Year 2 Review
Paris, November 8th and 9th, 2006

Testing & Verification Scientific Highlights :

Coverage metrics for testing

Ed Brinksma

University of Twente, Enschede, NL

Embedded Systems Institute, Eindhoven, NL

Coverage: Motivation

- Testing is inherently incomplete
  - exhaustive testing needs very large or infinite test suites
  - test selection is crucial
- Coverage metrics
  - quantitative evaluation of test suite
  - measure how much of specification/implementation has been examined
- Examples:
  - White box (implementation coverage):
    - Statement, path, condition coverage
  - Black box (specification coverage)
    - State, transition coverage

Common approaches

- Existing coverage measures
  - statement, path, condition coverage
  - state, transition coverage
  - etc ...
- Disadvantages
  - strongly syntactic in nature

    replacing the spec by an equivalent one yields different coverage
  - uniform, application-insensitive measures

    all system parts treated equally important; BUT
    - some bugs are more important than others;
    - should test crucial behaviour first/better

Our Approach

Considers black box coverage

- similar ideas could apply to white box coverage

Is semantic

- Semantically equivalent specs yield same coverage

Is risk-based

- more important bugs/system parts

→ higher contribution to coverage

Allows for optimization

- Cheapest test suite with 90% coverage
- Maximal coverage within cost budget

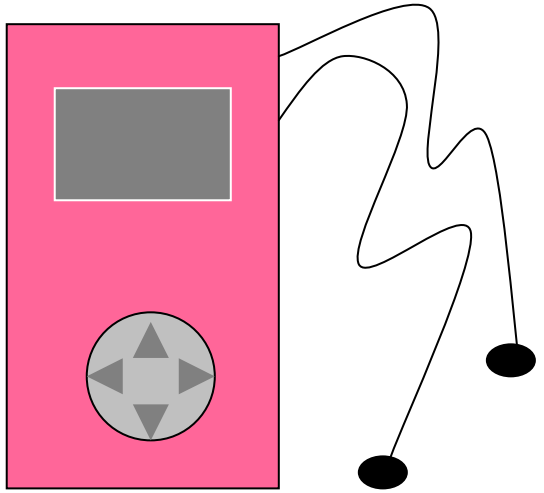
Fault models

f: *Traces* → $R^{\geq 0}$

f(play? song!) = 0 correct

f(play? silence!) = 10 incorrect

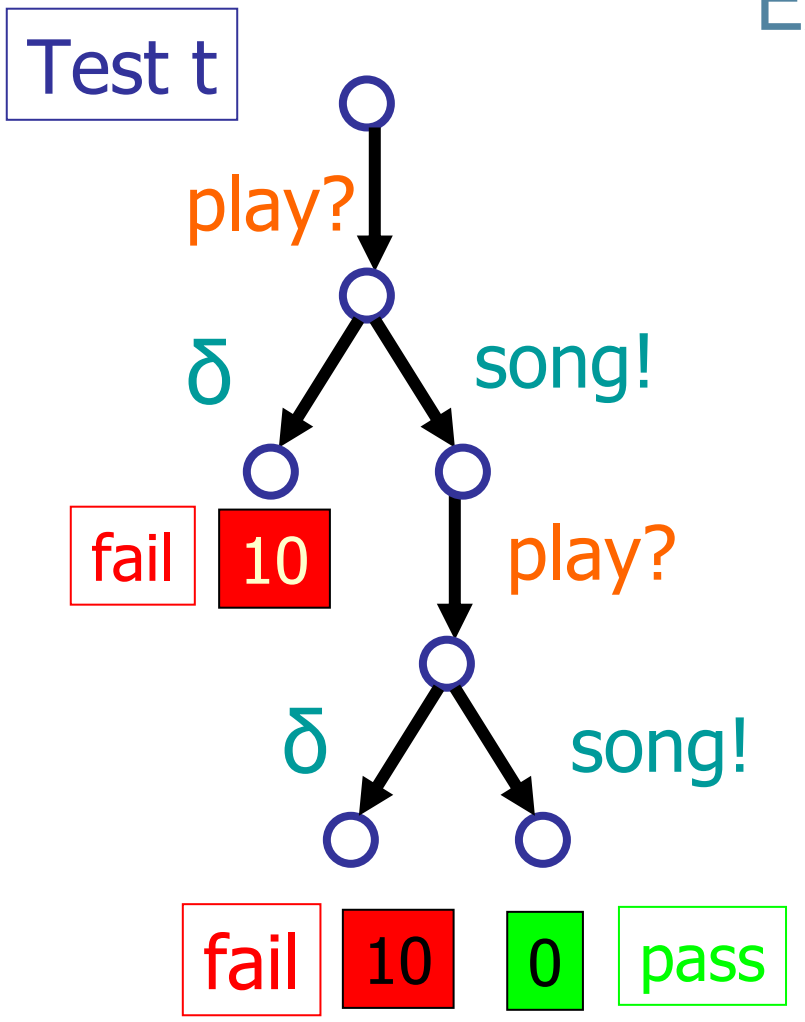
f(song!) = 3 incorrect



f(σ) denotes severity of errors after σ

- f(σ) = 0 : correct behaviour
- f(σ) > 0 : incorrect behaviour
- $0 < \Sigma_{\sigma} f(\sigma) < \infty$

