◆ Example (bad): $F = \{3, 5, 7, \dots\}$

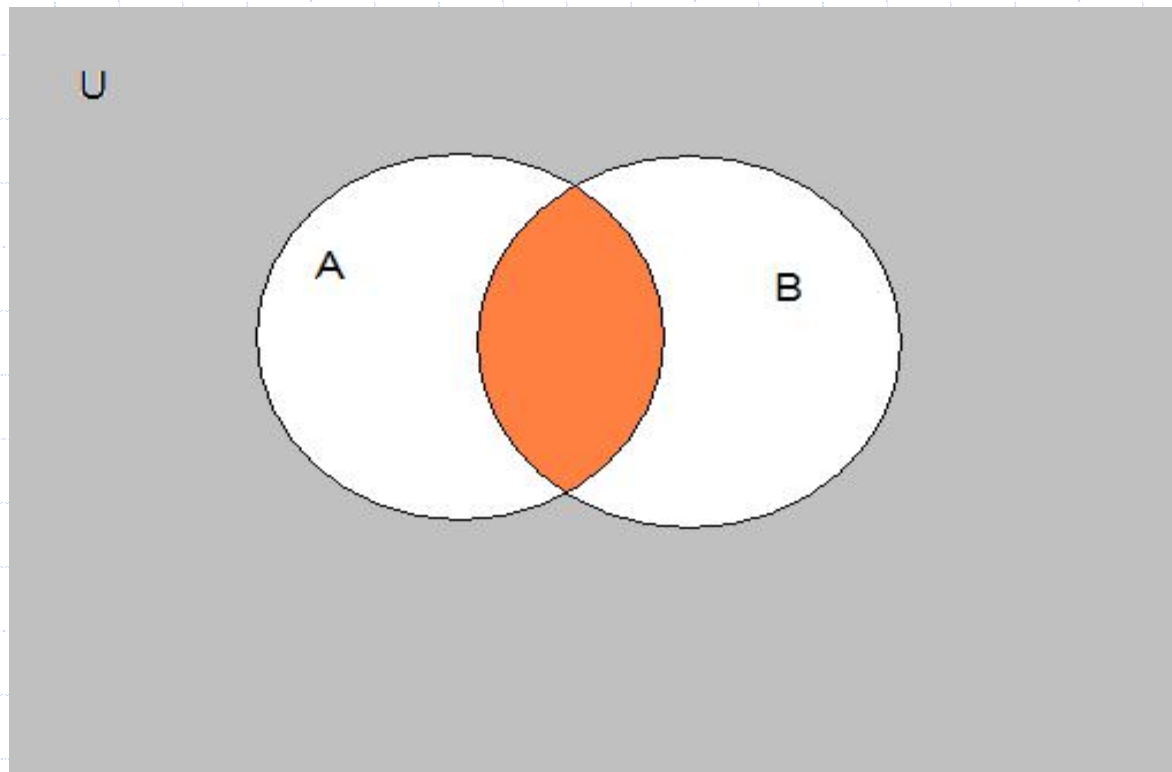
Set Theory - Operations

- ◆ Union: $A \cup B$ = the set of elements in either A or B or both
- ◆ Intersection: $A \cap B$ = the set of elements in both A and B
- ◆ Complement: To form the complement, you must know your Universe. Then A' = the set of elements not in A.
- ◆ Relative complement: $A - B = \{x : x \text{ in } A \text{ and } x \text{ not in } B\}$
- ◆ Symmetric difference: $A \oplus B = \{x : x \text{ in } A \oplus x \text{ in } B\}$
 $= (A \cup B) - (A \cap B)$

