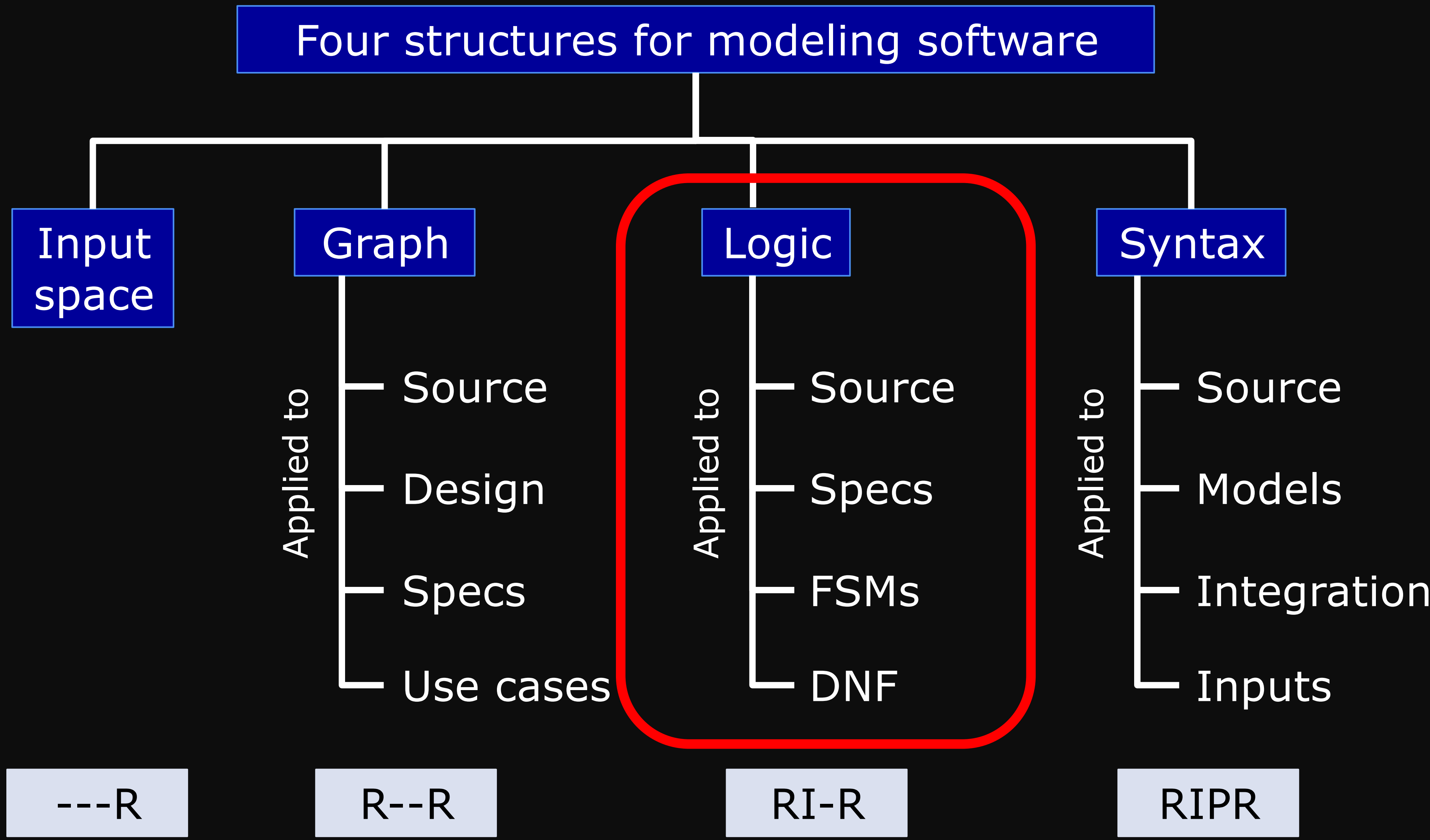
Logic Coverage

CS 3250
Software Testing

[Ammann and Offutt, "Introduction to Software Testing," Ch. 8]

