Introduction to Software Testing
Logic Coverage (Ch. 8.1.1)

Software Testing & Maintenance
SWE 437
http://go.gmu.edu/swe437

Dr. Brittany Johnson-Matthews
(Dr. B for short)
Semantic Logic Coverage (8.1)

Logical expressions can come from many sources

- Decisions in programs
- Decisions in **UML** activity graphs and **finite state machines**
- **Requirements**, both formal and informal
- **SQL** queries

Covering logic expressions is required by the US Federal Aviation Administration for safety critical software

- Used by other transportation industries

Used by Electronc Arts (EA) game company

- FIFA, Battlefield, …

Tests are intended to choose some subset of the total number of truth assignments to the expressions
Logic predicates and clauses

A **predicate** is an expression that evaluates to a **boolean** value.

Predicates can contain:
- boolean variables
- non-boolean variables that contain $>$, $<$, $==$, $<=$, $!=$
- boolean function calls

Internal structure is created by **logical operators**
- $\neg$ or ! - the negation operator
- $\land$ or & – the and operator
- $\lor$ or | – the or operator
- $\rightarrow$ – the implication operator
- $\Theta$ or xor – the exclusive or operator
- $\leftrightarrow$ – the equivalence operator

A **clause** is a predicate with no logical operators.
Example

\[ P = (a \& (b \mid c)) \]

P has three clauses:

- a, b, and c

Most predicates have few clauses.

- 88.5% have 1 clause
- 9.5% have 2 clauses
- 1.35% have 3 clauses
- Only 0.65% have 4 or more!
Logic Coverage Criteria (8.1.1)

We use predicates in testing as follows:
Develop a model of the software as one or more predicates
Require tests to satisfy some combination of clauses

**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.

PC: Each full predicate evaluates to true and false (2 tests)

CC: Each clause in each predicate evaluates to true and false (at least 2 tests per predicate)
Give predicate coverage (PC) and clause coverage (CC) abstract tests for our example predicate.

“Abstract tests” include truth assignments for each clause, for example:

\[ a = \text{true} \]
Problems with PC and CC

PC does not **fully exercise** all the clauses, especially in the presence of short circuit evaluation

CC does not always **ensure PC**

That is, we can satisfy CC without causing the predicate to be both true and false

This is *definitely* not what we want!

The simplest solution is to test **all combinations** …
Combinatorial Coverage (CoC)

CoC requires **every possible combination**

Sometimes called **Multiple Condition Coverage (MCC)**

Every possible combination of truth values

\[2^N\] possibilities, where \(N\) is the number of clauses

**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.
In-class Exercise

\[ P = (a \& (b \mid c)) \]

Give abstract tests to satisfy combinatorial coverage (CoC) for our example predicate.

Hint: There should be 8
Combinatorial Coverage (CoC)

This is simple, neat, clean, and comprehensive …
But can be **expensive**
  - Impractical for predicates with more than 3 or 4 clauses
The literature has lots of suggestions – some confusing
The general idea is simple:

**Test each clause independently from the other clauses**

Getting the details right is hard
What exactly does “independently” mean?
The book presents this idea as “**making clauses active**” …