Model-Based Testing

(DIT848 / DAT260)
Spring 2012

Lecture 6
Selecting your tests

Gerardo Schneider
Department of Computer Science and Engineering
Chalmers | University of Gothenburg

About coverage criteria

- Test selection criteria help to design black-box test suites
  - They do not depend on the SUT code
- Model coverage criteria and SUT code coverage are complementary
- In white-box testing, coverage criteria are used for:
  - Measuring the adequacy of test suite
  - Deciding when to stop testing
- Coverage criteria may be used prescriptively
  - "Try to cover all branches"
- Test generation tools can provide metrics on how well the coverage was, and which parts of the model where not covered

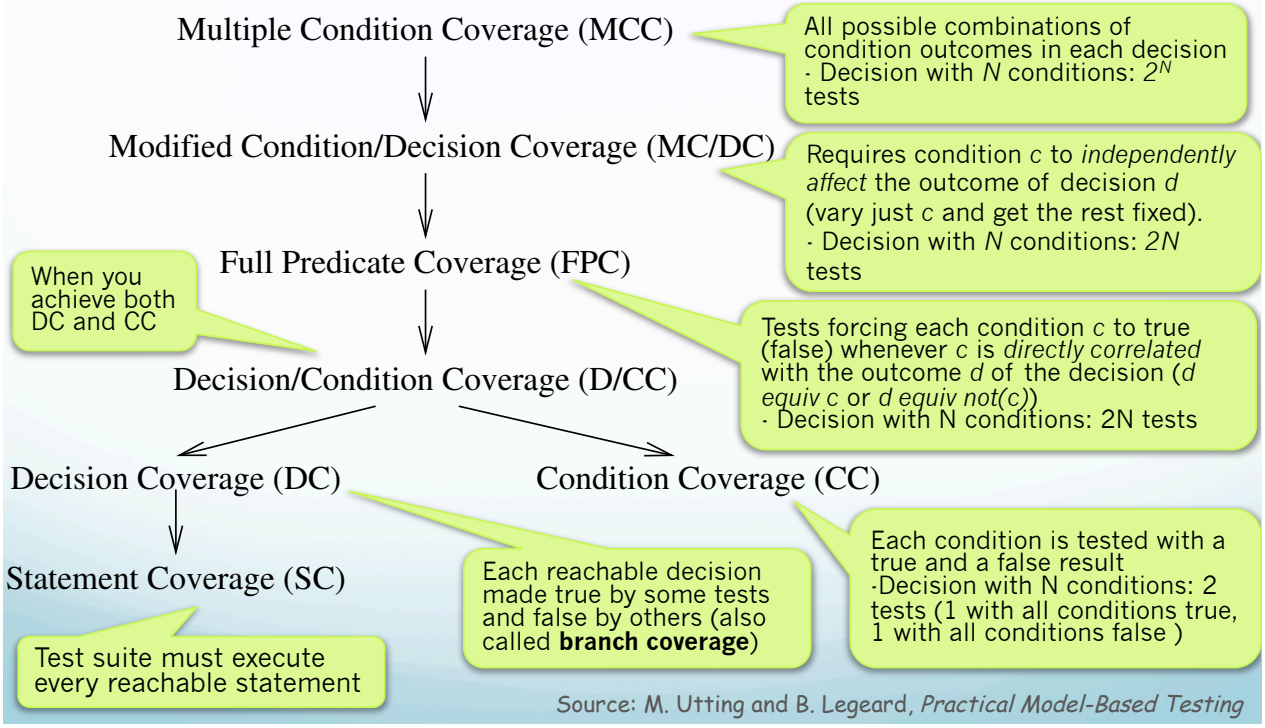
Test selection criteria

1. Structural model coverage criteria
2. Data coverage criteria
3. Fault-model criteria
4. Requirements-based criteria
5. Explicit test case specifications
6. Statistical test generation methods

