Software Testing

- Overview and Terminology
- Inspections and Walkthroughs
- Control Flow Coverage Criteria
- Dataflow Coverage Criteria
- Propagating Path Conditions
- Equivalence Partition

What is Testing ?

“The process of executing the checked program under certain preconditions and parameters in order to find errors”

Goal: to reveal errors - not to prove they don’t exist

Sequential programs

Different Testing Approaches

- Unit (module) testing
- Integration testing
- System testing
- Acceptance testing
- Regression testing
- Stress testing

Black box testing: testing a system using only knowledge of its external interface - no internal structure.

White (transparent) box testing: knowledge of the internal structure of the system is used in testing.

Execution path: a sequence of instructions in the code.

Code coverage analysis: a way to assess the "quality and quantity" of testing.

Test case: preconditions and parameters for running the program, and the expected output & other criteria for passing the test.

Test suite: a set of test cases.

Test environment: allows executing the test cases and checking the result.

Inspections and Walkthroughs

- Manual testing methods

Code inspection: manually checking the code, possibly agains a list of potential errors.

Code walkthrough: “Simulating” some test cases.

Control Flow Coverage Criteria

Statement coverage: each statement of the program appears in at least one test case.

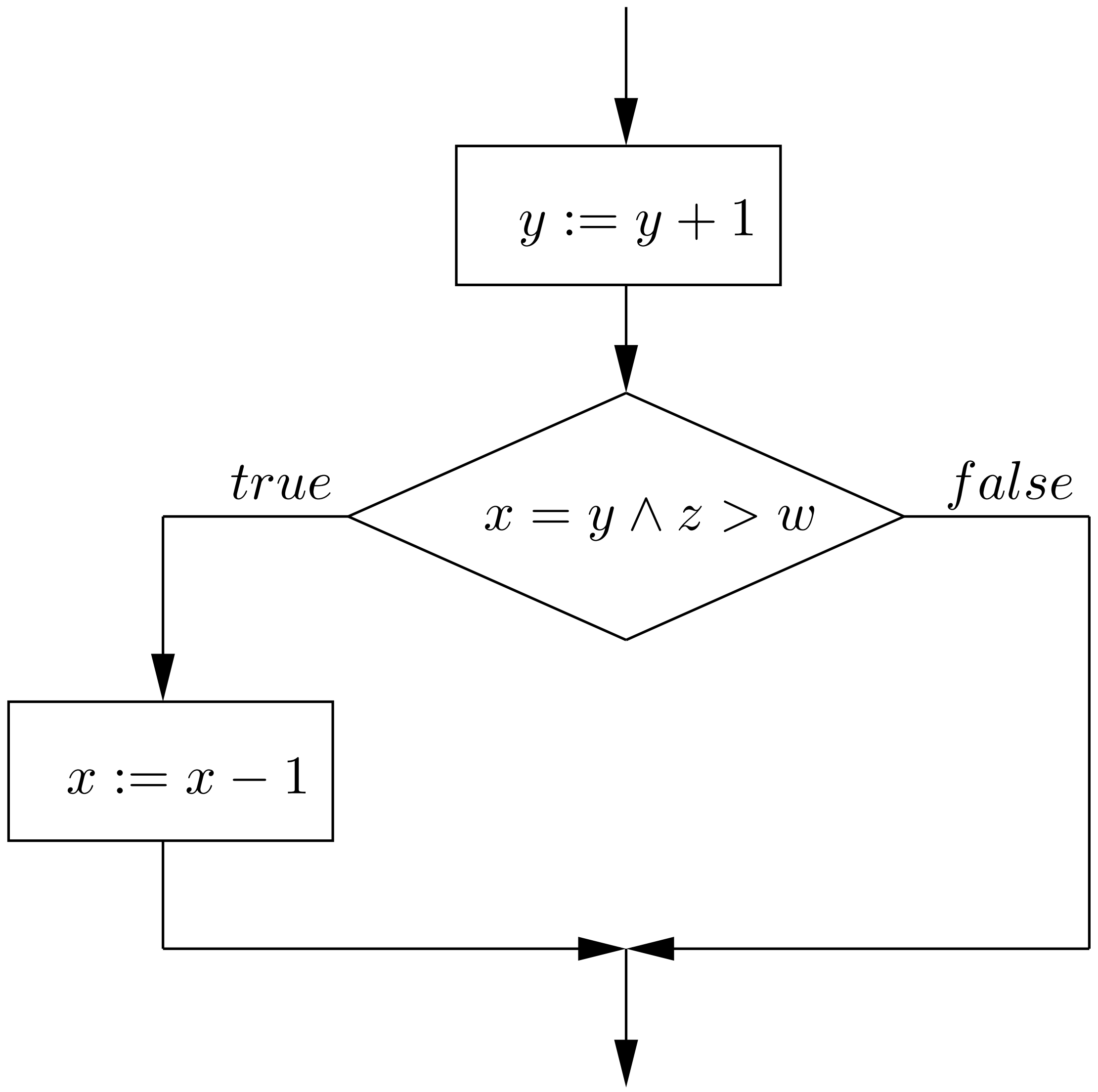
Edge coverage: each edge of the flowchart appears in some test case.

