Interpreting a successful testing process: risk and actual coverage

Mariëlle Stoelinga, **Mark Timmer**
University of Twente

3rd IEEE International Symposium on
Theoretical Aspects of Software Engineering

July 31, 2009

Contents

Introduction – Testing

Why testing?

- Software becomes more and more complex
- Research showed that billions can be saved by testing better
- No need for the source code (black-box perspective)

Introduction – Testing

Why testing?

- Software becomes more and more complex
- Research showed that billions can be saved by testing better
- No need for the source code (black-box perspective)

Model-based testing

- Precise and formal
- Automatic generation and evaluations of tests
- Repeatable and scientific basis for product testing

Introduction – Risk and coverage

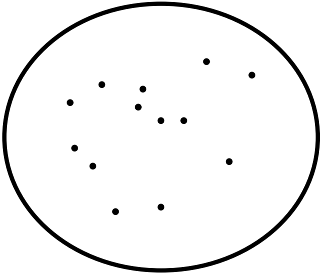
Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage

Introduction – Risk and coverage

Why do we need risk and coverage?

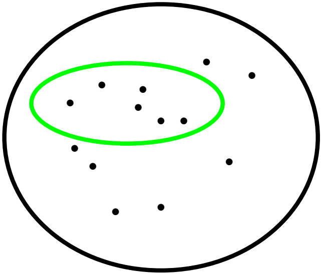
- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage



Introduction – Risk and coverage

Why do we need risk and coverage?

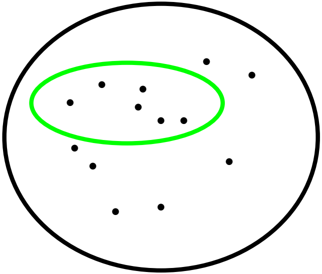
- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage



Introduction – Risk and coverage

Why do we need risk and coverage?

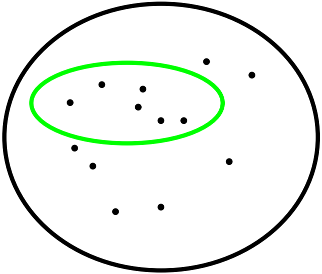
- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage



Informal calculation

Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage

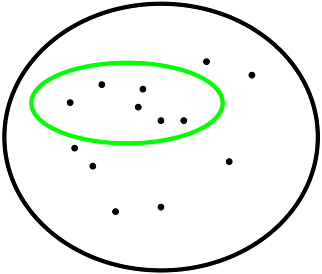


Informal calculation

Coverage: $\frac{6}{13} = 46\%$

Why do we need risk and coverage?

- Testing is inherently incomplete
- Testing does increase our confidence in the system
- A notion of *quality* of a test suite is necessary
- Two fundamental concepts: risk and coverage



Informal calculation

Coverage: $\frac{6}{13} = 46\%$

Risk: $7 \cdot 0.1 \cdot \$10 = \$7$

Existing coverage measures

- Statement coverage
- State/transition coverage

Existing coverage measures

- Statement coverage
- State/transition coverage

Limitations:

- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view

Introduction – Existing approaches

Existing coverage measures

- Statement coverage
- State/transition coverage

Limitations:

- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view

Existing risk measures

- Bach
- Amland

Introduction – Existing approaches

Existing coverage measures

- Statement coverage
- State/transition coverage

Limitations:

- all faults are considered of equal severity
- likely locations for fault occurrence are not taken into account
- syntactic point of view

Existing risk measures

- Bach
- Amland

Limitations:

- Informal
- Based on heuristics
- Only identify testing order for components

Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

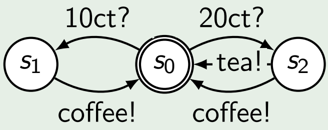
- System considered as black box
- Semantic point of view
- Fault weights

Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



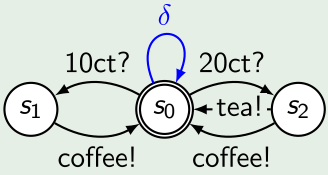
Introduction – Preliminaries

Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems

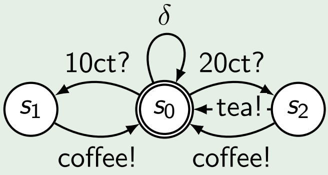


Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



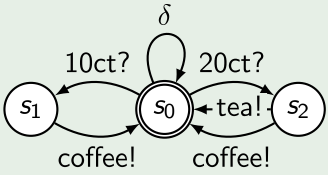
10ct? coffee! 20ct? tea! δ

Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

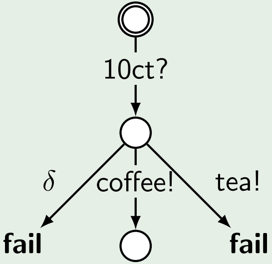
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



10ct? coffee! 20ct? tea! δ

Test cases

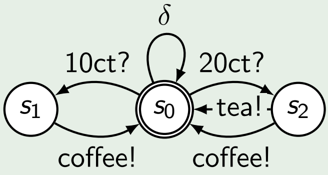


Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

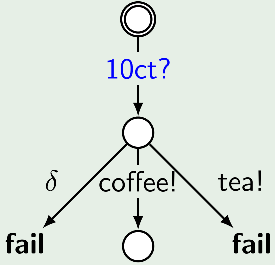
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



10ct? coffee! 20ct? tea! δ

Test cases

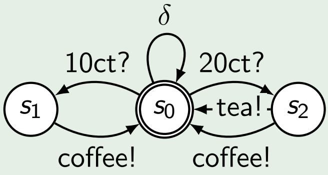


Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

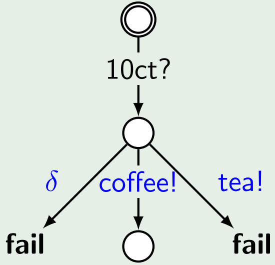
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



10ct? coffee! 20ct? tea! δ

Test cases

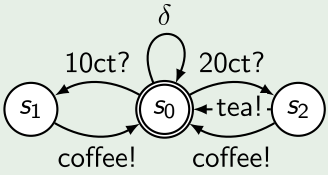


Starting point: semantic coverage

Previous work by Brandán Briones, Brinksma and Stoelinga

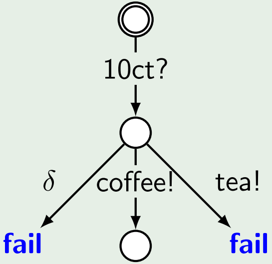
- System considered as black box
- Semantic point of view
- Fault weights

Labelled transition systems



10ct? coffee! 20ct? tea! δ

Test cases



The WFS Model – Weighted Fault Specifications

Weighted fault specification

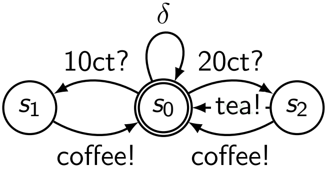
A WFS^- consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)

Weighted fault specification

A WFS$^-$ consists of

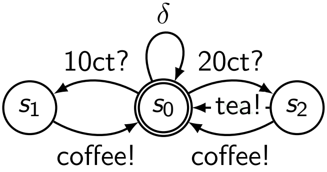
- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)



Weighted fault specification

A WFS$^-$ consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)

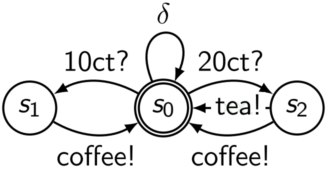


$$p_{\mathrm{err}}(10\text{ct? coffee!}) = 0.02$$
$$p_{\mathrm{err}}(20\text{ct? tea!}) = 0.03$$

Weighted fault specification

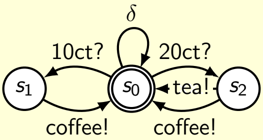
A WFS^{-} consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)



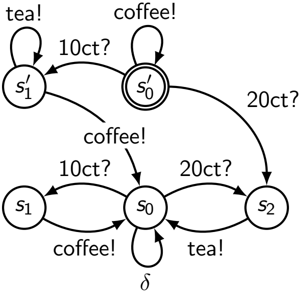
$$p_{\mathrm{err}}(\text{10ct? coffee!}) = 0.02$$
$$p_{\mathrm{err}}(\text{20ct? tea!}) = 0.03$$