Structural model coverage

- Major issue: measure and maximize coverage of the **model**
  - Not of the SUT
- Different "families" of structural model coverage criteria:
  1. Control-flow-oriented coverage criteria
  2. Data-flow-oriented coverage criteria
  3. Transition-based coverage criteria
  4. UML-based coverage criteria

Structural model coverage Control-flow oriented



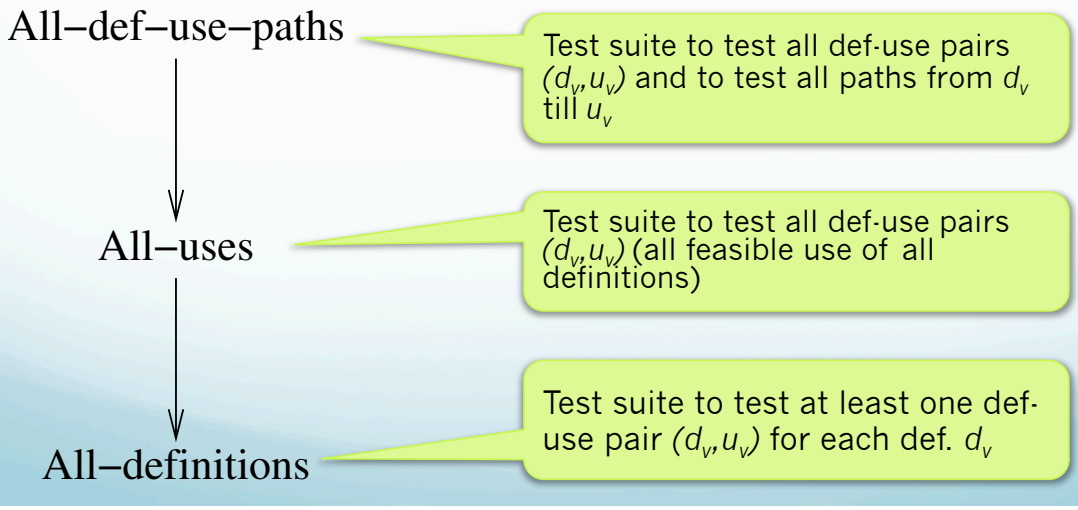
Source: M. Utting and B. Legeard, *Practical Model-Based Testing*

Structural model coverage Control-flow oriented

- **Often combined with transition-based and data-oriented coverage criteria**
- Code coverage are based on statements, decisions (branches), loops, and paths
- In some modeling notations (eg. UML/OCL, B) there are no loops!
- Path coverage (test suite must execute every satisfiable path through the control-flow graph) not possible in code-based testing
  - In pre/post notations: if all combinations of decision outcomes are tested, path coverage is obtained (?!)
  - … so, no path coverage in previous slide ☺

