CMPT 473
Software Quality Assurance

Logic Based Criteria

Nick Sumner
Material from Ammonn & Offutt

Logic Based Coverage Criteria

- Logical conditions are pervasive.

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, ...)
- Isn't branch coverage enough?

Logic Based Coverage Criteria

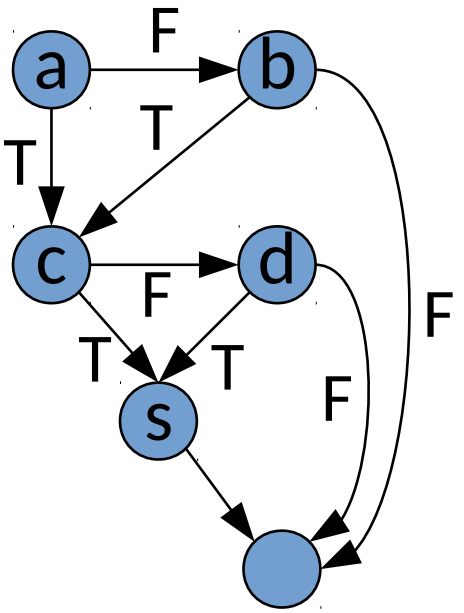
- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)
- Isn't branch coverage enough?

```
if (a || b) && (c || d):
  s
```

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, ...)
- Isn't branch coverage enough?

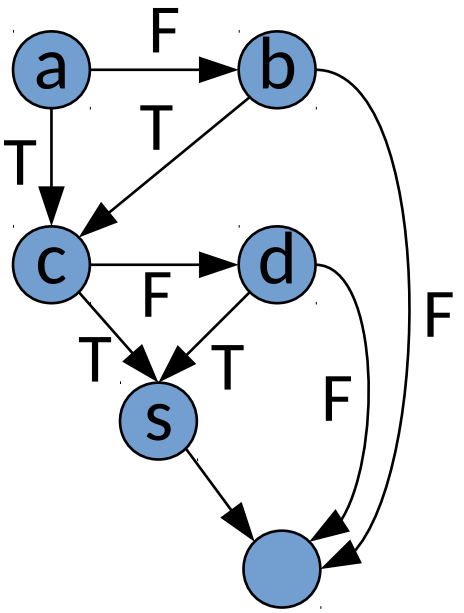
```
if (a || b) && (c || d):
  s
```



Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, ...)
- Isn't branch coverage enough?

```
if (a || b) && (c || d):
  s
```



What doesn't branch coverage test?

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)
- Isn't branch coverage enough?
- Why not just use path coverage?

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)
- Isn't branch coverage enough?
- Why not just use path coverage?

  1) Scalability

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)
- Isn't branch coverage enough?
- Why not just use path coverage?

1) Scalability
2)

```
if (a | b) & (c | d):
  s
```

Logic Based Coverage Criteria

- Logical conditions are pervasive.
- `if` statements are the most frequently fixed statements in bug fixes. [Pan, ESE 2008]
- Safety critical systems often involve many complex conditions. (avionics, medical, …)
- Isn't branch coverage enough?
- Why not just use path coverage?

1) Scalability
2)
```
if (a | b) & (c | d):
  s
```
3) Other languages (e.g., SQL)

Logic Based Coverage Criteria

- We want to reason about the *logical expressions* and how inputs affect their outcomes.

```
(a > 0) && foo(b) || c
```

Logic Based Coverage Criteria

- We want to reason about the *logical expressions* and how inputs affect their outcomes.

```
(a > 0) && foo(b) || c
```

- *Clauses* (in this context) are true or false and don't have logical operators.

Logic Based Coverage Criteria

- We want to reason about the *logical expressions* and how inputs affect their outcomes.

```
(a > 0) && foo(b) || c
```

- *Clauses* (in this context) are true or false and don't have logical operators.
- Logical coverage criteria identify a set values for clauses to test.

(a>0)	foo(b)	c	result
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Logic Based Coverage Criteria

- We want to reason about the *logical expressions* and how inputs affect their outcomes.

```
(a > 0) && foo(b) || c
```

- *Clauses* (in this context) are true or false and don't have logical operators.
- Logical coverage criteria identify a set values for clauses to test.

(a>0)	foo(b)	c	result
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression.

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression.
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression.
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.

```
if (a || b) && (c || d):
  s
```

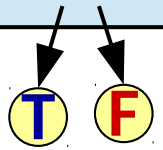
```
if (a | b) & (c | d):
  s
```

How does it do in these cases?