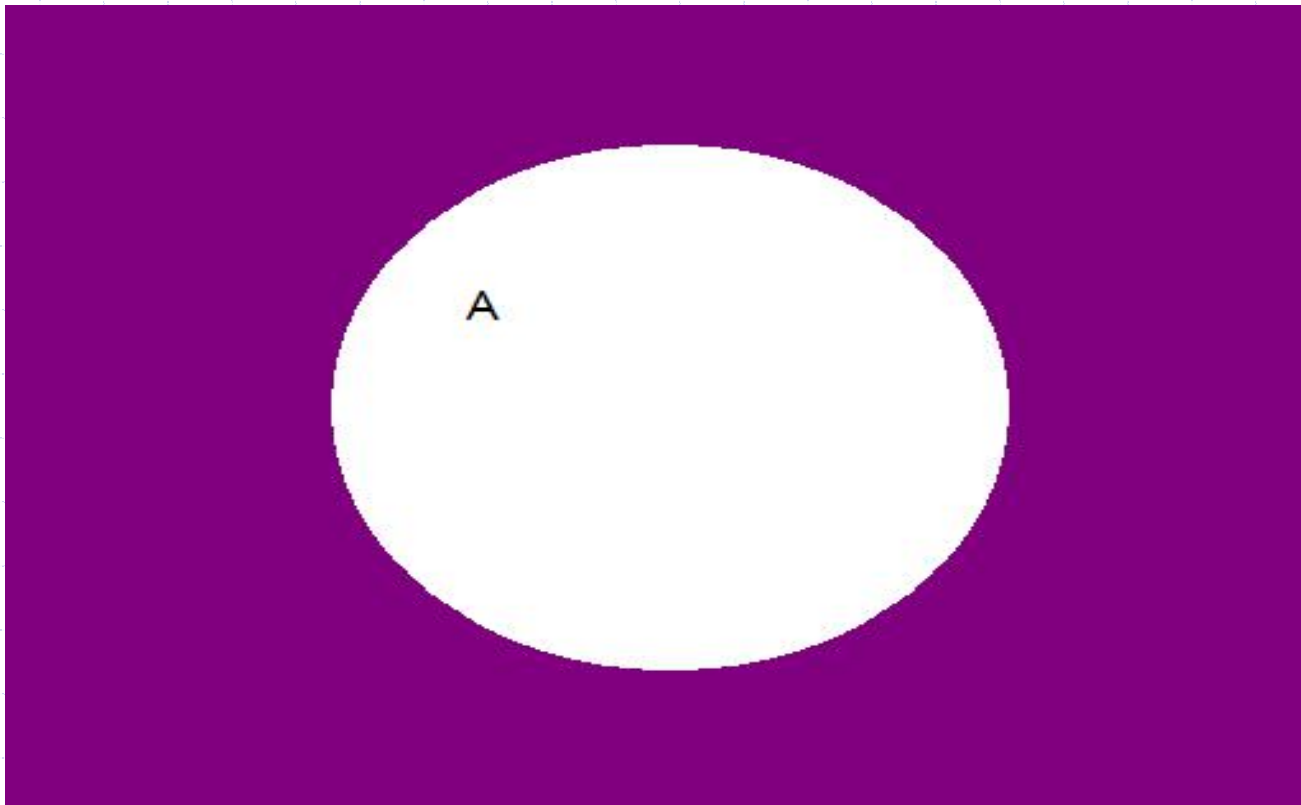
Set Union



Set Intersection



Set Complement



Set Theory - Examples

Suppose $A = \{1, 2, 3, 4, 5\}$, $B = \{2, 4, 6, 8\}$, and our universe is all positive integers.

Then $A \cup B = \{1, 2, 3, 4, 5, 6, 8\}$

$A \cap B = \{2, 4\}$

$A' = \{6, 7, 8, 9, 10, 11, 12, \dots\}$

$B' = \{1, 3, 5, 7, 9, 10, 11, 12, \dots\}$

Set Theory - Cartesian Product

- ◆ An ordered pair of elements is written as follows:
 - $\langle a, b \rangle$
 - if $a \neq b$, then $\langle a, b \rangle \neq \langle b, a \rangle$
- ◆ $A \times B$ is the set of all order pairs $\langle x, y \rangle$ where x comes from A and y comes from B .
- ◆ $\{1, 2, 3\} \times \{Q, \text{'dog'}\} = \{\langle 1, Q \rangle, \langle 2, Q \rangle, \langle 3, Q \rangle, \langle 1, \text{'dog'} \rangle, \langle 2, \text{'dog'} \rangle, \langle 3, \text{'dog'} \rangle\}$

Cartesian Product cont'd

- ◆ This concept occurs often in testing.
- ◆ Example: Suppose there are 4 possible sets of data periods: 1 week, 1 month, 3 months, 1 year
And suppose there are 3 possible graphs that can be displayed:
Amount invested, profit or loss, cumulative earnings
How many test cases?
 $A = \{1W, 1M, 3M, 1Y\}$, $B = \{Inv, P\&L, Earn\}$
 $A \times B$ has 12 elements
- ◆ We write $|A| = 4$, $|B| = 3$, $|A \times B| = 12$

Set Relations

- ◆ The empty set has no elements – it's denoted by \emptyset
- ◆ If every element of set A is also an element of set B, we say A is a subset of B and write $A \subset B$ or $A \subseteq B$.
- ◆ For any set X, $\emptyset \subset X$
- ◆ Two sets X and Y are equal, $X = Y$, if and only if $X \subseteq Y$ and $Y \subseteq X$
- ◆ If two sets E and F have no elements in common, they are disjoint and we write $E \cap F = \emptyset$

Partitions

We will definitely see this again

If we write a set A as a union of disjoint non-empty sets, then we say we have a partition of A .

In set notation: $A = A_1 \cup A_2 \cup \dots \cup A_n$, with $A_i \cap A_j = \emptyset$ when $i \neq j$

When we talk about Functional Testing we will discuss Equivalence Partitioning

Functions I

- ◆ A function is a mapping from one set to another.

- ◆ Example: $A = \{\text{the set of months in 2004}\}$,
 $B = \{\text{positive integers}\}$

For each element m of A we set $f(m) = \text{the number of days in } m$.