Structures for Criteria-Based Testing



Overview

- Logic coverage ensures that tests not only reach certain locations, but the internal state is infected by trying multiple combinations of truth assignments to the expressions
- Covering logic expressions is required by the US Federal Aviation Administration for safety critical avionics software
- Logical expressions can come from many sources
  - Decisions in programs
  - FSMs and statecharts
  - Requirements
  - SQL queries
- Tests are intended to choose some subset of the total number of truth assignments to the expressions

Logic Predicates and Clauses

- Predicate: An expression that evaluates to a Boolean value
  - May contain
    - Boolean variable
    - Non-Boolean variables that contain >, <, ==, >=, <=, !=
    - Boolean function calls
  - Created by the logical operators

$\neg$	*negation* operator
$\wedge$	*and* operator
$\vee$	*or* operator
$\rightarrow$	*implication* operator
$\oplus$	*exclusive or* operator
$\leftrightarrow$	*equivalence* operator

- Clause: A predicate with no logical operators

Let's Refresh Our Memory

Row #	a	b	!a	a ∧ b	a ∨ b	a → b	a ⊕ b	a ↔ b
1	T	T						
2	T							
3		T						
4								

Blank indicates F

Example

$(a = b) \vee (C \wedge f(x))$

A predicate with logical operators

Three clauses

A relational expression (a = b)

A boolean variable C

A boolean-valued function p(x)

Logically equivalent

$((a = b) \vee C) \wedge ((a = b) \vee f(x))$

A predicate with logical operators

Three clauses

A relational expression (a = b)

A boolean variable C

A boolean-valued function p(x)

Note on Predicates

- Most predicates have few clauses
- Sources of predicates
  - Decisions in program source code

```
public boolean isSatisfactory()
{
   if ((good && fast) || (good && cheap) || (fast && cheap))
      return true;
   else
      return false;
}
```

$(good \wedge fast) \vee (good \wedge cheap) \vee (fast \wedge cheap)$

  - Guards in finite state machines

button2 = true (when gear = park)

$(gear = park) \wedge (button2 = true)$

  - Precondition in specifications

pre: stack not full AND object reference parameter not null

$\neg\, stackFull() \wedge (newObj \neq null)$

Note on Predicates

- Be careful when translating from English

"I am interested in CS6501 and CS4501"

⬇

Which one ?

(course = CS6501) AND (course = CS4501) ✗
(course = CS6501) OR (course = CS4501) ✓

From a study of 63 open source programs (>400,000 predicates), most predicates have few clauses [Ammann and Offutt]

- 88.5% have 1 clauses
- 9.5% have 2 clauses
- 1.35% have 3 clauses
- Only .65% have 4 or more

Try to keep the predicate simple and short.
How? Refactor it.

Short Circuit Evaluation

- Impacted by the order of operation
- Evaluate an expression or predicate until an outcome is known

$$\underline{((a = b) \vee C)}\ \boxed{\wedge}\ \underline{f(x)}$$

If f(x) is evaluated to T, we evaluate $(a = b) \vee C$ which can be T or F

If f(x) is evaluated to F, we stop. The outcome of the predicate is F

Short Circuit Evaluation

Boolean variable

Method call – something can change

```
if (isHungry && (time == f(time)))
{

}
```

If isHungry is evaluated to T, we evaluate (time == f(time)) which can be T or F

If isHungry is evaluated to F, we stop. The outcome of the predicate is F

Stop evaluating the predicate when we know the outcome

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*

Predicate Coverage (PC)

- For each *p* in *P*, TR contains two requirements:
  - *p* evaluates to true
  - *p* evaluates to false

"Decision coverage"

$$p = ((a = b) \vee C) \wedge f(x)$$

Need 2 test cases to satisfy PC

a		b	C	f(x)	p
3	T	3	T	T	**T**
4	F	3	F	T	**F**

- PC does not evaluate all the clauses, especially in the presence of short circuit evaluation

