

A semantic framework for test coverage

(test coverage for risk-based specifications)

Ed Brinksma

Laura Brandan Briones

Mariëlle Stoelinga

Coverage: Motivation

- Testing is inherently incomplete
 - Test selection is crucial
- Coverage metrics
 - Quantitative evaluation of test suite
 - Count how much of specification/implementation have been examined
- Examples:
 - White box (implementation coverage):
 - Statement, path, condition coverage
 - Black box (specification coverage)
 - State, transition coverage

Disadvantages

- Existing coverage measures:
 - Statement, path, condition coverage
 - State, transition coverage
 - Etc ...
- syntactic
 - replacing the spec by an equivalent one yields different coverage
- Equality metrics
 - all system parts treated equally important;
 - BUT: some bugs are more important than others;
 - test crucial behavior first and better

Our Approach

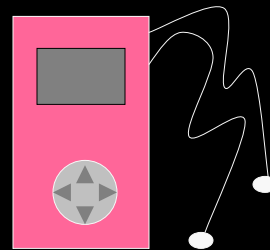
- Considers black box coverage
 - similar ideas could apply to white box coverage
- Is semantic
 - Semantically equivalent specs yield same coverage
- Is risk-based
 - more important bugs/system parts
 - higher contribution to coverage
- Allows for optimization
 - Cheapest test suite with 90% coverage
 - Maximal coverage within cost budget

Overview

1. Fault models
 - General framework for coverage
2. Fault automata
 - specification formalism for fault models
3. Computing coverage
 - of a test suite
 - optimization
4. Conclusions and Future Work

Fault models

- $f: Behaviors \rightarrow R^{\geq 0}$
 - $f(\sigma) = 0$: correct behavior
 - $f(\sigma) > 0$: incorrect behavior
: $f(\sigma)$ severity
 - $0 < \sum_{\sigma} f(\sigma) < \infty$
- Behaviors are traces
 - $Behaviors = L^*$
 - $L = (L_I, L_U)$
- How to obtain f ?
 - E.g. via fault automata

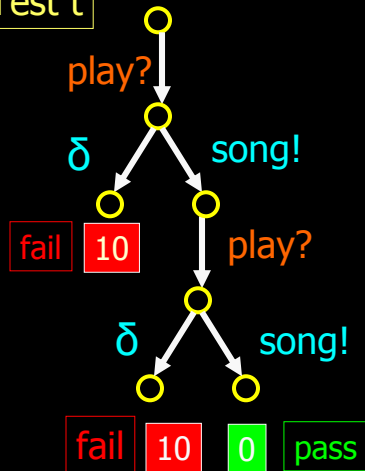


$f: L^* \rightarrow R^{\geq 0}$

$f(\text{play? song!}) = 0$	correct
$f(\text{play? silence!}) = 10$	incorrect
$f(\text{song!}) = 3$	incorrect

Example

Test t



- $f: L^* \rightarrow R$
 - $f(\text{play? song!}) = 0$
 - $f(\text{play? } \delta) = 10$
 - $f(\text{play? song! play? } \delta) = 10$
 - $f(\text{song!}) = 3$
- $\sum_{\sigma} f(\sigma) = 100$

- Absolute Coverage $\text{abscov}(f,t)$
 - sum the error weights
 - $10 + 10 + 0 = 20$

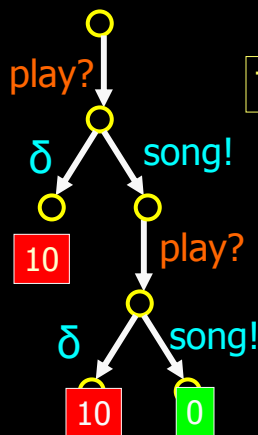
- Relative Coverage

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

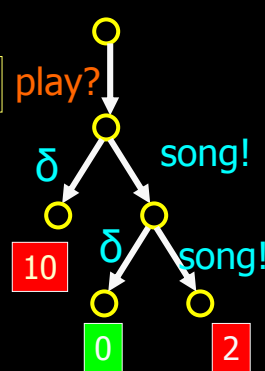
should be $\neq 0, \neq \infty$

Example

Test t



Test t'



Absolute Coverage

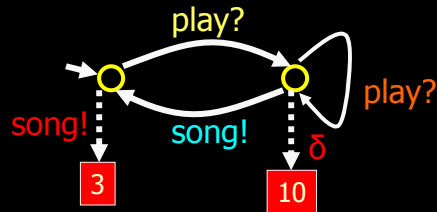
- count each trace once !
- $10 + 10 + 0 + 0 + 2 = 22$

Relative Coverage

$$\frac{\text{abscov}(f,t)}{\text{totcov}(f)} = \frac{22}{100}$$

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

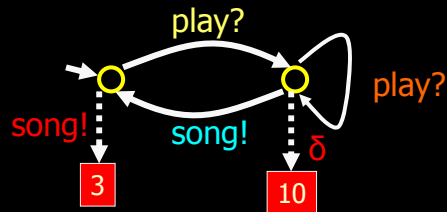
infinite total coverage !!