- ◆ Then $f : A \rightarrow B$; A is the ***domain*** of f , and B contains the ***range*** of f .

- ◆ What is the actual range of f here? What would $f(\text{'March'})$ be?

Functions II

- ◆ If $f : A \rightarrow B$ and C is a subset of A , then $f(C) = \{y \in B : \text{for some } x \in C, f(x) = y\}$
- ◆ $f(C)$ is the *image* of set C
- ◆ If the image of A is all of B , f is called *onto*
- ◆ If $f(a) = f(b)$ always implies that $a = b$, f is called *1-1* (one-to-one)
- ◆ If $f : A \rightarrow B$ is both *1-1* and *onto*, it has an inverse $f^{-1} : B \rightarrow A$

Function Composition

- ◆ If $f : A \rightarrow B$ and $g : B \rightarrow C$, then we can define a function $h : A \rightarrow C$ called the composition of f and g .
 $h(x) = g(f(x))$
- ◆ Notation: $h = g \circ f$
- ◆ Example: Suppose $A = B = C = \{\text{the set of positive integers}\}$, and $f : A \rightarrow B$ is defined by $f(a) = a + 1$; $g : B \rightarrow C$ is defined by $g(b) = 2b$.
Then $g \circ f(a) = 2a + 2$

Relations

- ◆ If A is a set, then a relation R on A is any subset of $A \times A$
- ◆ Thus a relation R is a set or family of ordered pairs of elements of A
- ◆ The relation is:
 - Reflexive if (a, a) is in R for every a in A
 - Symmetric if (a, b) in R always implies (b, a) is in R
 - Transitive if (a, b) and (b, c) in R always implies that (a, c) is in R
 - Antisymmetric if (a, b) and (b, a) in R implies $a = b$

Examples of Relations

- ◆ An *order relation* is reflexive, antisymmetric and transitive
- ◆ An *equivalence relation* is reflexive, symmetric and transitive
- ◆ Equivalence relations induce equivalence classes
- ◆ Consider the relation $a \leq b$ on the positive integers
- ◆ Consider the relation $a = b$ on the positive integers

Propositional Logic

- ◆ A proposition is a sentence that can be either true or false (T or F)
- ◆ If p and q are propositions, we have the following expressions:

p and q

$$p \wedge q$$

p or q

$$p \vee q$$

p implies q

$$p \rightarrow q$$

Not p

$$\neg p$$

p iff q

$$p \leftrightarrow q$$

p xor q

$$p \oplus q$$

- ◆ Each expression has its truth table
- ◆ p and q are considered logically equivalent when their truth tables are identical

Probability

The frequency definition

- ◆ If an “event” can occur in N ways (all equally “probable”), and if M of those ways are “favorable events”, then the *probability* of a favorable event is defined to be M/N . If F , say, is the set of favorable events, then we write $\Pr(F) = M/N$.
- ◆ What is the probability that a randomly chosen month has exactly 30 days in it?
- ◆ What is the probability that a randomly chosen month has 30 days in it?

Probability - Beware!

- ◆ Let p be the proposition “It rains tomorrow”
- ◆ Let q be the proposition “It does not rain tomorrow”
- ◆ Suppose both p and q have probability $\frac{1}{2}$ of occurring
- ◆ What is $\Pr(p \wedge q)$?
 - $\Pr(p \text{ and } q) = \Pr(p) * \Pr(q) = \frac{1}{2} * \frac{1}{2} = \frac{1}{4} ???$

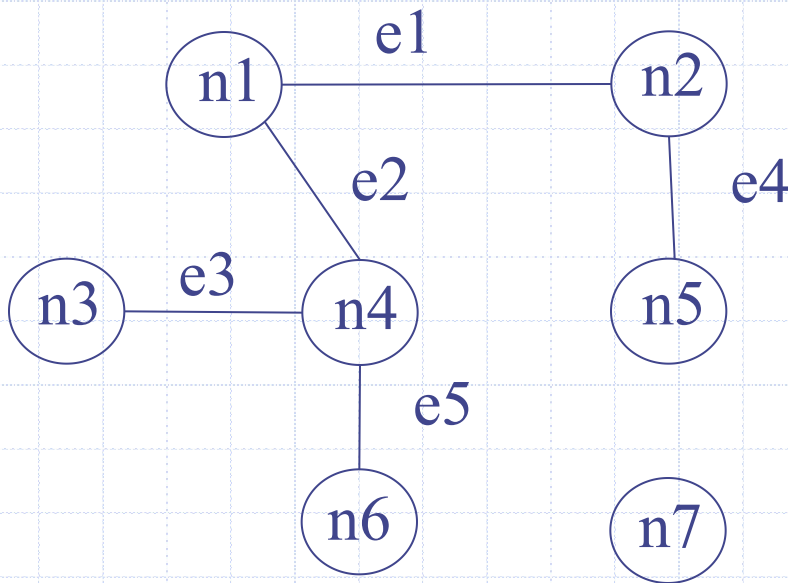
Outline

- ◆ Basics of Software Testing
- ◆ Testing Activities and Context
- ◆ Examples Used In Testing Discussions
- ◆ Discrete Mathematics for Testers
- ◆ Graph Theory for Testers

Graph

- ◆ A graph $G = (V, E)$ is composed of a finite and non empty set $V = \{n_1, n_2, n_m\}$ of nodes and a set $E = \{e_1, e_2, e_p\}$ of edges, where each edge $e_k = \{n_i, n_j\}$, for some nodes n_i, n_j in V .
 - Also called an undirected graph.

