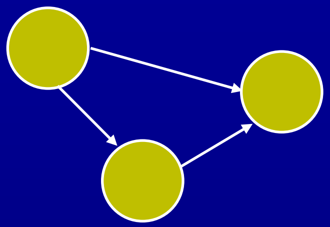
CS 575
Software Testing and Analysis

Test Coverage and Logic

Slightly modified versions of the textbook slides by Ammann & Offutt

Types of models

- **Graphs**
  - E.g., prime path coverage



- **Logical expressions**
  - E.g, clause coverage

(not X or not Y) and A and B

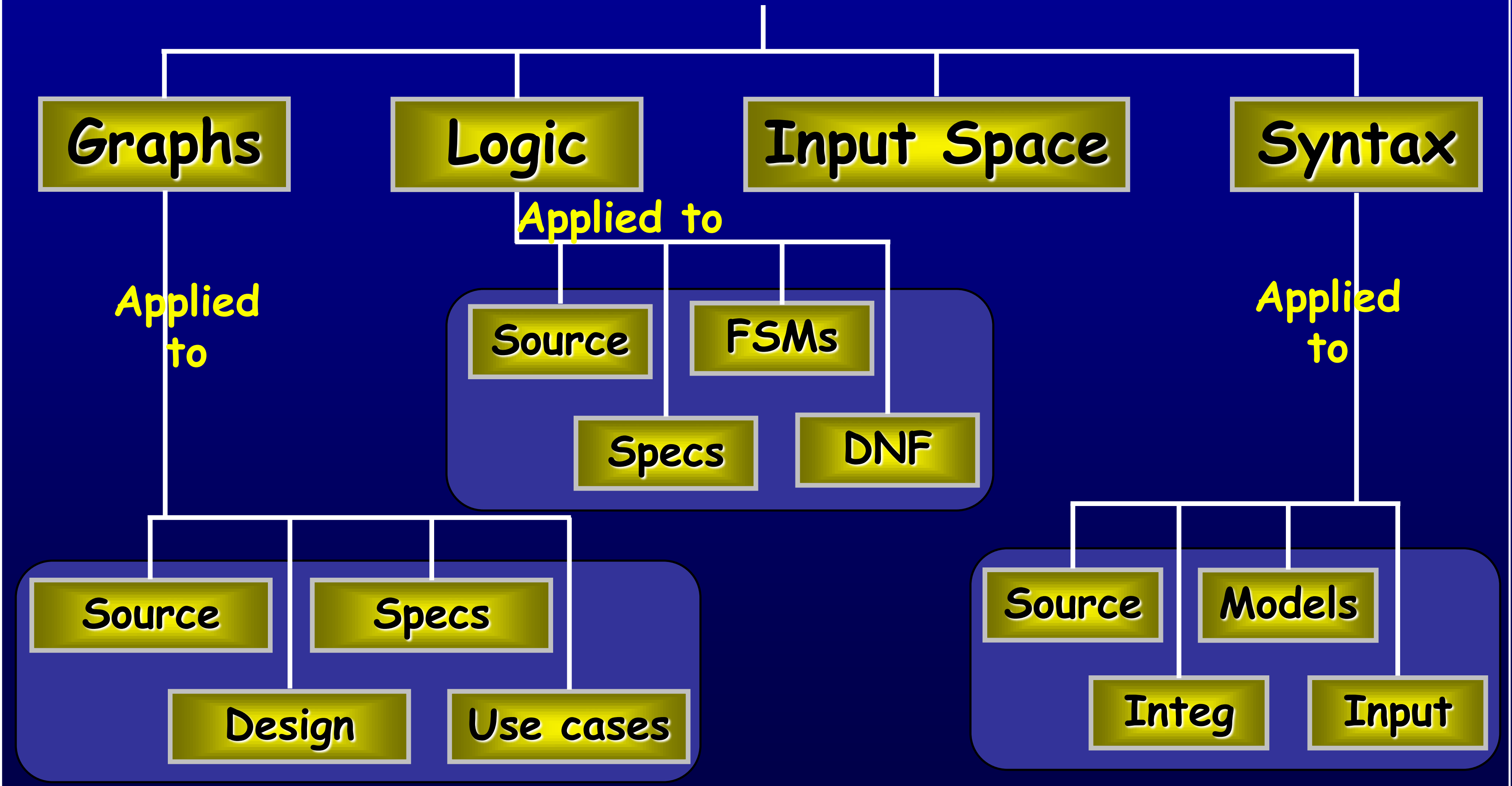
- **Input space**
  - E.g., pair-wise coverage