Condition coverage: each condition appears in some test case where it evaluates to *true*, and in another test case, where it is interpreted as *false*.

Edge/condition coverage: requires both the edges and the conditions to be covered.

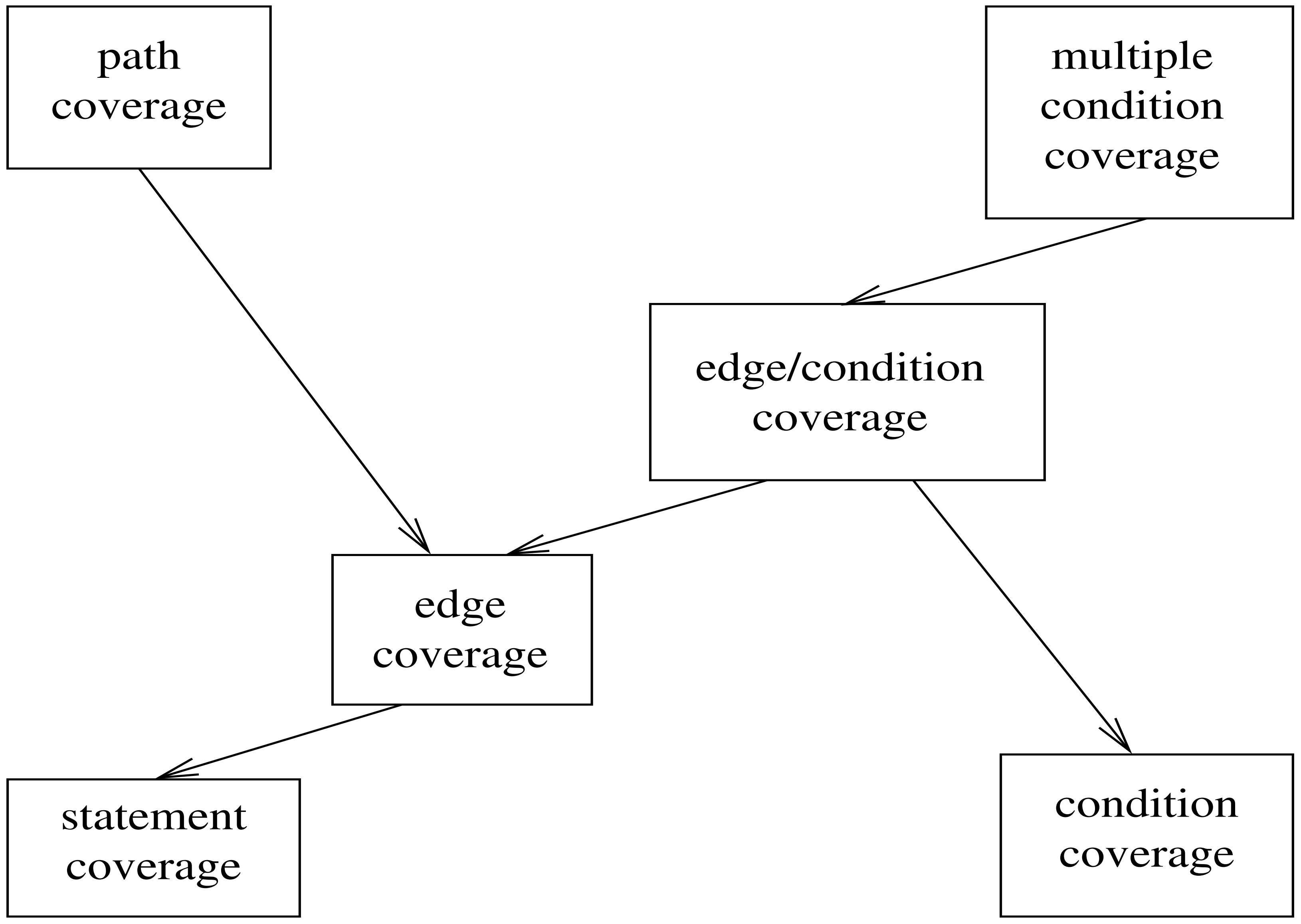
Multiple condition coverage: each boolean combination that may appear in any decision predicate must appear in some test case.

Path coverage: every executable path has to be covered by a test case.


$y := y + 1$
$x = y \wedge z > w$
$true$
$false$
$x := x - 1$

Limitations of Control Flow Coverage Criteria

- Not comprehensive
- Biased towards the way the code was written
- Difficult to assess the effectiveness of different coverage criteria


path coverage
multiple condition coverage
edge/condition coverage
edge coverage
statement coverage
condition coverage

Dataflow Coverage

Test case selection is based on paths between assignments to, and uses of variables.

def(x) the nodes where some value is assigned to x.

p-use(x) the nodes where x is used in a predicate.

c-use(x) the nodes where x is used in an expression other than a predicate.

def-clear(x) the paths that include only nodes not in *def*(x).

dpu(s, x) nodes s' such that there is a *def-clear*(x)path from s to s' (except the first node), and s' is in *p-use*(x).

dcu(s, x) nodes s' such that there is a *def-clear*(x)path from s to s', and s' is in *c-use*(x).