Example Graph



$V = \{n1, n2, n3, n4, n5, n6, n7\}$

$E = \{e1, e2, e3, e4, e5\}$
 $= \{\{n1, n2\}, \{n1, n4\}, \{n3, n4\}, \{n2, n5\}, \{n4, n6\}\}$

Degree of a node: the number of edges that have the node as an endpoint. Example: $\deg(n1) = 2$, $\deg(n7) = 0$, $\deg(n4) = 3$.

Paths

- ◆ A **path** is a sequence of edges such that, for any adjacent pair of edges e_i, e_j in the sequence, the edges share a common node as an endpoint.
- ◆ Paths can be described either as a sequence of edges, or as a sequence of nodes.
- ◆ Paths are defined between a *source* node and a *target* node.

Example Paths

Path (from-to)	Node sequence	Edge sequence
n1 – n5	n1, n2, n5	e1, e4
n6 – n5	n6, n4, n1, n2, n5	e5, e2, e1, e4
n3 – n2	n3, n4, n1, n2	e3, e2, e1

Connectedness

- ◆ Nodes n_i, n_j are connected iff there is a path from node n_i to node n_j
- ◆ “Connectedness” is an equivalence relation on the node set of a graph
- ◆ “Connectedness” defines a partition on the node set of a graph.
- ◆ A component of a graph is a maximal set of connected nodes
- ◆ Example: Two components in the sample graph $C1 = \{n1, n2, n3, n4, n5, n6\}$, and $C2 = \{n7\}$

Condensation Graphs

- ◆ Given a graph $G = (V, E)$, its condensation graph is formed by replacing each component by a condensing node.



$$C_1 = \{n1, n2, n3, n4, n5, n6\}$$

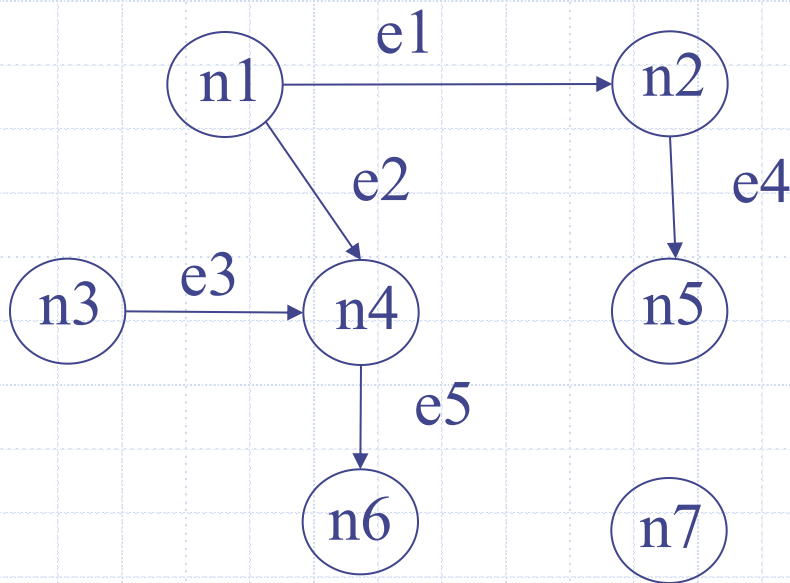
Complexity of a Graph

- ◆ A measure of the complexity of a graph is its **cyclomatic complexity** measure.
- ◆ The cyclomatic complexity of a graph G , is a number $V(G) = e - n + 2$, where e is the number of edges in the graph, and n is the number of vertices (nodes) in the graph.

Directed Graphs

- ◆ A **directed** graph (or **DiGraph**) $D = (V, E)$, consists of a finite set of vertices $V = \{n_1, n_2, \dots, n_m\}$, and a set of edges $E = \{e_1, e_2, e_p\}$, where each edge $e_k = \langle n_i, n_j \rangle$ is an ordered pair of nodes in V .

