Logic Coverage Overview part 1

Idaho State University | Computer Science

Isaac Griffith

CS 4422 and CS 5599
Department of Computer Science
Idaho State University

ROAR

Outcomes

At the end of Today's Lecture you will be able to:

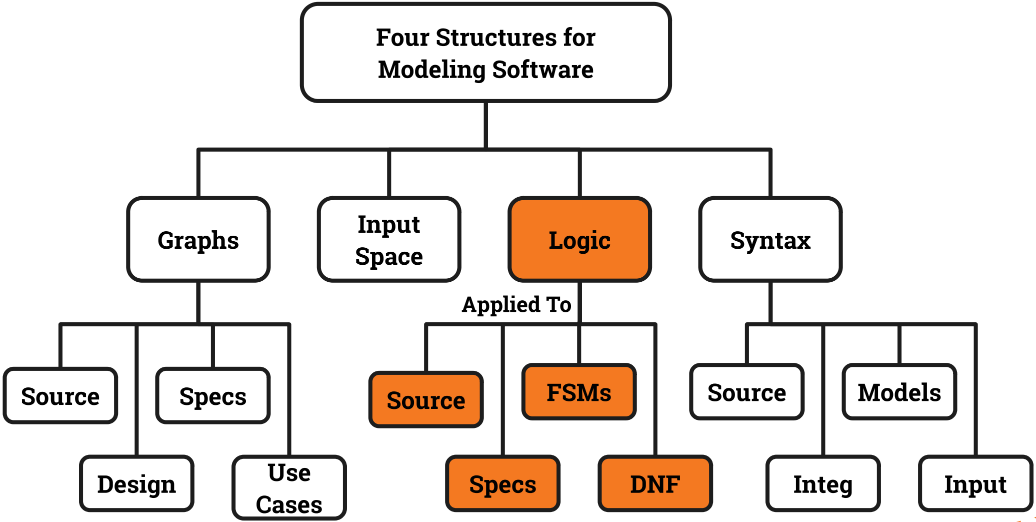
- Understand the basic concepts and notation for logic coverage
- Understand the different types of logic coverage criteria

Inspiration

“The trouble with programmers is that you can never tell what a programmer is doing until it’s too late.” – Seymour Cray

Idaho State University
Computer Science

Logic Coverage



Semantic Logic Criteria

- Logic expressions show up in many situations
- Covering logic expressions is required by the US Federal Aviation Administration for safety critical software
  - Used by other transportation industries
- Logical expressions can come from many sources
  - Decisions in programs
  - FSMs and statecharts
  - Requirements
- 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
  - $\neg$ – the negation operator
  - $\wedge$ – the and operator
  - $\vee$ – the or operator
  - $\Rightarrow$ – the implication operator
  - $\oplus$ – the exclusive or operator
  - $\Leftrightarrow$ – the equivalence operator

Example and Facts

- $(a < b) \vee f(z) \wedge D \wedge (m >= n * o)$ has four clauses:
  - $(a < b)$ - relational expression
  - $f(z)$ - Boolean-valued function
  - D - Boolean variable
  - $(m >= n * o)$ - relational expression
- 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

Example and Facts

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

Translating from English

- "I am interested in CS 4422 and CS 4458"
- course = cs4422 **OR** course = cs4458
  - Humans have trouble translating from English to Logic
- "If you leave before 6:30 AM, take Braddock to 495, if you leave after 7:00 AM, take Prosperity to 50, then 50 to 495"
- $(time < 6:30 \Rightarrow path = Braddock) \land (time > 7:00 \Rightarrow path = Prosperity)$
- Hmm ... this is incomplete!
- $(time < 6:30 \Rightarrow path = Braddock) \land (time \geq 6:30 \Rightarrow path = Prosperity)$

ROAR

Logic Coverage Criteria

- We use predicates in testing as follows:
  - Developing a model of the software as one or more predicates
  - Requiring tests to satisfy some combination of clauses
- Abbreviations:
  - P is the set of predicates
  - p is a single predicate in P
  - C is the set of clauses in P
  - C_p is the set of clauses in predicate p
  - c is a single clause in C

Idaho State University
Computer Science

Predicate and Clause Coverage

- The first (and simplest) two criteria require that each predicate and each clause be evaluated to both true and false

Predicate Coverage (PC)

For each p in P, TR contains two requirements: p evaluates to true, and p evaluates to false.