Structural model coverage Data-flow oriented

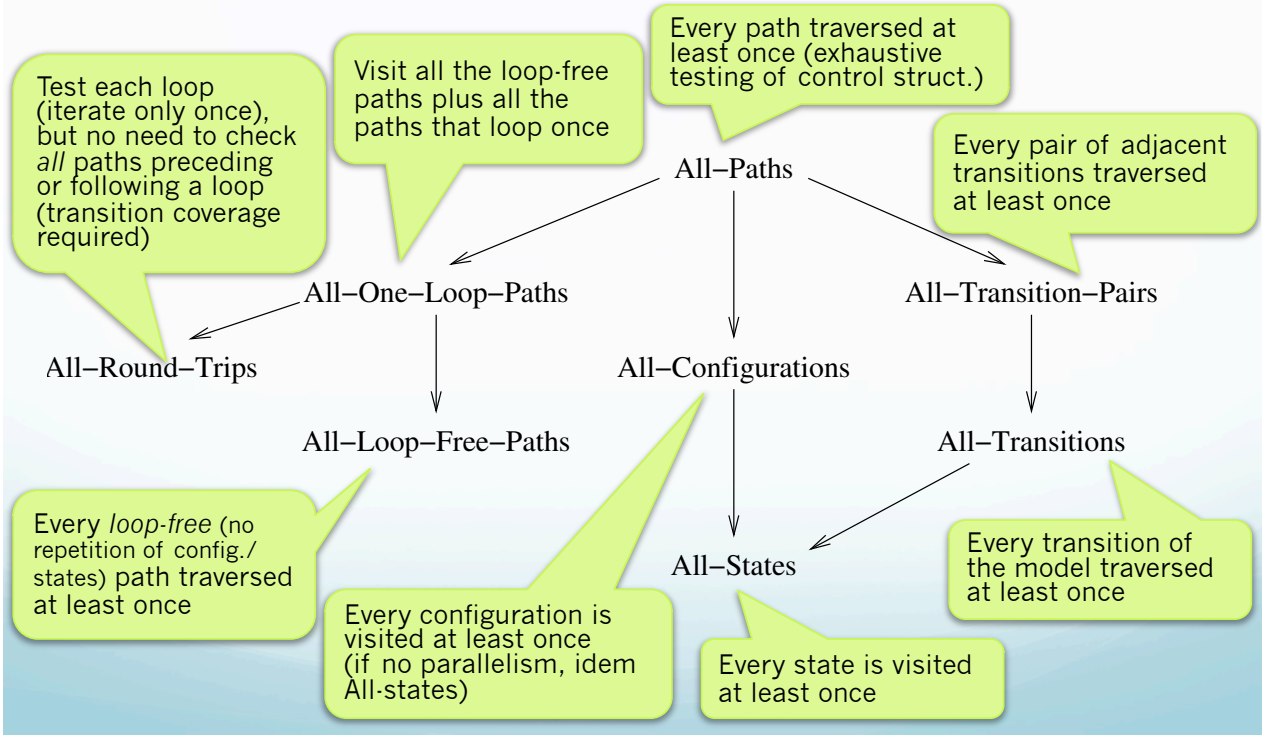
- Control-flow graphs can be annotated with extra information on the **definition** and **use** of data variables
- Def-use pair (d_v,u_v) – d_v is a definition of v, u_v is its use



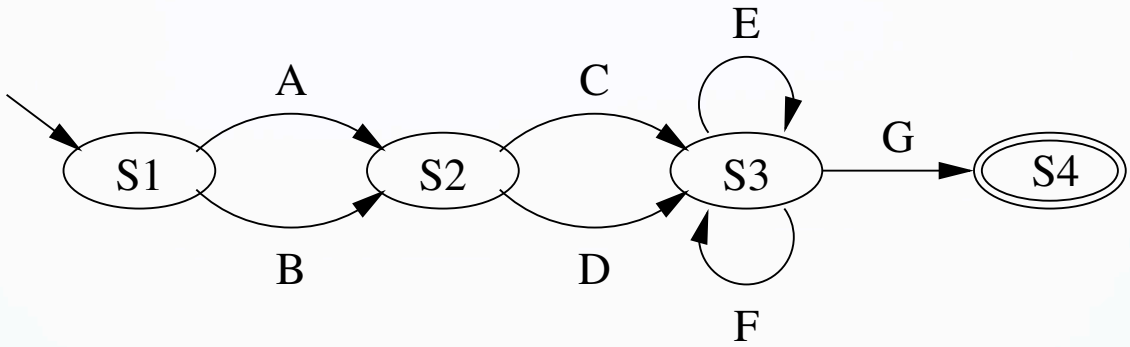
Structural model coverage Transition-based

- Transitions systems made up of states and transitions
- Depending on notation, transitions labeled with inputs, outputs, events, guards, and/or actions
- Usually models parallel systems
- A configuration is roughly a snapshot of the active states (of each parallel process)
- In this coverage criteria we restrict to **reachable** paths