A: {0, 1, >1}
B: {600, 700, 800}
C: {swe, cs, isa, infs}

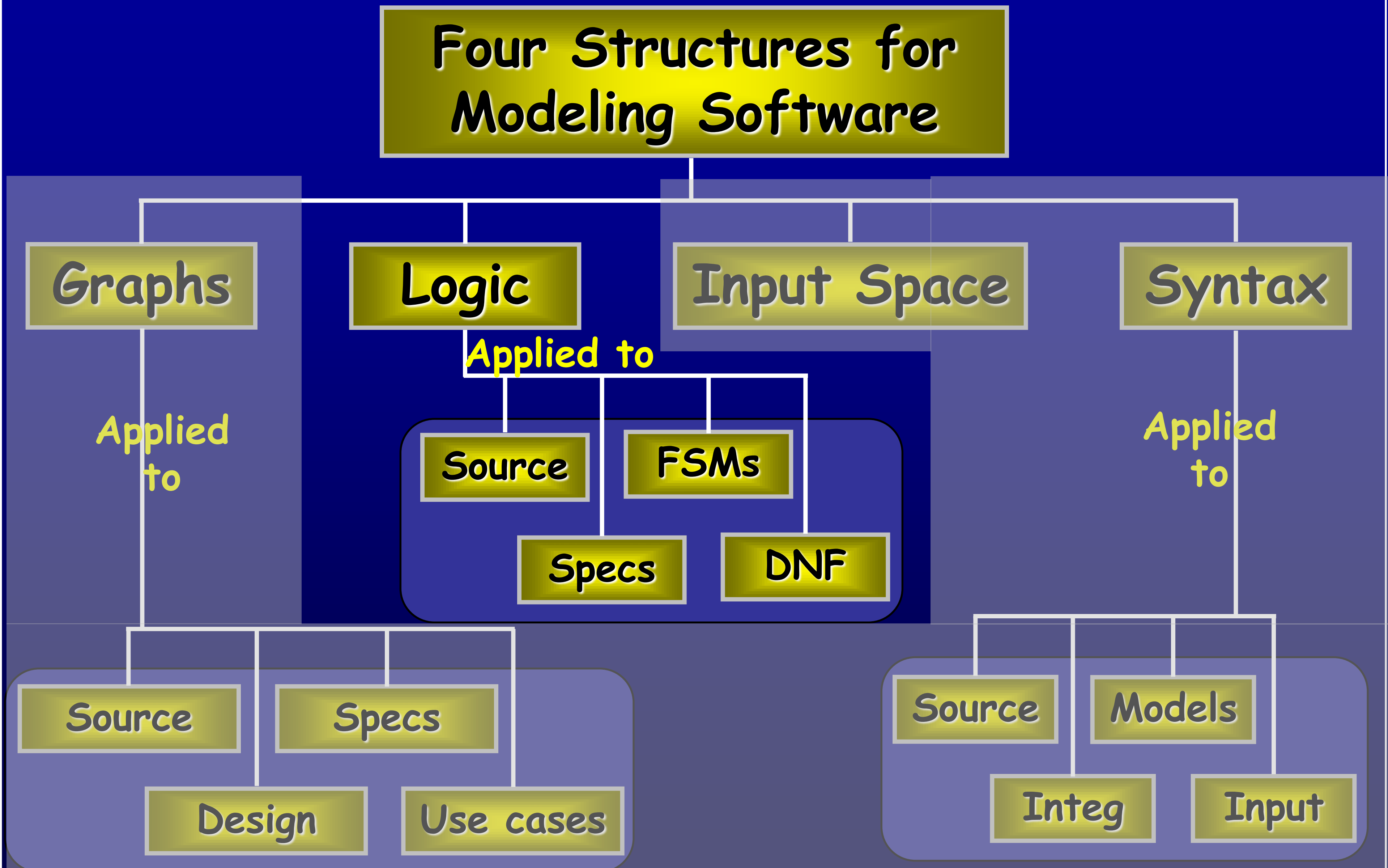
- **Syntax-based (grammars)**
  - E.g., production coverage

```
if (x > y)
   z = x - y;
else
  z = 2 * x;
```