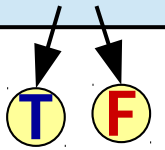
Predicate & Clause Coverage

- A *predicate* is simply a boolean expression.
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.

```
if (a || b) && (c || d):
  s
```




```
if (a | b) & (c | d):
  s
```



Predicate & Clause Coverage

- A *predicate* is simply a boolean expression.
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.

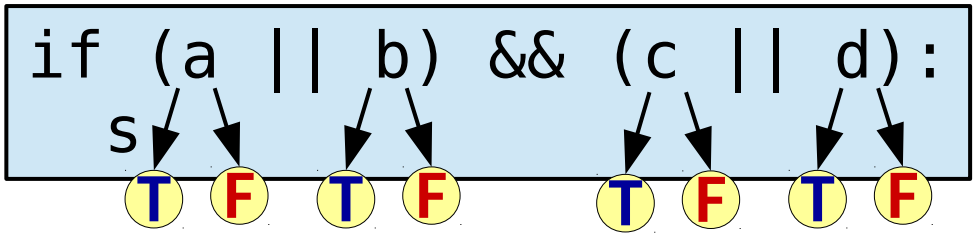
```
if (a || b) && (c || d):
  s
```

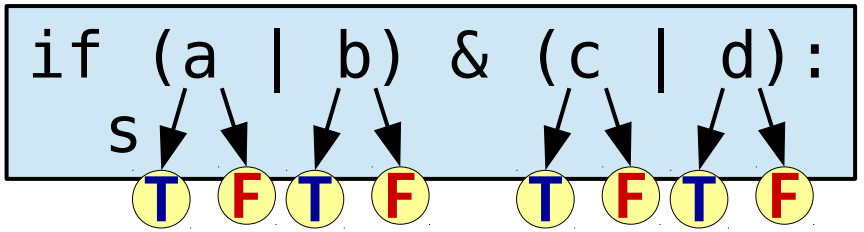
```
if (a | b) & (c | d):
  s
```

How does it do in these cases?

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.

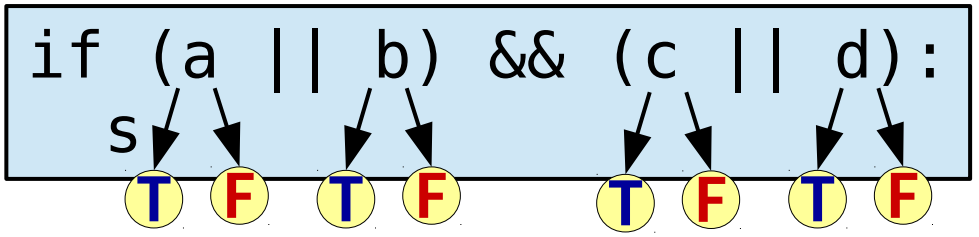


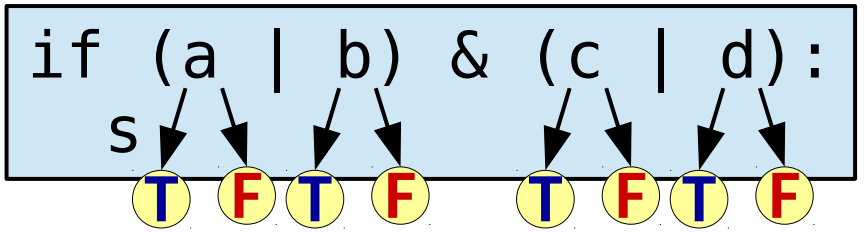


How many tests?

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.

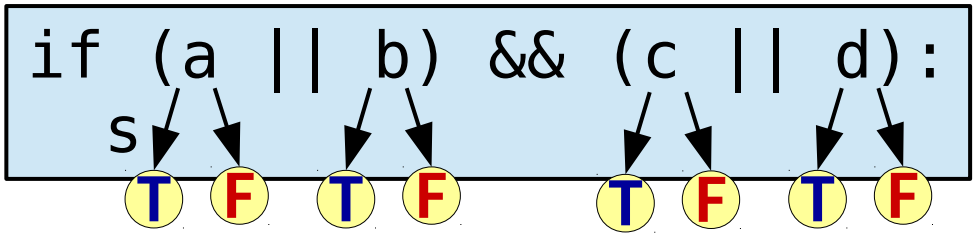


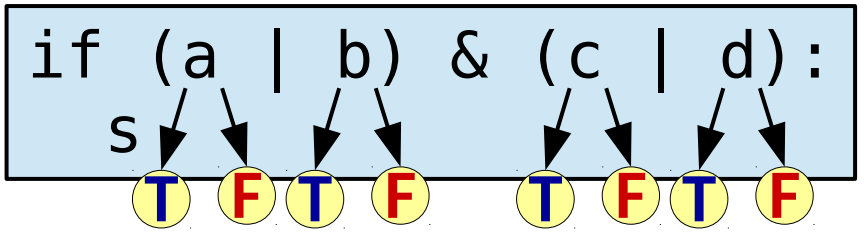


Minimum of **2** tests

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.