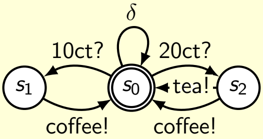
$$w(\epsilon) = 10$$
$$w(\text{10ct?}) = 15$$
$$w(\text{10ct? coffee!}) = 9.5$$



$$w(\epsilon) = 10$$
$$w(\text{10ct?}) = 15$$
$$w(\text{10ct? coffee!}) = 9.5$$

The WFS Model – Fault Weight



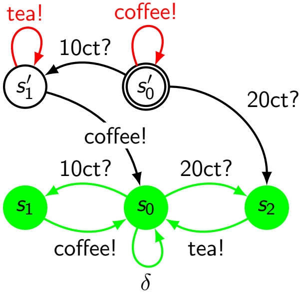


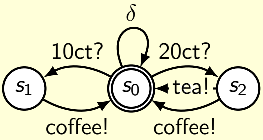
$$w(\epsilon) = 10$$

$$w(10ct?) = 15$$

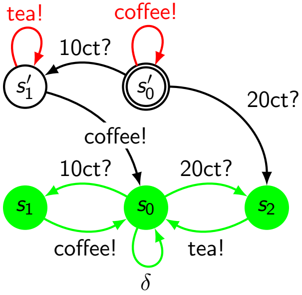
$$w(10ct?\ \ coffee!) = 9.5$$

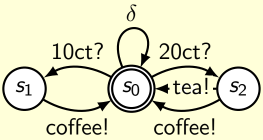
The WFS Model – Fault Weight





$$w(\epsilon) = 10$$
$$w(10ct?) = 15$$
$$w(10ct?\ coffee!) = 9.5$$

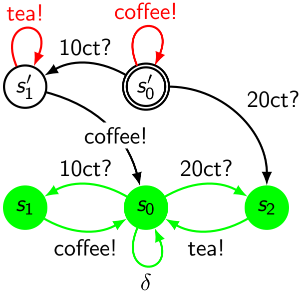


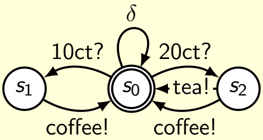


$$w(\epsilon) = 10$$
$$w(10ct?) = 15$$
$$w(10ct?\ coffee!) = 9.5$$

Fault weight: $10 + 15 = 25$

The WFS Model – Fault Weight





$w(\epsilon) = 10$

$w(10ct?) = 15$

$w(10ct?\ coffee!) = 9.5$

Fault weight: $10 + 15 = 25$

(We are only interested in whether a fault can occur, not in which one)

Risk

Definition

Given a test suite T and a passing execution E, we define

$$\text{risk}(T, E) = \mathbb{E}[w(\text{Impl}) \mid \text{observe } E]$$

i.e., the fault weight still expected to be present after observing E.

Risk

Definition

Given a test suite T and a passing execution E, we define

$$\mathrm{risk}(T, E) = \mathbb{E}[w(\mathrm{Impl}) \mid \text{observe } E]$$

i.e., the fault weight still expected to be present after observing E.

Observe:

$$\mathrm{risk}(\langle\rangle, \langle\rangle) =$$

Risk

Definition

Given a test suite T and a passing execution E, we define

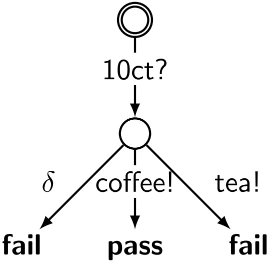
$$\mathsf{risk}(T, E) = \mathbb{E}[w(\mathsf{Impl}) \mid \mathsf{observe}\ E]$$

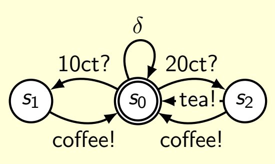
i.e., the fault weight still expected to be present after observing E.