Activity #1 (PC)

p = a ∨ (b ∧ c)

Row#	a	b	c	p
1	T	T	T	
2	T	T		
3	T		T	
4	T			
5		T	T	
6		T		
7			T	
8				

List test requirements that satisfy predicate coverage (PC)

Blank indicates F

Activity #2 (PC)

p = a ∧ (b ∨ c)

Row#	a	b	c	p
1	T	T	T	
2	T	T		
3	T		T	
4	T			
5		T	T	
6		T		
7			T	
8				

List test requirements that satisfy predicate coverage (PC)

Blank indicates F

Clause Coverage (CC)

- For each *c* in *C*, TR contains two requirements:
  - *c* evaluates to true
  - *c* evaluates to false

"Condition coverage"

$p = ((a = b) \lor C) \land f(x)$

(a = b) evaluates to $\boxed{T}$, $\boxed{F}$
C evaluates to $\boxed{T}$, $\boxed{F}$
f(x) evaluates to $\boxed{T}$, $\boxed{F}$

a		b	C	f(x)	p
3	T	3	T	F	F
4	F	3	F	T	F

Need 2 test cases to satisfy CC

- CC does not always ensure PC
- The simplest solution is to test all combinations

Activity #1 (CC)

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

Row#	a	b	c	p
1	T	T	T	
2	T	T		
3	T		T	
4	T			
5		T	T	
6		T		
7			T	
8				

List test requirements that satisfy clause coverage (CC)

Blank indicates F

Activity #2 (CC)

p = a ∧ (b ∨ c)

Row#	a	b	c	p
1	T	T	T	
2	T	T		
3	T		T	
4	T			
5		T	T	
6		T		
7			T	
8				

List test requirements that satisfy clause coverage (CC)

Blank indicates F

Combinatorial Coverage (CoC)

- Evaluate all possible combination of truth values

"Multiple Condition coverage"

$p = ((a = b) \vee C) \wedge f(x)$

a		b	C	f(x)	p
3	T	3	T	T	T
3	T	3	T	F	F
3	T	3	F	T	T
3	T	3	F	F	F
4	F	3	T	T	T
4	F	3	T	F	F
4	F	3	F	T	F
4	F	3	F	F	F

Need 2^N test cases to satisfy CoC, where N = number of clauses

Note on CoC

- Coc is simple 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 that makes a big difference …
”active clause”

Revisit CoC Example

Which clause makes a big difference

$p = ((a = b) \vee C) \wedge f(x)$

a		b	C	f(x)	p
3	T	3	T	T	T
3	T	3	T	F	F
3	T	3	F	T	T
3	T	3	F	F	F
4	F	3	T	T	T
4	F	3	T	F	F
4	F	3	F	T	F
4	F	3	F	F	F

Active Clauses

- To really test the results of a clause, the clause should be the determining factor in the value of the predicate

- Determination
  - A clause c_i in predicate p, called the major clause, determines p if an only if the values of the remaining minor clauses c_j are such that changing c_i changes the value p
  - That is:
    - Major clause – the clause (being considered) that determine the predicate
    - Minor clause – all other clauses in the predicate

- This is considered to make the clause active

Determination

- Goal: Find tests for each clause when the clause determines the value of the predicate

- Determination: the conditions under which a clause solely determines the outcome of a predicate

  - Given a major clause c_i in a predicate p, c_i determines p if the minor clauses $c_j \neq c_i$ ($j \neq i$)

  - Major clause – "active clause" – controls the behavior

- Consider `p = a ∨ b`

  - If b = true, the value of a does not matter

  - If b = false, the value of a is the determining factor in the value of the predicate

Revisit Coc Example (again)

Which clause determines the predicate

$p = ((a = b) \vee C) \wedge f(x)$

a		b	C	f(x)	p
3	T	3	T	T	T
3	T	3	T	F	F
3	T	3	F	T	T
3	T	3	F	F	F
4	F	3	T	T	T
4	F	3	T	F	F
4	F	3	F	T	F
4	F	3	F	F	F

f(x) determines the predicate – but when ??