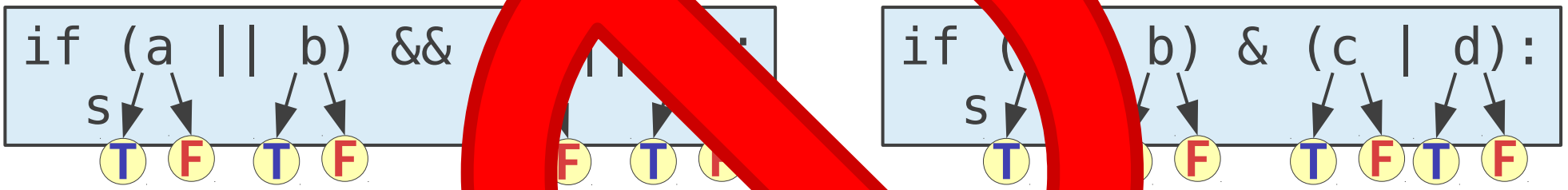


Minimum of **2** tests

a=true, b=true, c=false, d=false
a=false, b=false, c=true, d=true

Predicate & Clause Coverage

- A *predicate* is simply a boolean expression
- *Predicate Coverage* requires each predicate to be true in one test & be false in one test.
- *Clause Coverage* requires each clause to be true in one test & be false in one test.



Minimum tests

a=true, c=false
a=false, d=true

Combinatorial Coverage

- *Combinatorial/Multiple Condition Coverage* requires each possible combination of clauses to be tested. (Each row of the ***truth table***)

Combinatorial Coverage

- *Combinatorial/Multiple Condition Coverage* requires each possible combination of clauses to be tested. (Each row of the ***truth table***)

```
(a > 0) && foo(b) || c
```

Combinatorial Coverage

- *Combinatorial/Multiple Condition Coverage* requires each possible combination of clauses to be tested. (Each row of the ***truth table***)

```
(a > 0) && foo(b) || c
```

(a>0)	foo(b)	c	result
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

How many tests?

Defining a Better Goal

- Clause coverage takes each clause into account.

Defining a Better Goal

- Clause coverage takes each clause into account.
- Combinatorial coverage tests the impact of a combination.

Defining a Better Goal

- Clause coverage takes each clause into account.
- Combinatorial coverage tests the impact of a combination.
- Can we test for the impact of each clause?

Defining a Better Goal

- Clause coverage takes each clause into account.
- Combinatorial coverage tests the impact of a combination.
- Can we test for the impact of each clause?

```
if (a | b) & (c | d):
  s
```

How can we test the impact of a clause?

Defining a Better Goal

- Clause coverage takes each clause into account.
- Combinatorial coverage tests the impact of a combination.
- Can we test for the impact of each clause?

```
if (a | b) & (c | d):
  s
```

How can we test the impact of a clause?

The *relative* behavior when changing one clause matters.

Defining a Better Goal

- Clause coverage takes each clause into account.
- Combinatorial coverage tests the impact of a combination.
- Can we test for the impact of each clause?
  - This is the intuition behind MC/DC testing…

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*
  1) Each entry & exit is used

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*
  1) Each entry & exit is used
  2) Each decision/branch takes every possible outcome

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*

  1) Each entry & exit is used
  2) Each decision/branch takes every possible outcome
  3) Each clause takes every possible outcome

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*

  1) Each entry & exit is used
  2) Each decision/branch takes every possible outcome
  3) Each clause takes every possible outcome
  4) Each clause independently impacts the the outcome

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*
  1) Each entry & exit is used
  2) Each decision/branch takes every possible outcome
  3) Each clause takes every possible outcome
  4) Each clause independently impacts the the outcome
- Use in safety critical systems: avionics, spacecraft, …

Modified Condition/Decision Coverage

- *Modified Condition/Decision Coverage*
  1) Each entry & exit is used
  2) Each decision/branch takes every possible outcome
  3) Each clause takes every possible outcome
  4) Each clause independently impacts the the outcome
- Use in safety critical systems: avionics, spacecraft, …
- Not only ensures that clauses are tested,
  but that each has an impact