Observe:

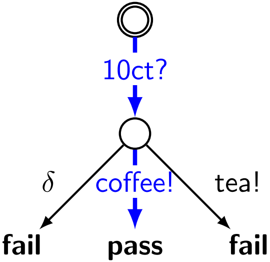
$$\mathsf{risk}(\langle\rangle, \langle\rangle) = \sum_{\sigma} w(\sigma) \cdot p_{\mathrm{err}}(\sigma)$$

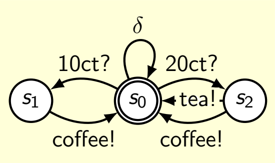
Risk


10ct?
δ
coffee!
tea!
fail
pass
fail

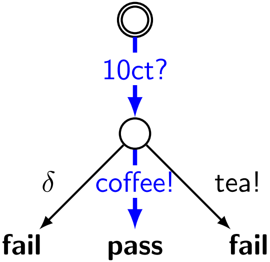

δ
10ct?
20ct?
s_1
s_0
tea!
s_2
coffee!
coffee!

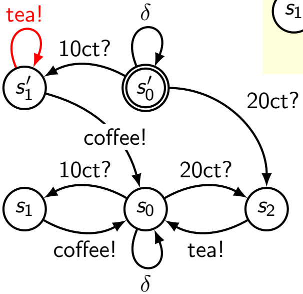
Risk

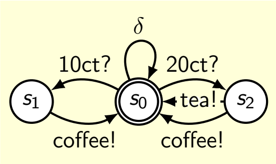

10ct?
δ
coffee!
tea!
fail
pass
fail


δ
10ct?
20ct?
s_1
s_0
tea!
s_2
coffee!
coffee!

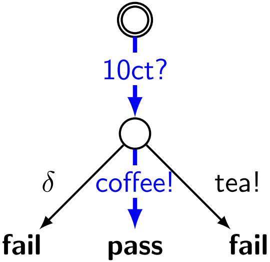
Risk

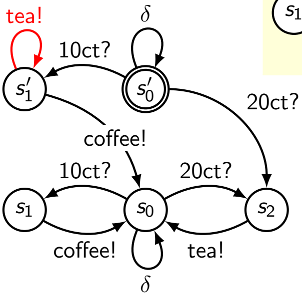

10ct?
δ
coffee!
tea!
fail
pass
fail

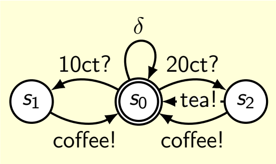

tea!
δ
10ct?
s_1'
s_0'
20ct?
coffee!
10ct?
20ct?
s_1
s_0
s_2
coffee!
tea!
δ


δ
10ct?
20ct?
s_1
s_0
tea!
s_2
coffee!
coffee!

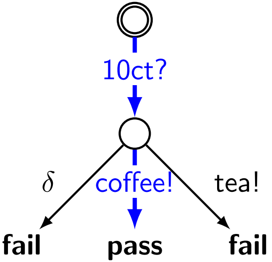
Risk

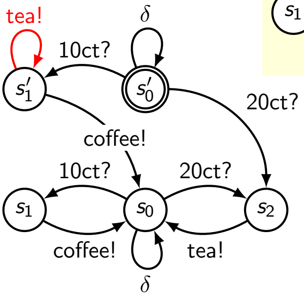


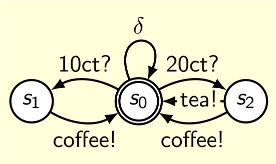




Nondeterministic output behaviour yields difficulties.



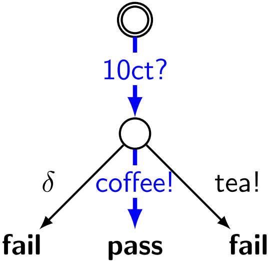


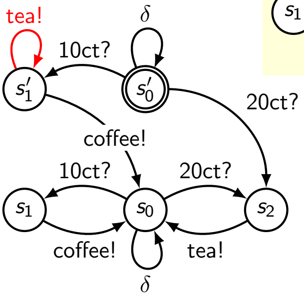


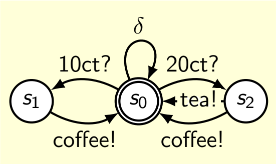
Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

Risk





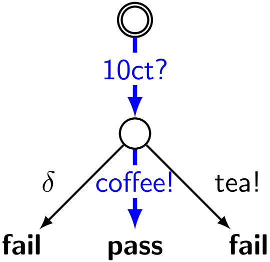


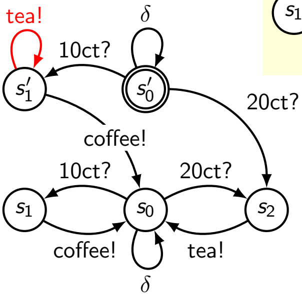
Nondeterministic output behaviour yields difficulties.