Dataflow Coverage Criteria

For each program variable x, and for each statement in *def* (x), include at least the following *def-clear* (x)paths:

all-defs a path to some node in $dpu(s, x)$or in $dcu(s, x)$.

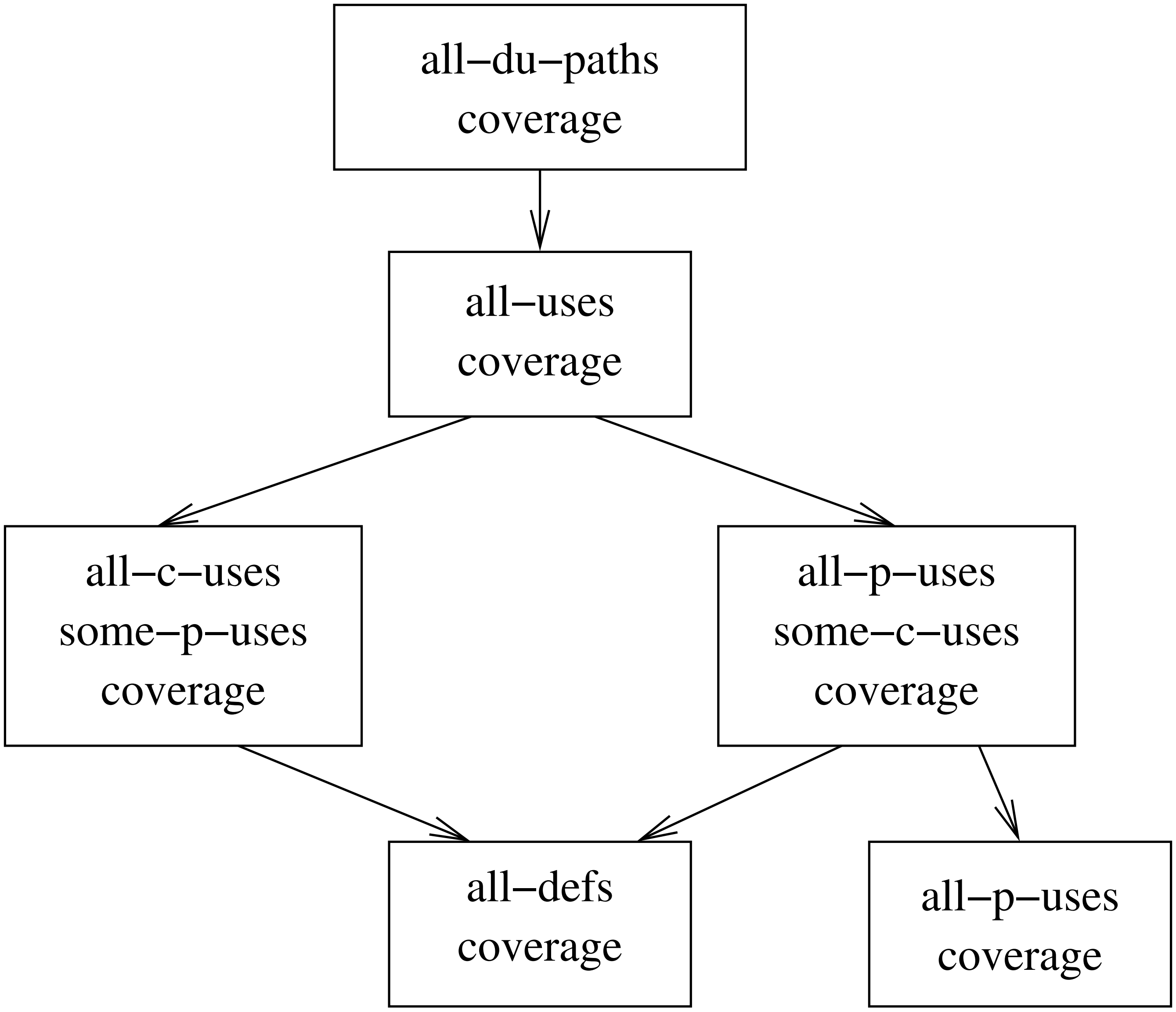
all-p-uses a path to each node in $dpu(s, x)$.

all-p-uses/some-c-uses a path to each node in $dpu(s, x)$, but if $dpu(s, x)$is empty, at least one path to some node in $dcu(s, x)$.

all-c-uses/some-p-uses a path to each node in $dcu(s, x)$, but if $dcu(s, x)$is empty, at least one path to some node in $dpu(s, x)$.

all-uses a path to each node in $dpu(s, x)$and to each node in $dcu(s, x)$.

all-du-paths all the paths to each node in $dpu(s, x)$and to each node in $dcu(s, x)$.


all–du–paths coverage
all–uses coverage
all–c–uses some–p–uses coverage
all–p–uses some–c–uses coverage
all–defs coverage
all–p–uses coverage