Derivation of models



Ch. 3 : Logic Coverage



Covering Logic Expressions (3.1)

- **Logic expressions show up in many situations**
- **Covering logic expressions is required by the US Federal Aviation Administration for safety critical software**
- **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
- A *clause* is a predicate with no logical operators

Examples

- $(a < b) \lor f(z) \land D \land (m >= n*o)$
- Four clauses:
  - $(a < b)$ – relational expression
  - $f(z)$ – boolean-valued function
  - D – boolean variable
  - $(m >= n*o)$ – relational expression
- Sources of predicates
  - Decisions in programs
  - Guards in finite state machines
  - Decisions in UML activity graphs
  - Requirements, both formal and informal
  - SQL queries

Predicates Derived from Decisions in Programs

- Most predicates have only a few clauses
  - 88.5% have 1 clauses
  - 9.5% have 2 clauses
  - 1.35% have 3 clauses
  - Only .65% have 4 or more !

from a study of 63 open source programs, >400,000 predicates

Testing and Covering Predicates

- **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 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.

Predicate Coverage Example

$((a < b) \lor D) \land (m >= n*o)$

predicate coverage

Predicate = true

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

$= (5 < 10) \lor true \land (1 >= 1*1)$

$= true \lor true \land TRUE$

= true

Predicate = false

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

$= (10 < 5) \lor false \land (1 >= 1*1)$

$= false \lor false \land TRUE$

= false

Clause Coverage Example

$$((a < b) \lor D) \land (m >= n*o)$$

Clause coverage

(a < b) = true	(a < b) = false	D = true	D = false
a = 5, b = 10	a = 10, b = 5	D = true	D = false

m >= n*o = true	m >= n*o = false
m = 1, n = 1, o = 1	m = 1, n = 2, o = 2

true cases

false cases

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 Cp to evaluate to each possible combination of truth values.

	a < b	D	m >= n*o	((a < b) $\vee$ D) $\wedge$ (m >= 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 Clause Coverage

<u>Active Clause Coverage (ACC)</u> : For each *p* in *P* and each major clause *ci* in *Cp*, choose minor clauses *cj*, *j != i*, so that *ci* determines *p*. TR has two requirements for each *ci* : *ci* evaluates to true and *ci* evaluates to false.

<u>p = a ∨ b</u>

1) a = true, b = false
2) a = false, b = false
3) a = false, b = true
4) ~~a = false, b = false~~

a is major clause

b is major clause

Duplicate

Active Clause Coverage

- This is a form of **MCDC**, which is required by the FAA for safety critical software
- ***N+1*** tests are sufficient for coverage, where ***N*** is the number of clauses
- **Ambiguity** : Do the minor clauses have to have the **same values** when the major clause is true and false?

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 this allowed ?

- **Separate criteria defined to avoid ambiguity**
  - Minor clauses do not need to be the same
    - General Active Clause Coverage (GACC)
  - Minor clauses do need to be the same
    - Restricted Active Clause Coverage (RACC)

Exercise

```
for(n = 0;
   n < max_size && (c = getc( yyin )) != EOF && c != '\n';
   n++)
          buf[n] = (char) c;
```

- Devise a set of test cases that satisfy the GACC and RACC criteria with respect to the loop condition
- Hint: There exist 3 clauses. So, 4 test cases should be sufficient to satisfy GACC and RACC

Adopted from the textbook slides by Pezze & Young

Exercise

```
for(n = 0;
   n < max_size && (c = getc( yyin )) != EOF && c != '\n';
   n++)
        buf[n] = (char) c;
```

- Entries marked with "-" normally can be either of true or false to satisfy GACC. However, we should set them to true for satisfying RACC.