- When predicates come from conditions on edges, this is equivalent to edge coverage
- PC does not evaluate all the clauses, so ...

Clause Coverage (CC)

For each c in C, TR contains two requirements: c evaluates to true, and c evaluates to false.

ROAR

Predicate Coverage Example

$$((a < b) \vee D) \wedge (m \geq n * o)$$

Predicate = true

$$\begin{aligned} a = 5, b = 10, D = true, m = 1, n = 1, o = 1 \\ &= (5 < 10) \vee \top \wedge (1 \geq 1 * 1) \\ &= t \vee t \wedge \top \\ &= \top \end{aligned}$$

Predicate = false

$$\begin{aligned} a = 10, b = 5, D = false, m = 1, n = 1, o = 1 \\ &= (10 < 5) \vee false \wedge (1 \geq 1 * 1) \\ &= false \vee false \wedge \top \\ &= false \end{aligned}$$

Clause Coverage Example

True Cases

expression	a	b	D	m	n	o
$(a < b)$	5	10	–	–	–	–
D	–	–	true	–	–	–
$(m \geq n * o)$	–	–	–	1	1	1

False Cases

expression	a	b	D	m	n	o
$(a < b)$	10	5	–	–	–	–
D	–	–	false	–	–	–
$(m \geq n * o)$	–	–	–	1	2	2

These yield the following two tests:

1. a = 5, b = 10, D = true, m = 1, n = 1, o = 1
2. a = 10, b = 5, D = false, m = 1, n = 2, o = 2

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 <u>not</u> what we want!
- The simplest solution is to test **all combinations**...

Combinatorial Coverage

- CoC requires every possible combination
- Sometimes called Multiple Condition Coverage

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 true values

	$a < b$	D	$m \geq n * o$	$((a < b) \vee D) \wedge (m \geq n * o)$
1	T	T	T	T
2	T	T	F	F
3	T	F	T	T
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

Combinatorial Coverage

- This is simple, neat, clean, and comprehensive
- But quite **expensive**!
- 2^N tests, where N is the number of clauses
  - 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**" ...

Active Clauses

- Clause coverage has a **weakness**: The values do not always make a difference
- Consider the first test for **clause coverage**, which caused each clause to be true:
  - $(5 < 10) \lor true \lor (1 >= 1 * 1)$
- Only the first clause **counts**!
- To really test the results of a clause, the clause should be the **determining factor** in the value of the predicate

Definition (Determination)

A clause C_i in predicate p, called the major clause, **determines** p if and only if the values of the remaining minor clauses C_j are such that changing C_i changes the value of p.

- This is called **making the clause active**

Determining Predicates

$\underline{P = A \vee B}$

If $B = true$, p is always true.
so if $B = false$, A determines p
if $A = false$, B determines p.

$\underline{P = A \wedge B}$

If $B = false$, p is always false.
so if $B = true$, A determines p
if $A = true$, B determines p.

- **Goal**: Find tests for each clause when the clause determines the value of the predicate
- This is formalized in a **family of criteria** that have subtle, but very important, differences

Active Clause Coverage

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.

$\underline{p = a \vee b}$

1. **a = true**, b = false
2. **a = false**, b = false
3. a = false, **b = true**
4. ~~a = false, **b=false**~~

in 1 & 2 a is major, and in 3 & 4 b is major.

- This is a form of **MCDC**, which is required by the FAA for safety critical software
- **Ambiguity**: Do the minor clauses have to have the **same values** when the major clause is true and false?

ROAR

Resolving the Ambiguity

$p = a \vee (b \wedge c)$

Major clause: a

$a = true, b = false, c = true$

$a = false, b = false, c = false$

is $c = false$ allowed?

- This question caused **confusion** among testers for years
- Considering this carefully leads to **three** separate criteria:
  - Minor clauses **do not** need to be the same
  - Minor clauses **do** need to be the same
  - Minor clauses **force the predicate** to become both true and false

Are there any questions?