Example Directed Graph



$V = \{n1, n2, n3, n4, n5, n6, n7\}$

$E = \{e1, e2, e3, e4, e5\}$
 $= \{ \langle n1, n2 \rangle, \langle n1, n4 \rangle, \langle n3, n4 \rangle, \langle n2, n5 \rangle, \langle n4, n6 \rangle \}$

Properties of Directed Graphs

- ◆ The **indegree** (or fan-in) of a node in a directed graph is the number of distinct edges that have the node as a terminal node (target node). Example: $\text{indeg}(n1) = 0$, $\text{indeg}(n4) = 2$
- ◆ The **outdegree** (or fan-out) of a node in a directed graph is the number of distinct edges that have the node as a start node (source node). Example: $\text{outdeg}(n1) = 2$, $\text{outdeg}(n6) = 0$
- ◆ A node with in indegree = 0 is called a **source** node ($n1, n3, n7$)
- ◆ A node with outdegree = 0 is called a **sink** node ($n5, n6, n7$)
- ◆ A node with indegree $\neq 0$ and outdeg $\neq 0$ is called a **transfer** node ($n2, n4$)
- ◆ A node that is both a source and a sink is an **isolated** node
- ◆ The source and sink nodes in a graph constitute the external boundary of the graph

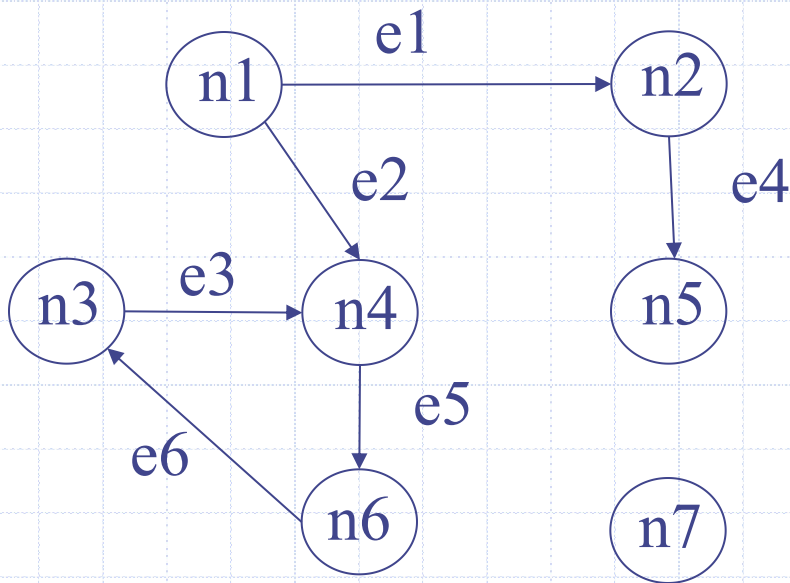
Paths and Semi-Paths

- ◆ A **directed path** is a sequence of edges such that, for any adjacent set of edges e_i, e_j in the sequence, the terminal node of e_i is the initial node of e_j
- ◆ A **cycle** is a directed path that begins and ends in the same node
- ◆ A **directed semi-path** is a sequence of edges such that, for at least one adjacent pair of edges e_i, e_j in the sequence, the initial node of the first edge is the initial node of the second edge or the terminal node of the first edge is the terminal node of the second edge
- ◆ Directed paths are also called **chains**
- ◆ Example: There is a path from $n1$ to $n6$,
 - There is a semi-path between $n1$ and $n3$
 - There is a semi-path between $n2$ and $n4$
 - There is a semi-path between $n5$ and $n6$

N-Connectedness

- ◆ Two nodes n_i and n_j in a directed graph are:
 - 0-connected: iff there is no path from n_i to n_j
 - 1-connected: iff there is a semi-path but no path between n_i, n_j
 - 2-connected: iff there is a path between n_i, n_j
 - 3-connected: iff there is a path from n_i to n_j and a path from n_j to n_i