Structural model coverage Transition-based



Structural model coverage Transition-based

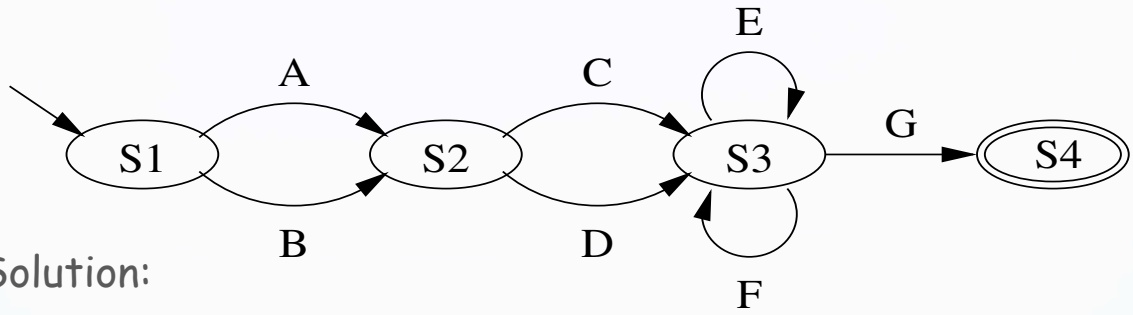


- Write down examples of each transition-based coverage criteria for the above FSM
  - All-states
  - All-configurations
  - All-transitions
  - All-transition-pairs
  - All-loop-free-paths
  - All-one-loop-paths
  - All-round-trips
  - All-paths

Groups 2-5 persons: 15-20 min

Source: M. Utting and B. Legeard, *Practical Model-Based Testing*

Structural model coverage Transition-based



Solution:

- All-states
  - A;C;G
- All-configurations
  - Equal to All-states
- All-transitions
  - A;C;E;F;G and B;D;G
- All-transition-pairs
  - Eg..at state S2: A;C, A;D, B;C, B;D
- All-loop-free-paths
  - A;C;G, A;D;G, B;C;G, B;D;G
- All-one-loop-paths
  - 4 paths of all-loop-free-paths + combination of each of these with a single loop around either E or F transition (4+2*4=12 tests)
- All-round-trips
  - A;C;E, A;C;F, A;C;G, A;D, B
- All-paths
  - 4 paths of all-loop-free-paths but extended with any number of E and F transitions