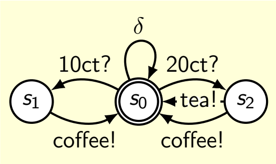
How to calculate risk (expected fault presence)?

$$\text{risk}(\langle\rangle, \langle\rangle) = \sum_{\sigma} w(\sigma) \cdot p_{\text{err}}(\sigma)$$

Risk





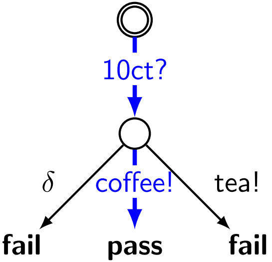


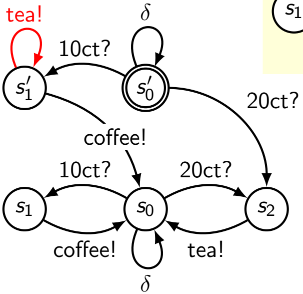
Nondeterministic output behaviour yields difficulties.

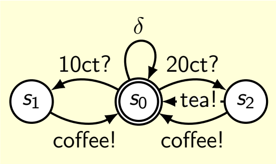
How to calculate risk (expected fault presence)?

$$\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + f(10\text{ct?})$$

Risk





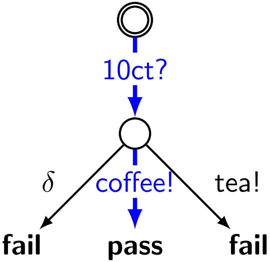


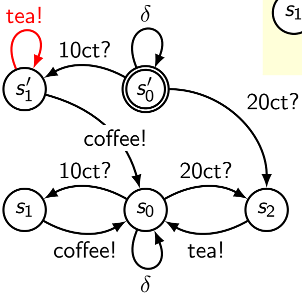
Nondeterministic output behaviour yields difficulties.

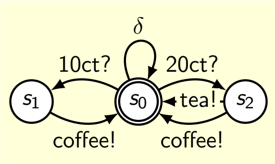
How to calculate risk (expected fault presence)?

$$\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + f(10\text{ct?})$$

Risk







Nondeterministic output behaviour yields difficulties.

How to calculate risk (expected fault presence)?

$$\text{risk}(T, E) = \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after } 10\text{ct?} \mid E]$$

Weighted Fault Specifications (revisited)

Weighted fault specification

A WFS consists of

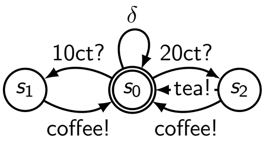
- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)

Weighted Fault Specifications (revisited)

Weighted fault specification

A WFS consists of

- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)

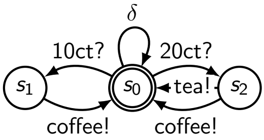


Weighted Fault Specifications (revisited)

Weighted fault specification

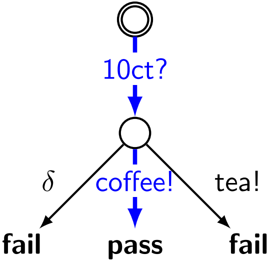
A WFS consists of

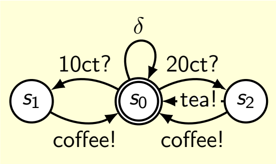
- An LTS (expected system behaviour)
- An error function (probability of faults)
- A weight function (severity of faults)
- A failure function (probability of failure in case of fault)



$$p_{\text{fail}}(\epsilon) = 1.0$$
$$p_{\text{fail}}(10ct?) = 0.5$$

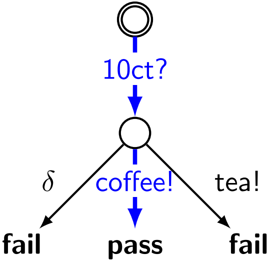
Risk

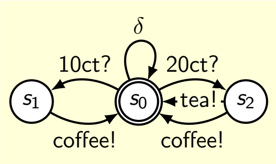




$$\text{risk}(T, E)$$
$$= \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after 10ct?} \mid E]$$

