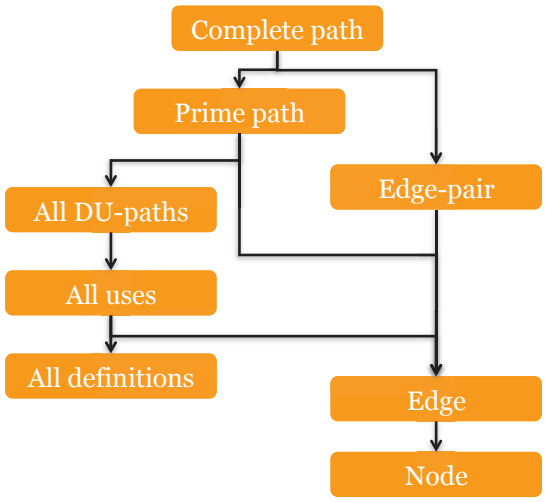
Graph-Based Criteria: Subsumption Hierarchy

- **A -> B** indicates that **A** subsumes **B**
- **Question 2:** The subsumption hierarchy is defined for fully satisfied criteria (i.e., 100% coverage). Does it hold for any degree of coverage? For example, does 80% edge coverage imply at least 80% node coverage?



Logic-Based Adequacy Criteria

MÄLARDALEN UNIVERSITY SWEDEN

Logic-Based Adequacy Criteria Intro.

- Consider the following statement:

  if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Logic-Based Adequacy Criteria Intro.

- Consider the following statement:

  if ((a>b AND c==10) OR (d<4 AND (c!=a)))

This is a *predicate,* i.e., an expression that evaluates to a boolean value

Logic-Based Adequacy Criteria Intro.

- Consider the following statement:

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

A predicate is made up from one or more *clauses,* i.e., predicates that contain no boolean operators (AND, OR, etc...)

Logic-Based Adequacy Criteria Intro.

- Consider the following statement:

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

- Also, remember, logical expressions are a fundamental part of programming that determine the flow of control in any type of software
- Using predicates and clauses, a whole new family of structural adequacy criteria can be defined

Logic-Based Adequacy Criteria

Definitions and examples

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Predicate coverage

The set of coverage items contains two requirements for each p in P: p evaluates to true, and p evaluates to false.

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Predicate coverage

The set of coverage items contains two requirements for each *p* in *P*: *p* evaluates to true, and *p* evaluates to false.

P: The set of all predicates in the software under test.

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Predicate coverage

The set of coverage items contains two requirements for each *p* in *P*: *p* evaluates to true, and *p* evaluates to false.

Let's call this *p*

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Predicate coverage

The set of coverage items contains two requirements for each *p* in *P*: *p* evaluates to true, and *p* evaluates to false.

Set of coverage items:

{p evaluates to *true*, p evaluates to *false*}

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Clause coverage

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

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Clause coverage
The set of coverage items contains two requirements for each *c* in *C*: *c* evaluates to true, and *c* evaluates to false.

C: The set of all clauses in P.

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Clause coverage
The set of coverage items contains two requirements for each *c* in *C*: *c* evaluates to true, and *c* evaluates to false.

a>b	c==10	d<4	c!=a
TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Combinatorial coverage
For each p in P, the set of coverage items contains all possible combinations of truth values of the clauses of p.

a>b	c==10	d<4	c!=a
TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	FALSE
TRUE	TRUE	FALSE	TRUE
TRUE	TRUE	FALSE	FALSE
TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE
TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Problems

In many cases, predicate and clause coverage are too ”simple”, and combinatorial coverage is way too expensive.

a>b	c==10	d<4	c!=a

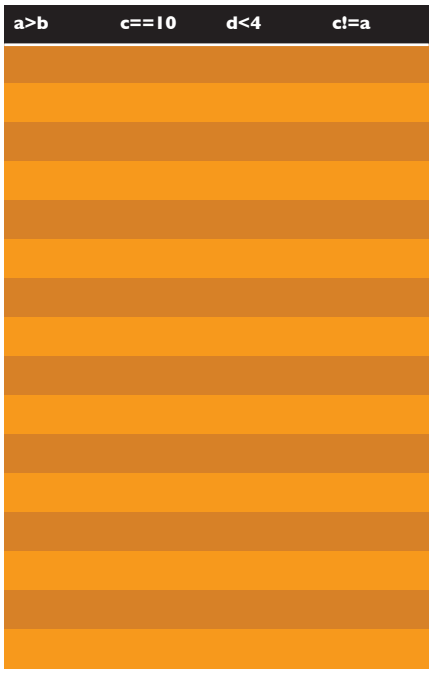
Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Solution: Active Clause Criteria

Basic idea: Make each individual clause affect the outcome of the predicate

Active clause criteria are also known as MC/DC, and are required, e.g., in testing of aviation software.



Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

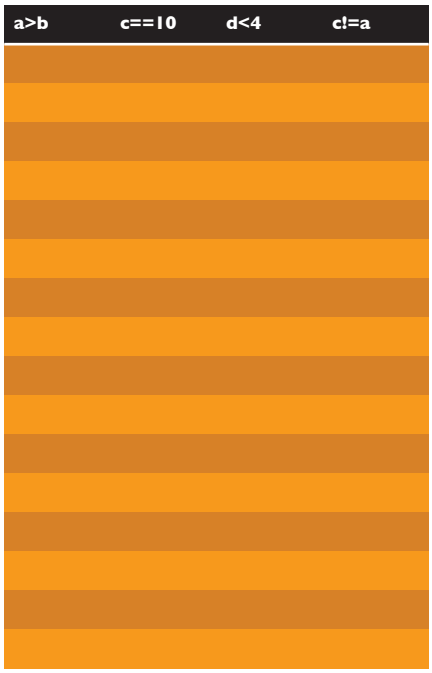
Solution: Active Clause Criteria

Basic idea: Make each individual clause affect the outcome of the predicate

Active clause criteria are also known as MC/DC, and are required, e.g., in testin aviation software.

There is an ambiguity here

The definition of MC/DC can actually be interpreted in three different ways

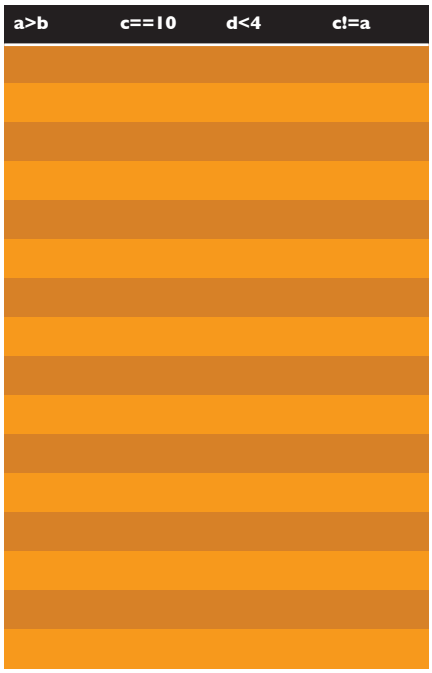


Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

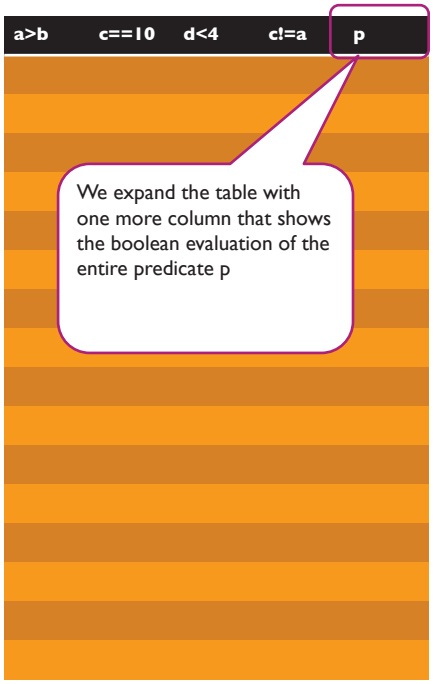


Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).



Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p

Then we initially consider a>b the **major clause** in p. We want to find values of the other clauses such that a change from TRUE to FALSE in the major clause would change the value of the entire predicate

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TR	FALSE	FALSE	FALSE

Then the process is repeated for all clauses in p (all of them get to act as major clause)

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE

Logic-Based Adequacy Criteria

if ((a>b AND c==10) OR (d<4 AND (c!=a)))

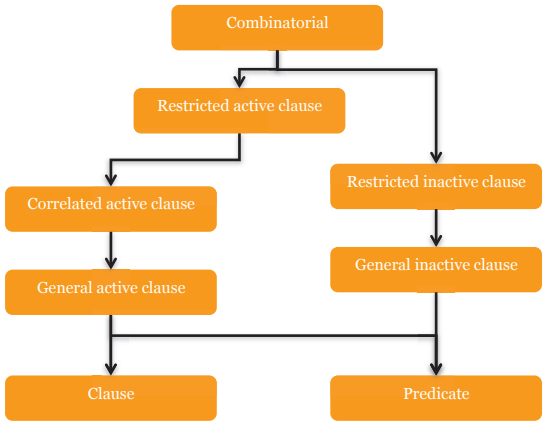
Restricted Active Clause Coverage: Example

For each predicate p:
For each clause c_i in p:
1.Let c_i be the major clause of p.
2.Choose values of minor clauses such that c_i ***determines*** p.
3.The set of coverage items includes two requirements: c_i evaluates to true and c_i evaluates to false (while all minor clauses remain unchanged).

a>b	c==10	d<4	c!=a	p
TRUE	TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE

Logic-Based Criteria: Subsumption Hierarchy

- **A -> B** indicates that **A** subsumes **B**



Important Considerations for Structural Testing

Exercise 1: Coverage

```
int function(int x, int y, int z)
{
   if (x<y && x<z)
      return x;
   else if (y<x && y<z)
            return y;
         else
            return z;
}
```

Question:

What does this function do?

Exercise 2: Coverage of test suite?

```
int function(int x, int y, int z)
{
   if (x<y && x<z)
      return x;
   else if (y<x && y<z)
         return y;
      else
         return z;
}
```

Test suite:

x=1, y=2, z=3, exp 1, output 1
x=2, y=1, z=3, exp 1, output 1
x=2, y=1, z=0, exp 0, output 0

Questions:

*2a) What is the **statement** coverage of the test suite?*
*2b) What is the **predicate** coverage of the test suite?*

Exercise 3: With one added test case?

```
int function(int x, int y, int z)
{
   if (x<y && x<z)
      return x;
   else if (y<x && y<z)
         return y;
      else
         return z;
}
```

Test suite:

x=1, y=2, z=3, exp 1, output 1
x=2, y=1, z=3, exp 1, output 1
x=2, y=1, z=0, exp 0, output 0
x=1, y=1, z=3, exp 1, output ?

Questions:

*3a) What is the **statement** coverage of the test suite?*
*3b) What is the **output value** of the added test case?*

Combining Specification-Based with Implementation-Based Test Design

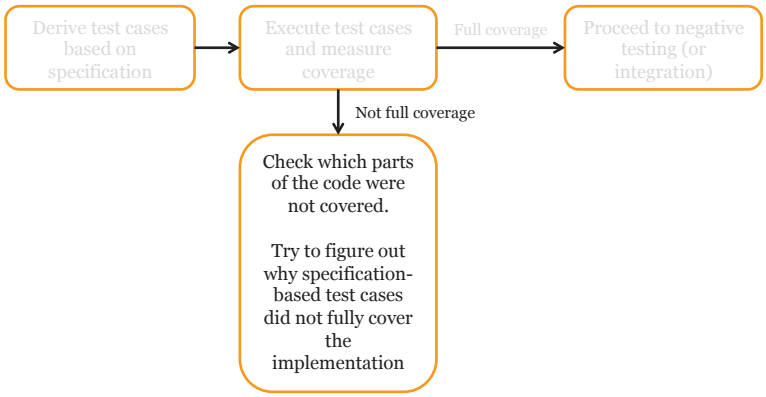
Derive test cases based on specification

Combining Specification-Based with Implementation-Based Test Design

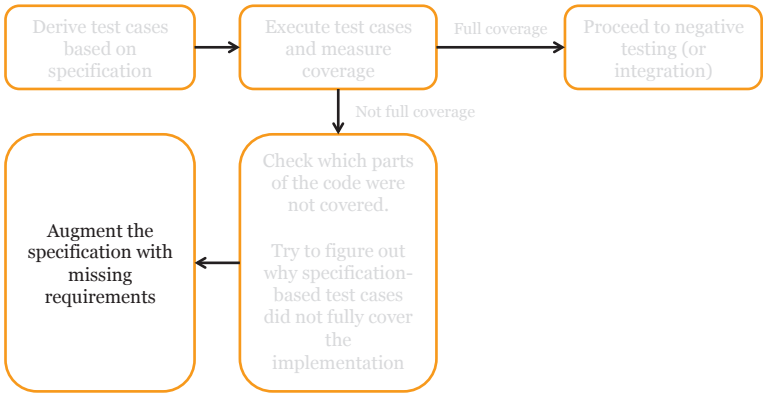
Derive test cases based on specification → Execute test cases and measure coverage