Source: M. Utting and B. Legeard, *Practical Model-Based Testing*

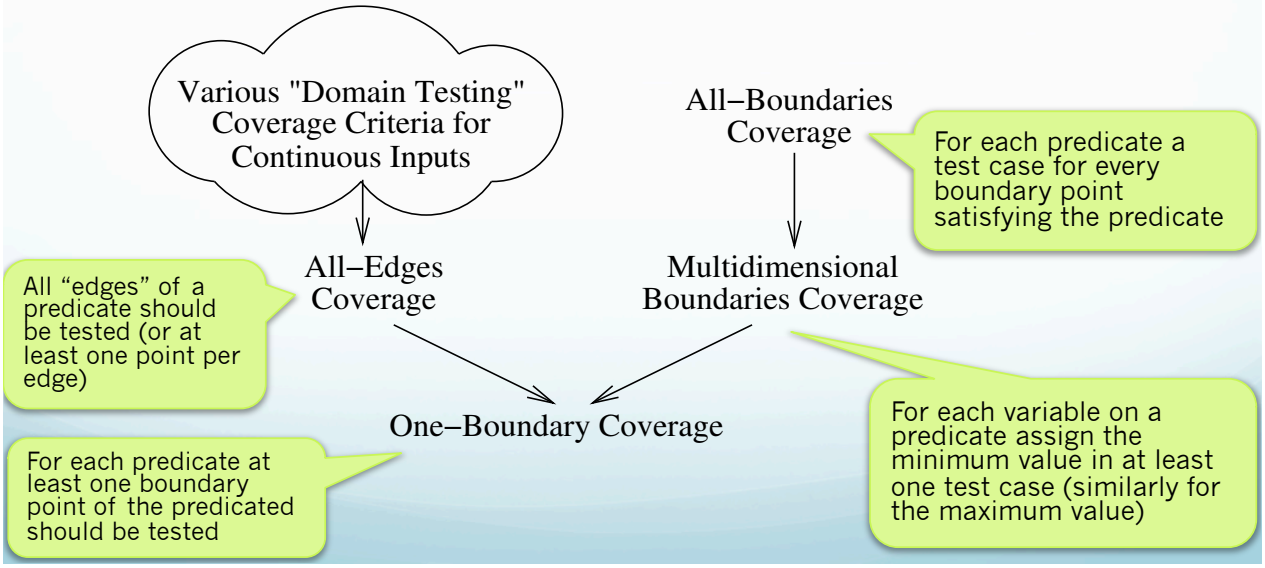
Data coverage criteria

- Useful for choosing good data value representatives as test inputs
- Over a domain *D*, two extreme data coverage criteria
  - **One-value**: at least one value from *D* (in combination with other test criteria might be useful)
  - **All-values**: every value in *D*. Not practical in general
- More realistic:
  1. Boundary values
  2. Statistical data coverage
  3. Pairwise testing

Data coverage criteria Boundary value testing

Choosing values at the boundaries of input domains

- Usually constraints on values are predicates representing some regions: 1<=x<=5 and 2<=y<=7



Data coverage criteria Statistical data coverage

- Choosing random tests is as good as finding faults as partition testing
  - Could then be more cost-effective
- Criterion: Random-value coverage (with distribution D)
  - Values of a given data variable in the test suit to follow the statistical distribution D

Example:

car_speed >50 and rain_level >5 (with car_speed: 0..300 and rain_level: 0..10)

- Boundary testing: 4 tests (51 and 300 for car, 6 and 10 for rain)
- If we want 50 tests: generate them randomly with some distribution:

Data coverage criteria Boundary value testing

1. Write a geometrical representation of the following predicate, and consider what could be the boundary values for such predicate (integer)

$$(x^2+y^2<=25)\ \&\ (0<=y)\ \&\ (x+y<=5)$$

2. Write boundary-oriented coverage for the case above so you achieve
   - All-boundaries coverage
   - Multidimensional-boundaries coverage
   - All-edges coverage

Groups 2-5 persons: 10 min

Data coverage criteria Boundary value testing

Solution:

- All-boundaries coverage
  - The 22 boundary points depicted in the picture
- Multidimensional-boundaries coverage
  - Tests: (5,0), (-5,0), (0,5), and (x,0), for any -5<=X<=5
- All-edges coverage
  - Eg. (5,0) and (0,5)

Utting & Legeard book: Fig. 4.7, pp.125!

Source: M. Utting and B. Legeard, *Practical Model-Based Testing*

Fault-based criteria

- A software testing technique using test data designed to demonstrate the absence of a set of **pre-specified faults** (known or recurrent faults)
- Mutation testing: program mutants are created by syntactic transformation of the SUT
  - Using **mutation operators**
- Executing a test suite on all mutants allows to measure the percentage of mutants **killed** by the test suite (exposing a fault in the mutant)
- Mutation of operators also guide the design of tests
  - Tests helping to distinguish a program from its mutant

Requirements-based criteria

- Each requirement (*a testable statement of some functionality that the product must have*) should be tested
- Requirements can be used both to measure a level of coverage for the generated test case and to drive the test generation itself
- All-requirements coverage
  - Record the requirements inside the behavioral model (as annotations)
  - Formalize each requirement and use it as a test selection criterion

Explicit test case specifications

- Besides the model, the tester writes test case specifications in some formal notation
- Used to determine which tests to generate
- Notation could be the same as the modeling language, but not necessarily
- FSMs, regular expressions, temporal logic, Markov chains, etc.
- Give precise control over generated tests

Statistical test generation methods

- In MBT usually used to generate test sequences from environmental models
- Usually using Markov chains (roughly, a FSM with probabilities)
- Test cases with greater probability to be generated first (and more often if organized in different classes)

Combining test selection criteria

- Criteria seen have different scopes and purposes: good to combine them
- See some interesting examples in Utting & Legeard, section 4.7 (pp.134-135)

References

- M. Utting and B. Legeard, *Practical Model-Based Testing*. Elsevier - Morgan Kaufmann Publishers, 2007
  - Chapter 4