Example N-Connectedness

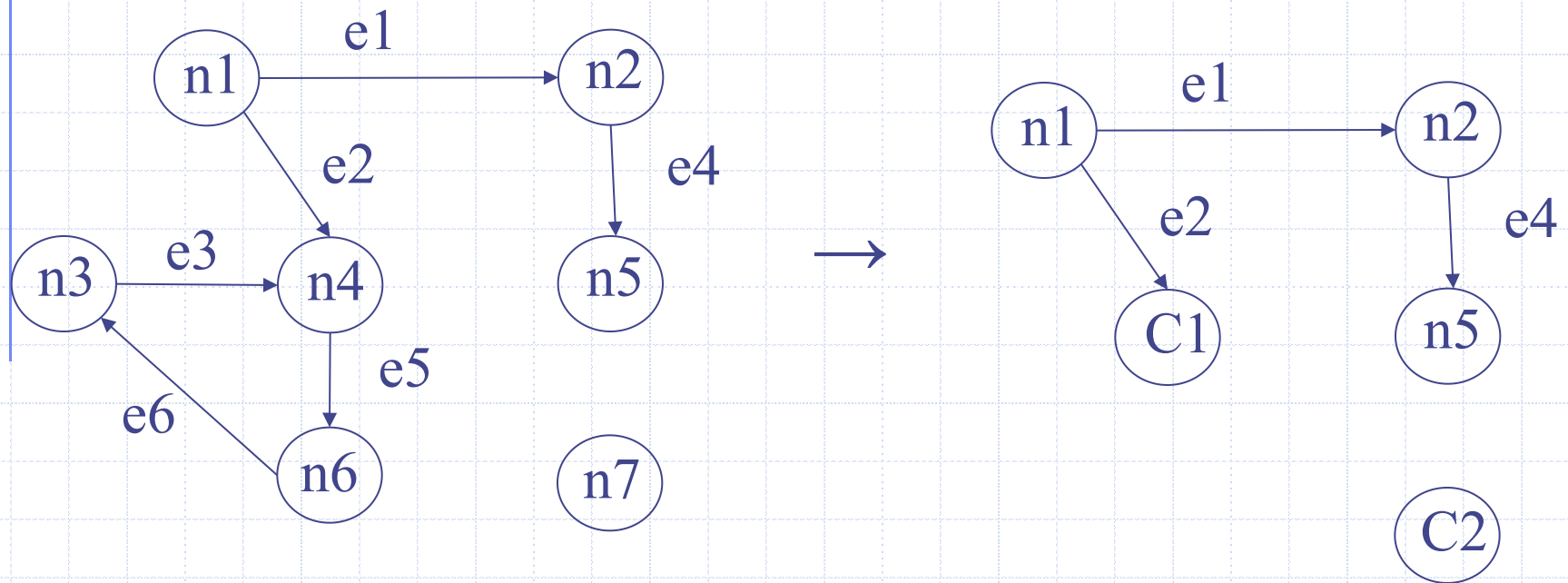


n1 and n7 are 0-connected
n2 and n6 are 1-connected
n1 and n6 are 2-connected
n3 and n6 are 3-connected

Strong Components

- ◆ A *strong component of a directed graph* is a maximal set of 3-connected nodes
- ◆ Practically loops and isolated nodes are removed and replaced by strong components (Directed Acyclic Graphs, DAG)