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate

$$\varphi(a,b,c) \neq \varphi(a,b,\neg c)$$

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate

$$\varphi(a,b,c) \neq \varphi(a,b,\neg c)$$

```
(a || b && c)
```

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate

$$\varphi(a,b,c) \neq \varphi(a,b,\neg c)$$

```
(a || b && c)
```

a=F	a=F
b=T	b=T
c=T	c=F
T	F

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate

$$\varphi(a,b,c) \neq \varphi(a,b,\neg c)$$

(a || b && c)	
a=F	a=F
b=T	b=T
c=T	c=F
T	F

This pair of tests shows the impact of c.

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate
- The basic steps come from & and |

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate
- The basic steps come from & and |

```
a & b
```

If `a=True`, `b` determines the outcome.

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate
- The basic steps come from & and |

`a & b`

If `a=True`, `b` determines the outcome.

`a | b`

If `a=False`, `b` determines the outcome.

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate
- The basic steps come from & and |

`a & b`
If `a=True`, `b` determines the outcome.

`a | b`
If `a=False`, `b` determines the outcome.

- By definition, solve $\varphi = \varphi_{c=true} \oplus \varphi_{c=false}$

How To Show a Clause Has Impact

- A clause *determines* the outcome of a predicate when changing only the value of that clause changes the outcome of the predicate
- The basic steps come from & and |

`a & b`
If `a=True`, `b` determines the outcome.

`a | b`
If `a=False`, `b` determines the outcome.

- By definition, solve $\varphi = \varphi_{c=true} \oplus \varphi_{c=false}$

Let's try:
a & b
a | b
a | (b & c)

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for **a**

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

  a has impact ↔ #T | (b & c) ≠ #F | (b & c)

by definition

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

  a has impact ↔ #T | (b & c) ≠ #F | (b & c)

  ↔ **#T** ≠

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

$$\text{a has impact} \leftrightarrow \#T \mid (b \,\&\, c) \neq \#F \mid (b \,\&\, c)$$
$$\leftrightarrow \#T \neq \mathbf{b \,\&\, c}$$

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

$$
\begin{aligned}
\text{a has impact} &\leftrightarrow \#T \mid (b \,\&\, c) \neq \#F \mid (b \,\&\, c) \\
&\leftrightarrow \#T \neq b \,\&\, c \\
&\leftrightarrow \#T = \neg b \mid \neg c
\end{aligned}
$$

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

$$
\begin{aligned}
\text{a has impact} &\leftrightarrow \#T \mid (b \,\&\, c) \neq \#F \mid (b \,\&\, c) \\
&\leftrightarrow \#T \neq b \,\&\, c \\
&\leftrightarrow \#T = \neg b \mid \neg c \\
&\leftrightarrow \textbf{b is false or c is false}
\end{aligned}
$$

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

a has impact ↔ #T | (b & c) ≠ #F | (b & c)

↔ #T ≠ b & c

↔ #T = ¬b | ¬c

↔ b is false or c is false

defines two different ways to test a

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

a has impact ↔ #T | (b & c) ≠ #F | (b & c)

↔ #T ≠ b & c

↔ #T = ¬b | ¬c

↔ b is false or c is false

defines two different ways to test a

Have b be #F

a=#T, b=#F, c=#T
a=#F, b=#F, c=#T

How To Show a Clause Has Impact

- Given **a | (b & c)**, generate tests for a

a has impact ↔ #T | (b & c) ≠ #F | (b & c)

↔ #T ≠ b & c

↔ #T = ¬b | ¬c

↔ b is false or c is false

defines two different ways to test a

Have b be #F	Have c be #F
a=#T, b=#F, c=#T	a=#T, b=#T, c=#F
a=#F, b=#F, c=#T	a=#F, b=#T, c=#F

Showing Impact

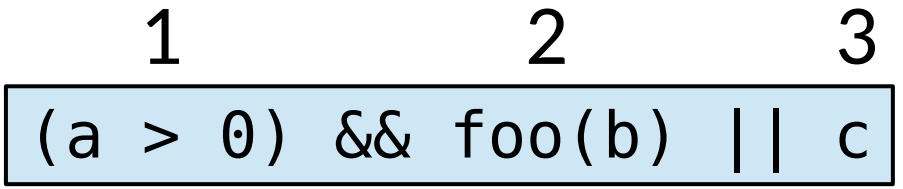
- What about (a & b) | (a & ¬b)?
  - Can you show the impact of a?
  - Can you show the impact of b?