Risk





$$\begin{aligned}
&\text{risk}(T, E) \\
&= \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + w(10\text{ct?}) \cdot \mathbb{P}[\text{error after } 10\text{ct?} \mid E] \\
&= \sum_{\sigma \neq 10\text{ct?}} w(\sigma) \cdot p_{\text{err}}(\sigma) + \\
&\quad w(10\text{ct?}) \cdot \frac{(1 - p_{\text{fail}}(10\text{ct?})) \cdot p_{\text{err}}(10\text{ct?})}{(1 - p_{\text{fail}}(10\text{ct?})) \cdot p_{\text{err}}(10\text{ct?}) + (1 - p_{\text{err}}(10\text{ct?}))}
\end{aligned}$$

Risk

$\text{risk}(T, E) = \mathbb{E}[w(\text{Impl}) \mid \text{observe } E]$

Calculation of risk

$$\text{risk}(T, E) = \text{risk}(\langle\rangle, \langle\rangle) -$$
$$\sum_{\sigma \in E} w(\sigma) \cdot \left(p_{\text{err}}(\sigma) - \frac{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma)}{(1 - p_{\text{fail}}(\sigma))^{\text{obs}(\sigma, E)} \cdot p_{\text{err}}(\sigma) + 1 - p_{\text{err}}(\sigma)} \right)$$

with $\text{obs}(\sigma, E)$ the number of observations in E after σ.

Risk

$\text{risk}(T, E) = \mathbb{E}[w(\text{Impl}) \mid \text{observe } E]$

Calculation of risk

$$\text{risk}(T, E) = \text{risk}(\langle\rangle, \langle\rangle) -$$
$$\sum_{\sigma \in E} w(\sigma) \cdot \left(p_{\mathrm{err}}(\sigma) - \frac{(1 - p_{\mathrm{fail}}(\sigma))^{\text{obs}(\sigma,E)} \cdot p_{\mathrm{err}}(\sigma)}{(1 - p_{\mathrm{fail}}(\sigma))^{\text{obs}(\sigma,E)} \cdot p_{\mathrm{err}}(\sigma) + 1 - p_{\mathrm{err}}(\sigma)} \right)$$

with $\text{obs}(\sigma, E)$ the number of observations in E after σ.

Although $\text{risk}(\langle\rangle, \langle\rangle) = \sum_\sigma w(\sigma) \cdot p_{\mathrm{err}}(\sigma)$ is an infinite sum, it can be calculated smartly:

- w defined by truncation: the sum is already finite
- w defined by discounting: system of linear equations

Compute test suite quality in advance

- Estimate correct system behaviour
- Compute expected risk after passing the test suite

Other Applications

Compute test suite quality in advance

- Estimate correct system behaviour
- Compute expected risk after passing the test suite

Optimisation

- Find the optimal test suite of a given size
- Apply history-dependent backwards induction (Markov Decision Theory)

Other Applications

Compute test suite quality in advance

- Estimate correct system behaviour
- Compute expected risk after passing the test suite

Optimisation

- Find the optimal test suite of a given size
- Apply history-dependent backwards induction (Markov Decision Theory)

Actual Coverage

- Only consider the traces that were actually tested
- Use error probability reduction as coverage measure
- Methods very similar to risk

Limitations and Possibilities

Probabilities might be hard to find, but

- We show what can be calculated, and the required ingredients
- We facilitate sensitivity analysis
- To compute numbers, we have to start with numbers. . .

Limitations and Possibilities

Probabilities might be hard to find, but

- We show what can be calculated, and the required ingredients
- We facilitate sensitivity analysis
- To compute numbers, we have to start with numbers. . .

It looks like we need many probabilities and weights, but

- The framework can be applied at higher levels of abstraction
- Compute risk based on error / failure probabilities of modules

Conclusions and Future Work

Main results

- Formal notion of risk
- Both evaluation of risk *and* computation of optimal test suite
- Easily adaptable to be used as a coverage measure

Conclusions and Future Work

Main results

- Formal notion of risk
- Both evaluation of risk *and* computation of optimal test suite
- Easily adaptable to be used as a coverage measure

Directions for Future Work

- Validation of the framework: tool support, case studies
- Dependencies between errors
- On-the-fly test derivation

Questions