Combining Specification-Based with Implementation-Based Test Design

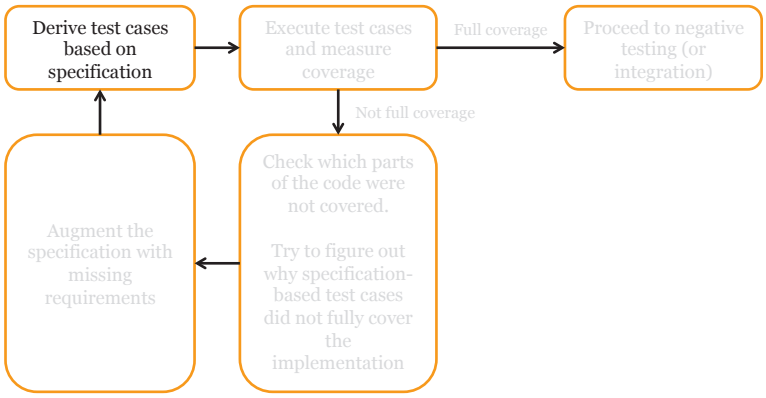
Combining Specification-Based with Implementation-Based Test Design



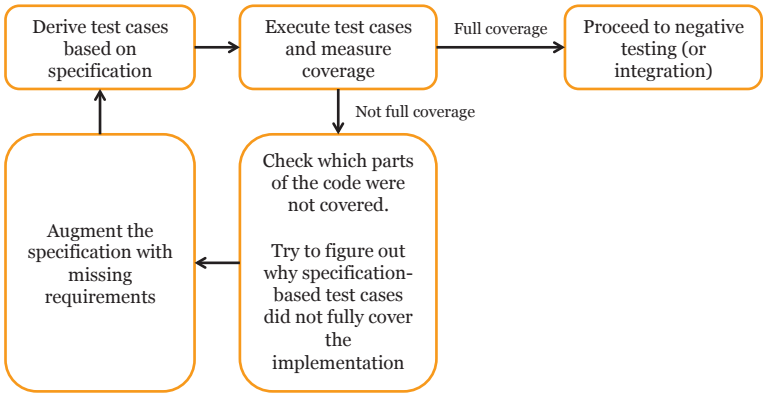
Combining Specification-Based with Implementation-Based Test Design



Combining Specification-Based with Implementation-Based Test Design



Combining Specification-Based with Implementation-Based Test Design



Applying Criteria in Practice

- Measuring coverage of the implementation requires ***instrumentation*** of the source code
- For example, when measuring statement coverage it is required that the statements exercised during testing are monitored and kept track of
- This requires tool support

Applying Criteria in Practice (cont.)

- Measuring coverage of the implementation requires ***instrumentation*** of the source code
- For example, when measuring statement coverage it is required that the statements exercised during testing are monitored and kept track of
- This requires tool support

Again note that different coverage tools may interpret and measure criteria differently (even though the names are identical).

Feasibility

- Some coverage items may not be feasible
  - Dead code
  - Incompatible boolean requirements
- It should be kept in mind that infeasibility of coverage items is a fundamental part of testing,
  - I.e., not always possible to reach 100% coverage (of any type)
  - It is recommended that coverage items not reached during testing are identified and analysed
- However, in the general case, determining whether a coverage item is feasible or not is ***undecidable***

Test Case Generation for Structural Testing

Typical Question: How do we select x and y such that a > b?

In some cases, this can be resolved manually. Often, it requires tool support from constraint solvers. Other times, no solution can be found.

Automated test case generation is a highly active research field.

Additional problem: What is the expected output?

```
int foo(int x, int y)
{
    ...
    if (a>b)
    {
     …
    }
     else
    {
     …
    }
}
```

Summary

- Structural Testing Fundamentals
- Adequacy Criteria and Coverage
  - Graph-based Adequacy Criteria
  - Logic-based Adequacy Criteria
- Important Considerations for Structural Testing