Example Condensation DiGraph

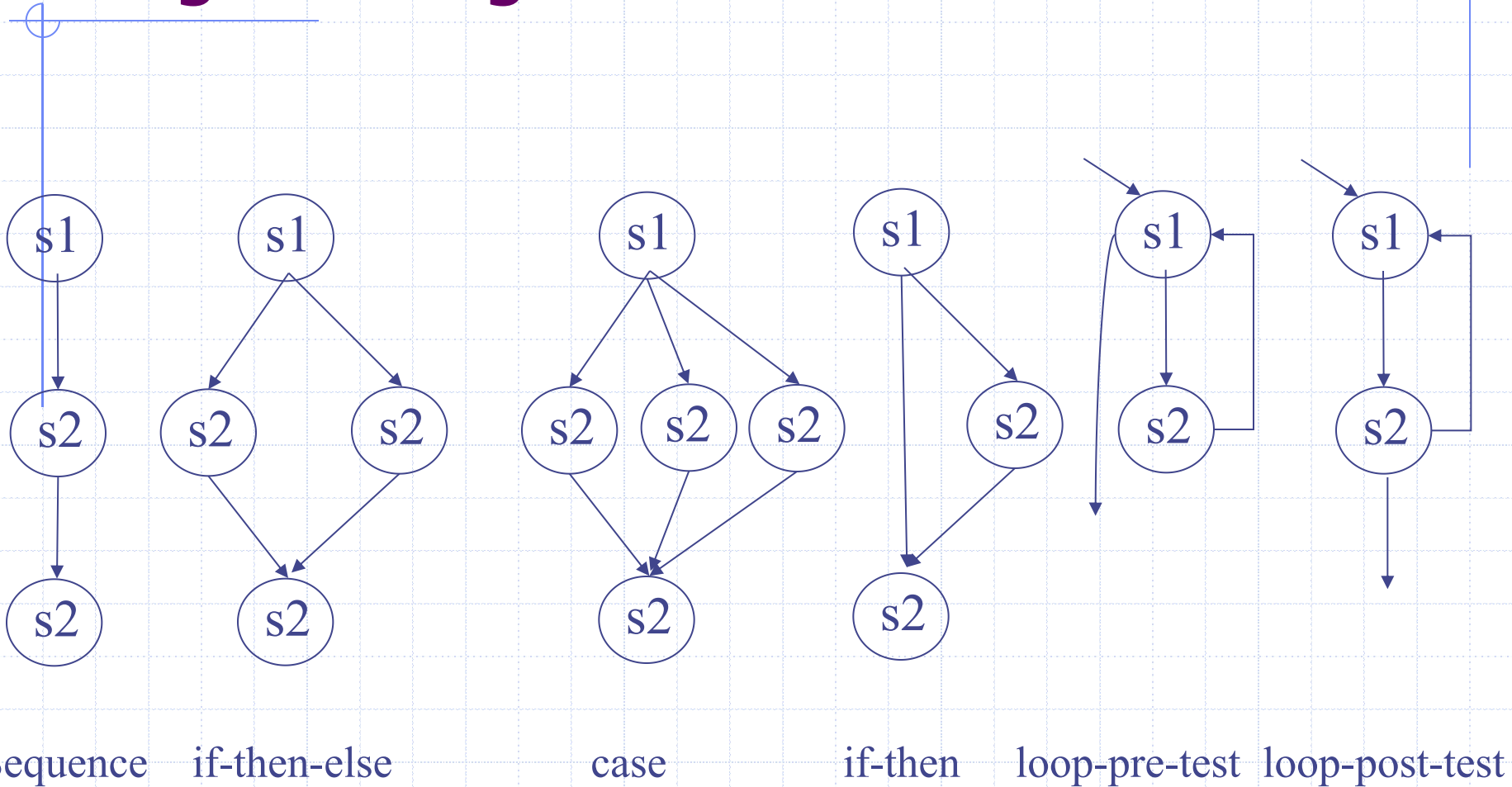


$$C_1 = \{n_3, n_4, n_6\}$$
$$C_2 = \{n_7\}$$

Program Graphs

- ◆ Given a program written in an imperative programming language, its program graph is a directed graph in which:
 - Nodes are program statements (or statement fragments), and edges represent flow of control
- ◆ Control flow of imperative programs is defined by sequencing, iteration, and choice constructs
- ◆ Primarily used at the unit testing level

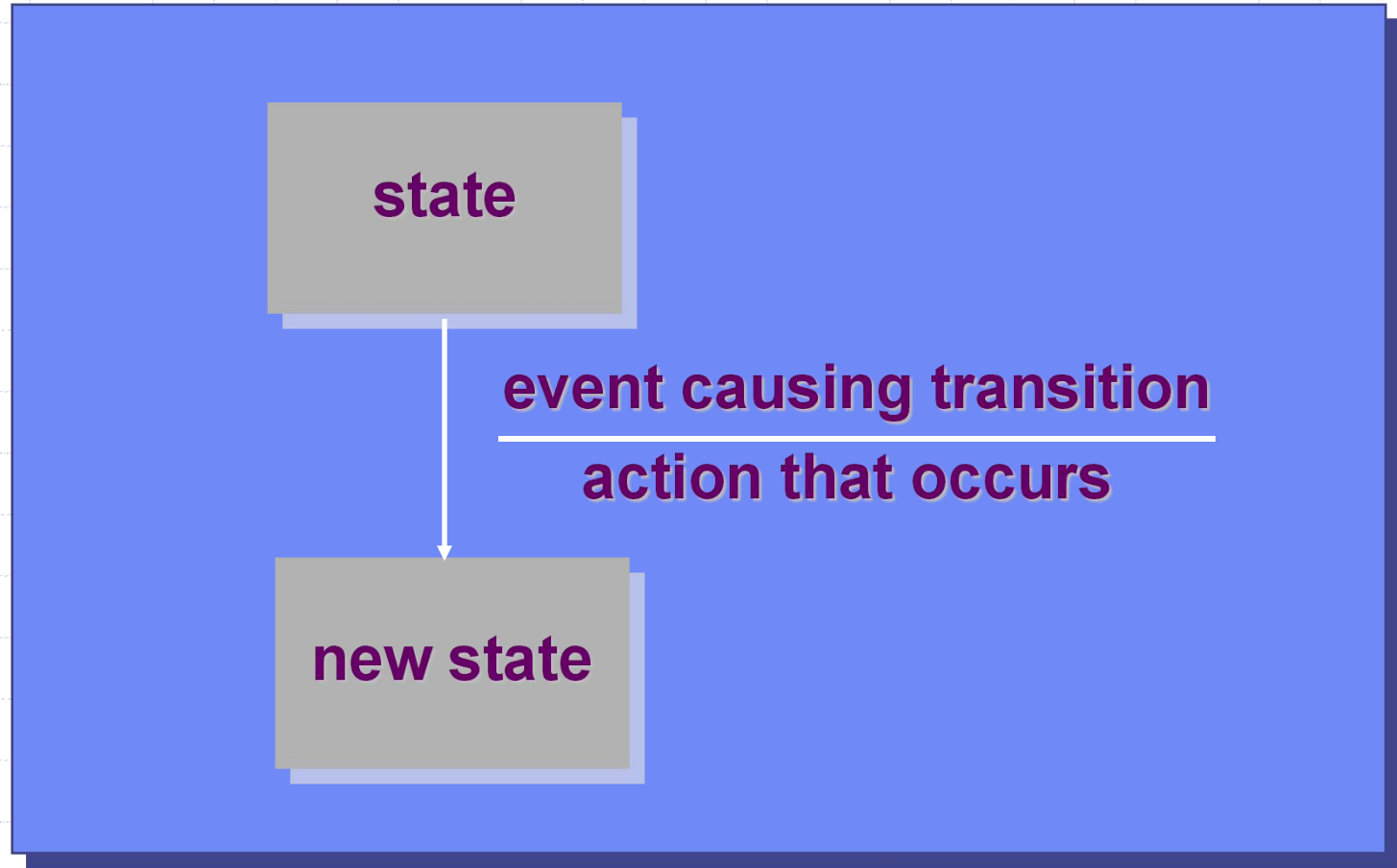
Digraphs of the Structured Programming Constructs