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

solution: discounting

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

Fault specifications



fault model

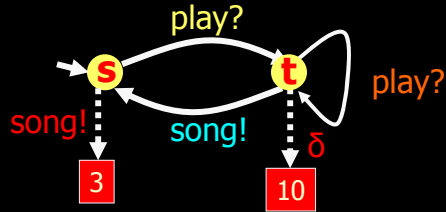
- $f(\sigma) = 0$ if σ trace of automaton
- $f(\sigma) = 3 \cdot \alpha^{|\sigma|-1}$ if σ end in 3-state
- $f(\sigma) = 10 \cdot \alpha^{|\sigma|-1}$ if σ ends in 10-state

Example

- $f(\text{play?}) = 0$
- $f(\text{play? } \delta) = 10 \cdot \alpha$
- $f(\text{play? song! song!}) = 3 \cdot \alpha^2$
-

- $\alpha < 1/\text{out}(\text{spec}) = 1/2$
- α can vary per transition
- Tune α

Fault specifications



Total coverage becomes finite & computable:

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

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

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

$$tc(s) =$$

$$\frac{10 - 3\alpha}{1 - \alpha - \alpha^2}$$

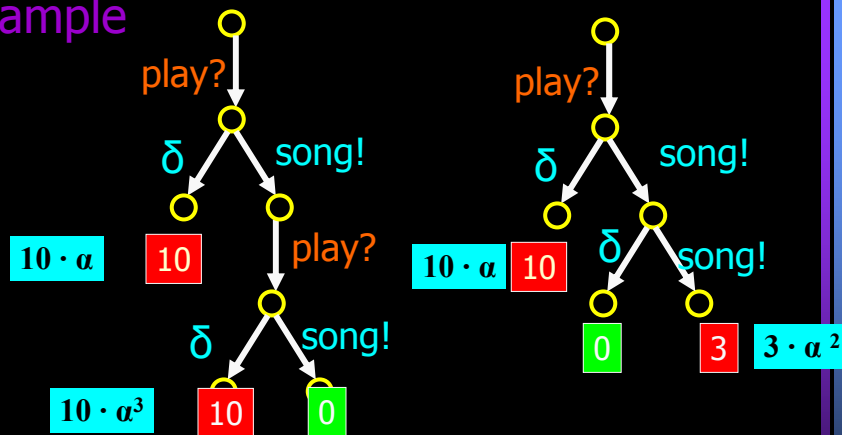
Solve linear equations

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

Relative Coverage

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

Example



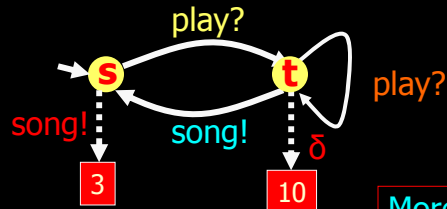
Absolute Coverage

- count each trace once !
- Do it smart;
- Merge test cases first

Relative Coverage

$$\frac{abscov(f,t)}{totcov(f)} = \frac{10\alpha + 10\alpha^2 + 3\alpha^3}{1 - \alpha - \alpha^2}$$

Optimization



Find best test case of length n

$$v_1(s) = 3$$

$$v_1(t) = 10$$

$$v_{k+1}(s) = \max(3, \alpha v_k(t))$$

$$v_{k+1}(t) = \max(10 + \alpha v_k(s), \alpha v_k(t))$$

Complexity: $O(n \text{ #transitions in spec})$

More optimization:

Test suite of k tests
& length n ;

Best test case in
budget;

Add costs

Properties

Framework for black box coverage

- **Robustness**
 - Small changes in weight yield small changes in coverage
 - $\text{Relcov}(\text{spec})$ continuous
- **Is tunable**
 - Change α : get as much total coverage as desired

Conclusions

Framework for black box coverage

- Semantic:
 - Equivalent fault specs have same coverage
- Risk-based
 - Important bugs contribute more to coverage
- Enables optimization
 - Based on optimization techniques

Future work

- Application to conference/chat protocol
 - Several people can join/leave chat sessions
 - Bench mark in model-driven testing
 - TorX/Maple
- Theory
 - Add costs/probabilities
- Implementation:
 - In Torx, model-based test tool