Example



Here

- f(play? song!) = 0
- f(play? δ) = 10
- f(play? song! play? δ) = 10
- f(song!) = 3

Suppose $\Sigma_\sigma f(\sigma) = 100$

Absolute Coverage abscov(f,t)

- sum the error weights
- 10 + 10 + 0 = 20

Relative Coverage =

$$\frac{\text{abscov(f,t)}}{\text{totcov(f)}} = \frac{20}{100}$$

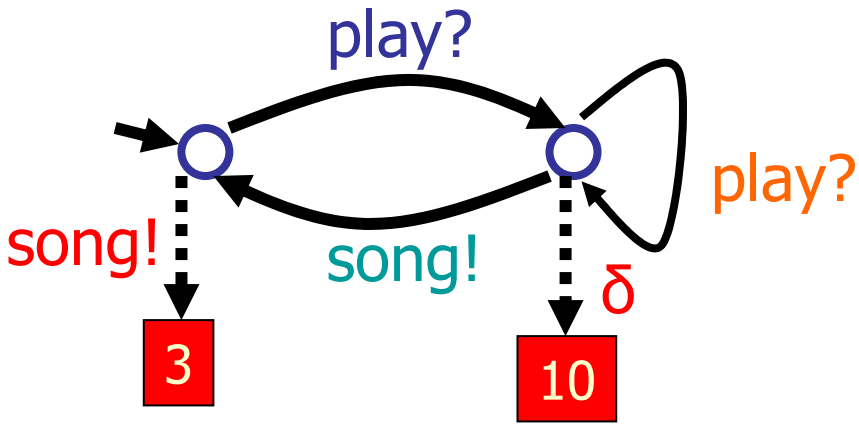
should be ≠ 0, ≠∞

Fault specifications

Use your favorite Formalism, e.g. UML, state charts, LOTOS, etc

fault model

- $f(\sigma) = 0$ if σ trace of automaton
- $f(\sigma) = 3 \cdot \alpha^{|\sigma|-1}$ if σ end in 3-state
- $f(\sigma) = 10 \cdot \alpha^{|\sigma|-1}$ if σ ends in 10-state



infinite total coverage !!

- $\sum_{\sigma} f(\sigma) = 3 + 10 + 3 + 10 + \ldots = \infty$

Solution 1: restrict to traces of lenght K

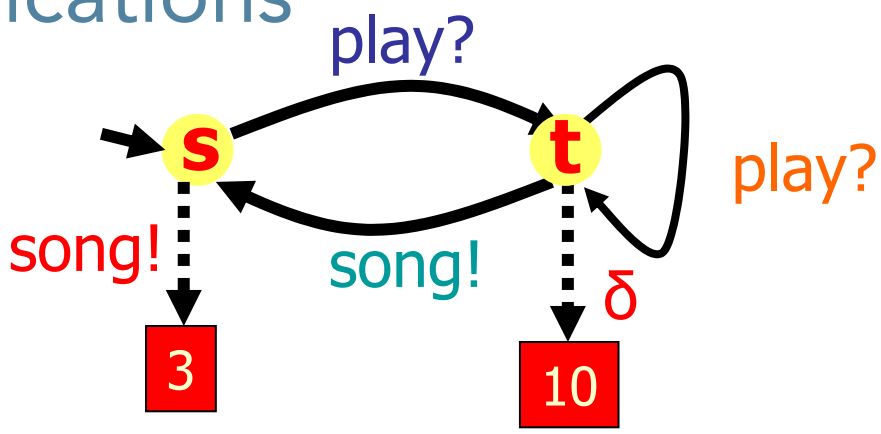
- Omit here, works as solution 2, less efficient, more boring

Solution 2: discounting

- errors in short traces are worse
- lower the weight proportional to length

- α < 1/out(state)
- α can vary per transition
- tune α

Fault specifications



Total coverage becomes finite & computable:

$tc(s) = 3 + \alpha\, tc(t)$

$tc(t) = 10 + \alpha\, tc(t) + \alpha\, tc(s)$

$tc(x) = wgt(x) + \alpha \sum_{y:\, succ(x)} tc(y)$

Solve linear equations

$tc = wgt\ (I - \alpha A)^{-1}$ A adjacency matrix

$tc(s) = \dfrac{10 - 3\alpha}{1 - \alpha - \alpha^2}$

Relative Coverage

$\dfrac{abscov(f,t)}{totcov(f)}$

Properties

Framework for black box coverage

- Robustness
  - Small changes in weight yield small changes in coverage
  - RelCov(spec) continuous in the weights
- Calibration
  - tune α : shift coverage as deep into the tree as desired

Conclusions

Framework for black box coverage

- Semantic:
  - Equivalent fault specs have same coverage
- Risk-based
  - Important bugs contribute more to coverage
- Allows for optimization
  - based on optimization techniques
  - In discounted model (linear algebra)
  - In undiscounted (fixed length) model (graph theory)

Future work

- Application to testing larger protocols
  - bench mark protocols exist for model-driven testing
  - TorX/Maple
- Theory
  - addition of costs/probabilities
- Implementation:
  - In Torx, model-based test tool