Software Testing, Quality Assurance and Maintenance	Winter 2010
Lecture 15 — February 5, 2010	
Patrick Lam	*version 2*

Logic Coverage

We now shift from graphs to logical expressions, for instance:

```
if (visited && x > y || foo(z))
```

Graphs are made up of nodes, connected by edges. Logical expressions, or predicates, are made up of clauses, connected by operators.

Predicates. A *predicate* is an expression that evaluates to a logical value. Example:

$$a \wedge b \leftrightarrow c$$

Here are the operators we allow, in order of precedence (high to low):

- $\neg$: negation
- $\wedge$: and (not short-circuit)
- $\vee$: or (not short-circuit)
- $\rightarrow$: implication
- $\oplus$: exclusive or
- $\leftrightarrow$: equivalence

We do not allow quantifiers; they are harder to reason about. Note also that our operators are not quite the same as the ones in typical programming languages.

Clauses. Predicates without logical operators are *clauses*; clauses are, in some sense, "atomic". The following predicate contains three clauses:

```
(x > y) || foo(z) && bar
```

Logical Equivalence. Two predicates may be logically equivalent, e.g.

$$x \wedge y \vee z \equiv (x \vee z) \wedge (y \vee z)$$

and these predicates are not equivalent to $x \leftrightarrow z$. Equivalence is harder with short-circuit operators.

Sources of Predicates: source code, finite state machines, specifications.

Logic Expression Coverage Criteria

We'll use the following notation:

- P: a set of predicates;
- C: all clauses making up the predicates of P.

Let $p \in P$. Then we write C_p for the clauses of the predicate p, i.e.

$$C_p = \{c \mid c \in p\}; \qquad C = \bigcup_{p \in P} C_p$$

Given a set of predicates P, we might want to cover all of the predicates.

Criterion 1 Predicate Coverage *(PC). For each $p \in P$, TR contains two requirements: 1) p evaluates to true; and 2) p evaluates to false.*

PC is analogous to edge coverage on a CFG. (Let P be the predicates associated with branches.)

Example:

PC gives a very coarse-grained view of each predicate. We can break up predicates into clauses to get more details.

Criterion 2 Clause Coverage *(CC). For each $c \in C$, TR contains two requirements: 1) c evaluates to true; 2) c evaluates to false.*

Example:

Subsumption. Are there subsumption relationships between CC and PC?

The obvious exhaustive approach: try everything. (This obviously subsumes everything else).

Criterion 3 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.*

This is also known as multiple condition coverage. Unfortunately, the number of test requirements, while finite, grows ______________ and is hence unscalable.