Deriving Determination Predicates, Using Mathematical Approach

Deriving (Mathematical Approach) Determination Predicates

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

$$
\begin{aligned}
p_a &= p_{a=true} \oplus p_{a=false} \\
&= (true \wedge (b \vee c)) \oplus (false \wedge (b \vee c)) \\
&= (b \vee c) \oplus false \\
&= b \vee c
\end{aligned}
$$

$$
\begin{aligned}
p_b &= p_{b=true} \oplus p_{b=false} \\
&= (a \wedge (true \vee c)) \oplus (a \wedge (false \vee c)) \\
&= (a \wedge true) \oplus (a \wedge c) \\
&= a \oplus (a \wedge c) \\
&= a \wedge \neg c
\end{aligned}
$$

$$
\begin{aligned}
p_c &= p_{c=true} \oplus p_{c=false} \\
&= (a \wedge (b \vee true)) \oplus (a \wedge (b \vee false)) \\
&= (a \wedge true) \oplus (a \wedge b) \\
&= a \oplus (a \wedge b) \\
&= a \wedge \neg b
\end{aligned}
$$

Deriving (Mathematical Approach) Determination Predicates

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

Major clause: a

$$
\begin{aligned}
p_a &= p_{a=true} \oplus p_{a=false} \\
&= (true \wedge (b \vee c)) \oplus (false \wedge (b \vee c)) \\
&= (b \vee c) \oplus false \\
&= b \vee c
\end{aligned}
$$

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T		
3	T		T	T	T		
4	T						
5		T	T		T		
6		T			T		
7			T		T		
8							

(Fill in a table to make it easy to read)

Blank indicates F

Deriving (Mathematical Approach) Determination Predicates

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

Major clause: b

$$
\begin{aligned}
p_b &= p_{b=true} \oplus p_{b=false} \\
&= (a \wedge (true \vee c)) \oplus (a \wedge (false \vee c)) \\
&= (a \wedge true) \oplus (a \wedge c) \\
&= a \oplus (a \wedge c) \\
&= a \wedge \neg c
\end{aligned}
$$

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T	T	
3	T		T	T	T		
4	T					T	
5		T	T		T		
6		T			T		
7			T		T		
8							

(Fill in a table to make it easy to read)

Blank indicates F

Deriving (Mathematical Approach) Determination Predicates

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

Major clause: c

$$
\begin{aligned}
p_c &= p_{c=true} \oplus p_{c=false} \\
&= (a \wedge (b \vee true)) \oplus (a \wedge (b \vee false)) \\
&= (a \wedge true) \oplus (a \wedge b) \\
&= a \oplus (a \wedge b) \\
&= a \wedge \neg b
\end{aligned}
$$

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T	T	
3	T		T	T	T		T
4	T					T	T
5		T	T		T		
6		T			T		
7			T		T		
8							

(Fill in a table to make it easy to read)

Blank indicates F

Deriving Determination Predicates, Using Tabular Approach

Identifying Determination Using Truth Table

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

Major clause: a

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T		
3	T		T	T	T		
4	T						
5		T	T		T		
6		T			T		
7			T		T		
8							

Blank indicates F

Identifying Determination Using Truth Table

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

Major clause: b

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T	T	
3	T		T	T	T		
4	T					T	
5		T	T		T		
6		T			T		
7			T		T		
8							

Blank indicates F

Identifying Determination Using Truth Table

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

Major clause: c

row	a	b	c	p	p_a	p_b	p_c
1	T	T	T	T	T		
2	T	T		T	T	T	
3	T		T	T	T		T
4	T					T	T
5		T	T		T		
6		T			T		
7			T		T		
8							

Blank indicates F

What's next?

- Use determination
- Apply logic coverage criteria to derive test requirements and design test cases
  - Active Clause Coverage (ACC)
  - General Active Clause Coverage (GACC)
  - Correlated Active Clause Coverage (CACC)
  - Restricted Active Clause Coverage (RACC)