Showing Impact

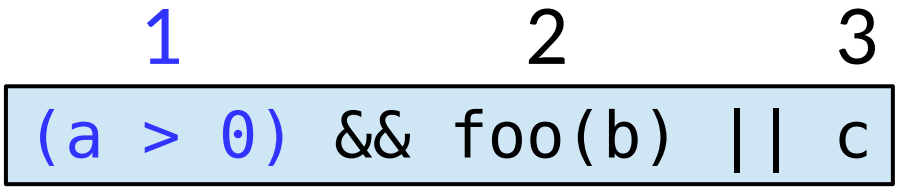
- What about (a & b) | (a & ¬b)?
  - Can you show the impact of a?
  - Can you show the impact of b?

MC/DC coverage also identifies redundancies that are likely bugs.

Generating a Test Suite with MC/DC

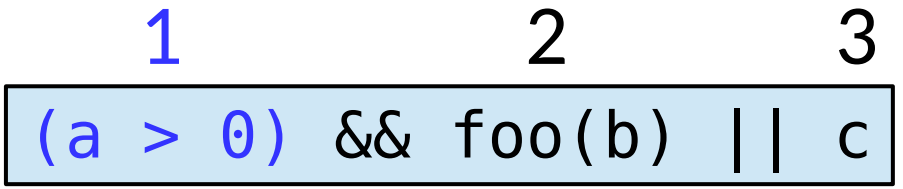


Generating a Test Suite with MC/DC



Select a first clause.

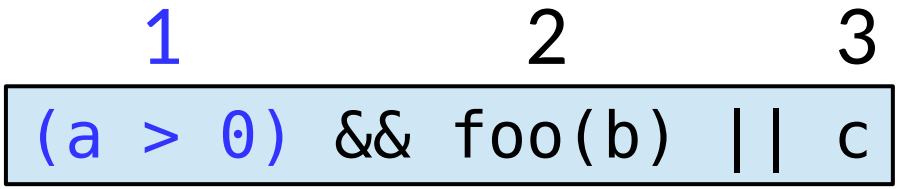
Generating a Test Suite with MC/DC



2=#T, 3=#F

Solve the constraints for other clauses.

Generating a Test Suite with MC/DC



2=#T, 3=#F

Include tests for the clause in the test suite.

Tests:

1=#T, 2=#T, 3=#F
1=#F, 2=#T, 3=#F

Generating a Test Suite with MC/DC

Consider the next clause.

Tests:

1=#T, 2=#T, 3=#F
1=#F, 2=#T, 3=#F

Generating a Test Suite with MC/DC

Try to add a test for it based on existing tests.

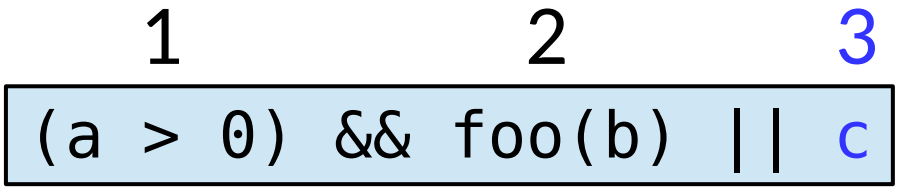
Tests:

1=#T, 2=#T, 3=#F

1=#F, 2=#T, 3=#F

1=#T, 2=#F, 3=#F

Generating a Test Suite with MC/DC



Repeat for the last clause.

Tests:

1=#T,	2=#T,	3=#F
1=#F,	2=#T,	3=#F
1=#T,	2=#F,	3=#F
1=#T,	2=#F,	3=#T

Generating a Test Suite with MC/DC?

- *BUT* NASA recommended *not* generating MC/DC coverage.

Generating a Test Suite with MC/DC?

- *BUT* NASA recommended *not* generating MC/DC coverage.
  - Use MC/DC as a means of evaluating test suites generated by other means

MC/DC Over Time

- Some historical ambiguities
  - Originally only required impact when changing clause
  - Changing other clauses at the same time was allowed!

MC/DC Over Time

- Some historical ambiguities
  - Originally only required impact when changing clause
  - Changing other clauses at the same time was allowed!
  - Why is this problematic?

MC/DC Over Time

- Some historical ambiguities
  - Originally only required impact when changing clause
  - Changing other clauses at the same time was allowed!
  - Why is this problematic?
- The form presented here is also known as *Restricted Active Clause Coverage*

Logic and MC/DC Testing

- Tests complex interactions in conditions.
- Required for avionics software.

Is it good? Bad?