From Chapter 1
Criteria Based on Four Structures

Structures: Ways to model software

1. Graphs

2. Logical Expressions

   (not X or not Y) and A and B

3. Input Domain Characterization

   A: \{0, 1, >1\}
   B: \{600, 700, 800\}
   C: \{swe, cs, isa, infs\}

4. Syntactic Structures

   if (x > y)
   
   z = x - y;
   else
   
   z = 2 * x;
Test Coverage Criteria

A tester’s job is simple: Define a model of the software, then find ways to cover it

- **Test Requirements**: Specific things that must be satisfied or covered during testing

- **Test Criterion**: A collection of rules and a process that define test requirements

Testing researchers have defined dozens of criteria, but they are all really just a few criteria on four types of structures.

Coverage Overview

Four Structures for Modeling Software

- **Graphs**
  - Source
  - Specs
  - Design
  - Use cases

- **Logic**
  - FSMs
  - Specs
  - DNF

- **Input Space**
  - Source
  - Models
  - Integ
  - Input

- **Syntax**

Introduction to Software Testing (Ch 1-5) © Ammann & Offutt
Ch. 2: Graph Coverage Criteria (1)

- Concepts for graphs
  - Visiting, touring
  - Tests and test paths
  - Structural and data flow coverage
  - Prime Paths
  - Defs and uses

- Simple graph coverage criteria

**Node Coverage (NC):** TR contains each reachable node in $G$.

**Edge Coverage (EC):** TR contains each reachable path of length up to 1 in $G$. 
Graph Coverage Criteria (2)

- Covering multiple edges

**Edge-Pair Coverage (EPC)**: TR contains each reachable path of length up to 2 in G.

**Complete Path Coverage (CPC)**: TR contains all paths in G.

**Specified Path Coverage (SPC)**: TR contains a set S of test paths, where S is supplied as a parameter.

Graph Coverage Criteria (3)

- Handling loops

**Simple Round Trip Coverage (SRTC)**: TR contains at least one round-trip path for each reachable node in G that begins and ends a round-trip path.

**Complete Round Trip Coverage (CRTC)**: TR contains all round-trip paths for each reachable node in G.

**Prime Path Coverage (PPC)**: TR contains each prime path in G.
Graph Coverage Criteria (4)

- Data flow coverage

\textbf{All-defs coverage (ADC)}: For each set of du-paths $S = du(n, v)$, TR contains at least one path $d$ in $S$.

\textbf{All-uses coverage (AUC)}: For each set of du-paths to uses $S = du(n_i, n_j, v)$, TR contains at least one path $d$ in $S$.

\textbf{All-du-paths coverage (ADUPC)}: For each set $S = du(n_i, n_j, v)$, TR contains every path $d$ in $S$.

Graph Coverage Criteria Subsumption
Ch. 3 Logic Coverage Criteria (1)

- Concepts in logical expressions
  - Active clauses
  - Determination of the predicate
- Simple logical expression criteria

**Predicate Coverage (PC):** For each $p$ in $P$, $TR$ contains two requirements: $p$ evaluates to true, and $p$ evaluates to false.

**Clause Coverage (CC):** For each $c$ in $C$, $TR$ contains two requirements: $c$ evaluates to true, and $c$ evaluates to false.
Logic Coverage Criteria (2)

- Testing multiple clauses at once

**Combinatorial Coverage (CoC)**: For each \( p \) in \( P \), TR has test requirements for the clauses in \( C_p \) to evaluate to each possible combination of truth values.

**Active Clause Coverage (ACC)**: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j, j \neq i \), so that \( c_i \) determines \( p \). TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false.

Logic Coverage Criteria (3)

**General Active Clause Coverage (GACC)**: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j, j \neq i \), so that \( c_i \) determines \( p \). TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false. The values chosen for the minor clauses \( c_j \) do not need to be the same when \( c_i \) is true as when \( c_i \) is false, that is, \( c_j(c_i = \text{true}) = c_j(c_i = \text{false}) \) for all \( c_j \) OR \( c_j(c_i = \text{true}) \neq c_j(c_i = \text{false}) \) for all \( c_j \).

**Restricted Active Clause Coverage (RACC)**: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j, j \neq i \), so that \( c_i \) determines \( p \). TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false. The values chosen for the minor clauses \( c_j \) **must be the same** when \( c_i \) is true as when \( c_i \) is false, that is, it is required that \( c_j(c_i = \text{true}) = c_j(c_i = \text{false}) \) for all \( c_j \).
**Logic Coverage Criteria (4)**

- Practical version

**Correlated Active Clause Coverage (CACC):** For each \( p \) in \( P \) and each major clause \( ci \) in \( Cp \), choose minor clauses \( cj, j \neq i \), so that \( ci \) determines \( p \). TR has two requirements for each \( ci \):

  - \( ci \) evaluates to true and \( ci \) evaluates to false. The values chosen for the minor clauses \( cj \) must cause \( p \) to be true for one value of the major clause \( ci \) and false for the other, that is, it is required that \( p(ci = true) \neq p(ci = false) \).