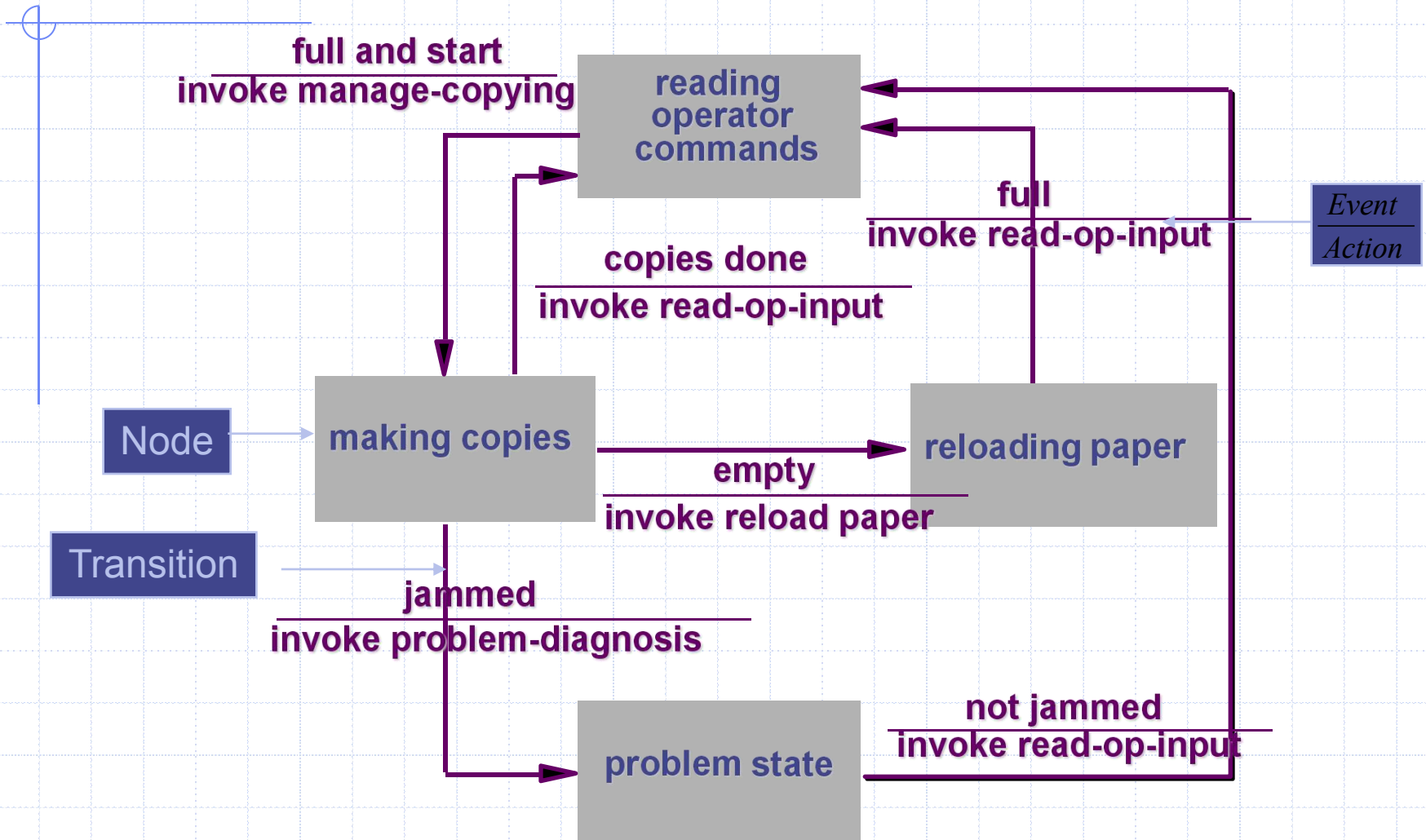
Finite State Machines

- ◆ A Finite State Machine (FSM) is a directed graph (S, T, Ev, Act) , in which S is a set of nodes, T is a set of edges, and Ev and Act are sets of events and actions associated with the transitions in T .
- ◆ Primarily used to describe system level behavior
 - Have become a fairly standard notation for requirements specification

FSM Notation



Example FSM



Questions

- ◆ Express $A \cup B$ in words.
- ◆ Express $A \cap B$ in words.
- ◆ What is an equivalence relation?
- ◆ Describe the Finite State Machines (FSM).

Reference

- ◆ *Paul C. Jorgensen, Software Testing: A Craftsman's Approach, 4th Edition, Chapter 1~4*