Test Case	n < max_size	(c = getc(yyin)) != EOF	c != '\n'	Outcome
(1)	false	-	-	false
(2)	true	false	-	false
(3)	true	true	false	false
(4)	true	true	true	true

Test Case	n < max_size	(c = getc(yyin)) != EOF	c != '\n'	Outcome
(1)	false	true	true	false
(2)	true	false	true	false
(3)	true	true	false	false
(4)	true	true	true	true

Correlated Active Clause Coverage

<u>Correlated Active Clause Coverage (CACC)</u> : For each *p* in *P* and each major clause c_i in *Cp*, choose minor clauses c_j, *j != 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. The values chosen for the minor clauses c_j must <u>cause *p* to be</u> true for one value of the major clause c_i and false for the other, that is, it is required that $p(c_i = true) \mathrel{!=} p(c_i = false)$.

- **A more recent interpretation**
- **Implicitly allows minor clauses to have different values**
- **Explicitly satisfies (subsumes) predicate coverage**

CACC and RACC

	a	b	c	a ∧ (b ∨ c)
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

	a	b	c	a ∧ (b ∨ c)
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

major clause

P_a : b=true or c = true

CACC can be satisfied by choosing any of rows 1, 2, 3 AND any of rows 5, 6, 7 – a total of nine pairs

RACC can only be satisfied by row pairs (1, 5), (2, 6), or (3, 7)

Only three pairs

Exercise by Paul Amman

- **Devise a set of test cases that satisfy the CACC and RACC criteria with respect to the predicate**

	a	b	c	P	P_a
1	T	T	T		T
2	T	T		T	T
3	T		T		T
4	T			T	
5		T	T	T	T
6		T			T
7			T	T	T
8				T	

$P = \bar{a}\bar{b} + a\bar{c} + \bar{a}c$

All pairs of p in Pa

- **Pa = T, a = T: 1, 2, 3**
- **Pa = T, a = F: 5, 6 7**

- **All Pairs:**

(1,5) (2,5) (3,5)
(1,6) (2,6) (3,6)
(1,7) (2,7) (3,7)

- **Satisfies GACC**

	a	b	c	P	P_a
1	T	T	T		T
2	T	T		T	T
3	T		T		T
4	T			T	
5		T	T	T	T
6		T			T
7			T	T	T
8				T	

$P = \bar{a}b + a\bar{c} + \bar{a}c$

Pairs that satisfy CACC

- **Pa = T, a = T: 1, 2, 3**
- **Pa = T, a = F: 5, 6 7**

- **All Pairs where P changes:**

(1,5) ~~(2,5)~~ (3,5)
~~(1,6)~~ (2,6) ~~(3,6)~~
(1,7) ~~(2,7)~~ (3,7)

	a	b	c	P	Pa
1	T	T	T		T
2	T	T		T	T
3	T		T		T
4	T			T	
5		T	T	T	T
6		T			T
7			T	T	T
8				T	

$P = \bar{a}\bar{b} + a\bar{c} + \bar{a}c$

Pairs that satisfy RACC

- **Pa = T, a = T: 1, 2, 3**
- **Pa = T, a = F: 5, 6 7**

- **All Pairs where minor clauses remain the same:**

(1,5) ~~(2,5)~~ ~~(3,5)~~
~~(1,6)~~ (2,6) ~~(3,6)~~
~~(1,7)~~ ~~(2,7)~~ (3,7)

	a	b	c	P	Pa
1	T	T	T		T
2	T	T		T	T
3	T		T		T
4	T			T	
5		T	T	T	T
6		T			T
7			T	T	T
8				T	

$P = \bar{a}\bar{b} + a\bar{c} + \bar{a}c$