**Logic Coverage Criteria (5)**

- Inactive clauses

**Inactive Clause Coverage (ICC):** For each \( p \) in \( P \) and each major clause \( ci \) in \( Cp \), choose minor clauses \( cj, j \neq i \), so that \( ci \) does not determine \( p \). TR has four requirements for each \( ci \):

  - \( ci \) evaluates to true with \( p \) true, \( ci \) evaluates to false with \( p \) true, \( ci \) evaluates to true with \( p \) false, and \( ci \) evaluates to false with \( p \) false.

**General Inactive Clause Coverage (GICC):** For each \( p \) in \( P \) and each major clause \( ci \) in \( Cp \), choose minor clauses \( cj, j \neq i \), so that \( ci \) does not determine \( p \). The values chosen for the minor clauses \( cj \) do not need to be the same when \( ci \) is true as when \( ci \) is false, that is, \( cj(ci = true) = cj(ci = false) \) for all \( cj \) OR \( cj(ci = true) \neq cj(ci = false) \) for all \( cj \).

**Restricted Inactive Clause Coverage (RICC):** For each \( p \) in \( P \) and each major clause \( ci \) in \( Cp \), choose minor clauses \( cj, j \neq i \), so that \( ci \) does not determine \( p \). The values chosen for the minor clauses \( cj \) must be the same when \( ci \) is true as when \( ci \) is false, that is, it is required that \( cj(ci = true) = cj(ci = false) \) for all \( cj \).
Logic Coverage Criteria Subsumption

Combinatorial Clause Coverage
\[ \text{COC} \]

Restricted Active Clause Coverage
\[ \text{RACC} \]

Restricted Inactive Clause Coverage
\[ \text{RICC} \]

Correlated Active Clause Coverage
\[ \text{CACC} \]

General Inactive Clause Coverage
\[ \text{GICC} \]

General Active Clause Coverage
\[ \text{GACC} \]

Clause Coverage
\[ \text{CC} \]

Predicate Coverage
\[ \text{PC} \]

Coverage Overview

Four Structures for Modeling Software

Input Space

Graphs

Logic

Syntax

Source

Specs

FSMs

DNF

Source

Design

Use cases

Source

Models

Integ

Input
Ch. 4: Input Space Criteria (1)

- Concepts for input space criteria
  - Input domains
  - No difference with level of software abstraction (unit, integration, etc.)
  - Partitioning, domains, blocks
  - Constraints among partitions
- Simple input space criterion

Each Choice (EC): One value from each block for each partition must be used in at least one test case.

Input Space Criteria (2)

- Combining values

Pair-Wise (PW): A value from each block for each partition must be combined with a value from every block for each other partition.

t-Wise (TW): A value from each block for each group of t partitions must be combined.
Input Space Criteria (2)

- Using domain knowledge of the program

**Base Choice (BC)**: A base choice block is chosen for each partition, and a base test is formed by using the base choice for each partition. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other parameter.

**Multiple Base Choice (MBC)**: One or more base choice blocks are chosen for each partition, and base tests are formed by using each base choice for each partition. Subsequent tests are chosen by holding all but one base choice constant for each base test and using each non-base choices in each other parameter.

**ISP Coverage Criteria Subsumption**

```
     All Combinations Coverage
          AC
      /   \
T-Wise Coverage    Multiple Base Choice Coverage
      /     \                                   /   \
Pair-Wise Coverage Base Choice Coverage        PW   BC
          /   \         /   \
Each Choice Coverage  EC
```

Introduction to Software Testing (Ch 1-5) © Ammann & Offutt 21

Introduction to Software Testing (Ch 1-5) © Ammann & Offutt 22
Ch. 5 : Syntax-based Criteria (1)

- Concepts for syntax-based testing
  - Grammars
  - Regular expressions, finite state machines
  - Mutation testing
  - Simple syntax-based criteria

**Terminal Symbol Coverage (TSC)**: TR contains each terminal symbol $t$ in the grammar $G$.

**Production Coverage (PC)**: TR contains each production $p$ in the grammar $G$.

**Derivation Coverage (DC)**: TR contains every possible string that can be derived from the grammar $G$. 
Mutation Testing

• Grammars describe both valid and invalid strings
• Both types can be produced as mutants
• A mutant is a variation of a valid string
• Mutation is based on “mutation operators” and “ground strings”

• Ground string: A string in the grammar
  • The term “ground” is used as a reference to algebraic ground terms

• Mutation Operator: A rule that specifies syntactic variations of strings generated from a grammar

• Mutant: The result of one application of a mutation operator
  • A mutant is a string

Syntax-based Criteria (2)

• Mutation criteria

**Mutation Coverage (MC):** For each \( m \in M \), TR contains exactly one requirement, to kill \( m \).

**Mutation Operator Coverage (MOC):** For each mutation operator, TR contains exactly one requirement, to create a mutated string \( m \) that is derived using the mutation operator.

**Mutation Production Coverage (MPC):** For each mutation operator, TR contains several requirements, to create one mutated string \( m \) that includes every production that can be mutated by that operator.
Summary of Criteria

• Chapter 2
  – Eleven criteria, five recommended

• Chapter 3
  – Eight criteria, three recommended

• Chapter 4
  – Six criteria, two recommended

• Chapter 5
  – Six criteria, three recommended

• Total: Thirty-one criteria, thirteen recommended