A Semantic Framework for Test Coverage (Extended Version)

Ed Brinksma[+*], Mariëlle Stoelinga[+], and Laura Brandán Briones[+]

[+] Faculty of Computer Science, University of Twente,
P.O.Box 217, 7500AE Enschede, The Netherlands.
{marielle,brandanl}@cs.utwente.nl
[*] Embedded Systems Institute,
P.O.Box 513, 5600MB Eindhoven, The Netherlands.
Ed.Brinksma@esi.nl

Abstract. Since testing is inherently incomplete, test selection is of vital importance. Coverage measures evaluate the quality of a test suite and help the tester select test cases with maximal impact at minimum cost. Existing coverage criteria for test suites are usually defined in terms of syntactic characteristics of the implementation under test or its specification. Typical black-box coverage metrics are state and transition coverage of the specification. White-box testing often considers statement, condition and path coverage. A disadvantage of this syntactic approach is that different coverage figures are assigned to systems that are behaviorally equivalent, but syntactically different. Moreover, those coverage metrics do not take into account that certain failures are more severe than others, and that more testing effort should be devoted to uncover the most important bugs, while less critical system parts can be tested less thoroughly.

This paper introduces a semantic approach to test coverage. Our starting point is a weighted fault model, which assigns a weight to each potential error in an implementation. We define a framework to express coverage measures that express how well a test suite covers such a specification, taking into account the error weight. Since our notions are semantic, they are insensitive to replacing a specification by one with equivalent behaviour. We present several algorithms that, given a certain minimality criterion, compute a minimal test suite with maximal coverage. These algorithms work on a syntactic representation of weighted fault models as fault automata. They are based on existing and novel optimization problems. Finally, we illustrate our approach by analyzing and comparing a number of test suites for a chat protocol.

1 Introduction

After years of limited attention, the theory of testing has now become a widely studied, academically respectable subject of research. In particular, the application of formal methods in the area of model-driven testing has led to a better understanding of the notion of conformance an implementation to a specification. Also, automated generation methods for test suites from specifications (e.g.

[12, 15, 13, 4, 10]) have been developed, which have lead to a new generation of powerful test generation and execution tools, such as SpecExplorer[5], TorX[2] and TGV[7].

A clear advantage of a formal approach to testing is the provable soundness of the generated test suites, i.e. the property that each generated test suite will only reject implementations that do not conform to the given specification. In many cases also a completeness or exhaustiveness result is obtained, i.e. the property that for each non-conforming implementation a test case can be generated that will expose its errors by rejecting it (cf. [12]).

In practical testing the above notion of exhaustiveness is usually problematic. For realistic systems an exhaustive test suite will contain infinitely many tests. This raises the question of test selection, i.e. the selection of well-chosen, finite test suites that can be generated (and executed) within the available resources. Test case selection is naturally related to a measure of coverage, indicating how much of the required conformance is tested for by a given test selection. In this way, coverage measures can assist the tester in choosing test cases with maximal impact against some optimization criterion (i.e. number of tests, execution time, cost).

Typical coverage measures used in black-box testing are the number of states and/or transitions of the specification that would be visited by executing a test suite against it [14]; white-box testing often considers the number of statements, conditional branches, and paths through the implementation code that are touched by the test suite execution [8, 9]. Although these measures do indeed help with the selection of tests and the exposure of faults, they share two shortcomings:

1. The approaches are based on syntactic model features, i.e. coverage figures are based on constructs of the specific model or program that is used as a reference. As a consequence, we may get different coverage results when we replace the model in question with a behaviorally equivalent, but syntactically different one.
2. The approaches fail to account for the non-uniform gravity of failures, whereas it would be natural to select test cases in such a way that the most critical system parts are tested most thoroughly.

It is important to realize that the appreciation of the weight of a failure cannot be extracted from a purely behavioral model, as it may depend in an essential way on the particular application of the implementation under test (IUT). The importance of the same bug may vary considerably between, say, its occurrence as part of an electronic game, and that as part of the control of a nuclear power plant.

Overview. This paper introduces a semantic approach for test coverage that aims to overcome the two points mentioned above. Our point of departure is a weighted fault model that assigns a weight to each potential error in an implementation and we define our coverage measures relative to these weighted fault models.

Since our weighted fault models are infinite semantic objects, we need to represent them finitely if we want to model them or use them in algorithms. We provide such representations by fault automata (Section 4). Fault automata are rooted in ioco test theory [12] (recapitulated in Section 3), but their principles apply to a much wider setting.

We provide two ways of deriving weighted fault models from fault automata, namely the finite depth model (Section 4.1) and the discounted fault model (Section 4.2). The coverage measures obtained for these fault automata are invariant under behavioral equivalence.

For both fault models, we provide algorithms that calculate and optimize test coverage (Section 5). These can all be studied as optimization problems in a linear algebraic setting. In particular, we compute the (total, absolute and relative) coverage of a test suite w.r.t. a fault model. Also, given a test length k, we present an algorithm that finds the test of length k with maximal coverage and an algorithm that finds the shortest test with coverage exceeding a given coverage bound. We apply our theory to the analysis and the comparison of several test suites derived for a small chat protocol (Section 6). We end by providing conclusions and suggestions for further research (Section 7).

2 Coverage measures in weighted fault models

Preliminaries. Let L be any set. The L^* denotes the set of all sequences over L, which we also call *traces* over L. The empty sequence is denoted by ε and $|\sigma|$ denotes the length of a trace $\sigma \in L^*$. We use $L^+ = L^* \setminus \{\varepsilon\}$. For σ, $\rho \in L^*$, we say that σ is a *prefix* of ρ and write $\sigma \sqsubseteq \rho$, if $\rho = \sigma\sigma'$ for some $\sigma' \in L^*$.

We denote by $\mathcal{P}(L)$ the power set of L and for any function $f : L \to \mathbb{R}$, we use the convention that $\sum_{x\in\emptyset} f(x) = 0$ and $\prod_{x\in\emptyset} f(x) = 1$.

2.1 Weighted fault models

A weighted fault model specifies the desired behavior of a system by not only providing the correct system traces, but also giving the severity of the erroneous traces. Technically, a weighted fault model is a function f that assigns a non-negative error weight to each trace $\sigma \in L^*$, where L is a given action alphabet. If $f(\sigma) = 0$, then σ is correct behavior; if $f(\sigma) > 0$, then σ is incorrect and $f(\sigma)$ denotes the severity of that error (i.e. the higher $f(\sigma)$, the worse the error). We require the total error weight in f, i.e. $\sum_\sigma f(\sigma)$, to be finite and non-zero, so that we can measure coverage of a test suite relative to the total error weight.

Definition 1. *A* weighted fault model *over an action alphabet L is a function $f : L^* \to \mathbb{R}^{\geq 0}$ such that $0 < \sum_{\sigma\in L^*} f(\sigma) < \infty$. We sometimes refer to traces $\sigma \in L^*$ with $f(\sigma) > 0$ as* error traces *and traces with $f(\sigma) = 0$ as* correct traces.

2.2 Coverage measures

For the abstract set up in this section, we do not need to know what exactly a test looks like. We just need that a test is some set of traces and a test suite is a set of tests, i.e. some family of traces sets. Thus, our coverage measures can be applied in a test context where every test cases can be characterized as a traces set, viz. those traces that can occur when the tester executes the test. This is the case e.g. in TTCN[6], ioco test theory[12] and FSM testing[14].

Definition 2. *Let* $f : L^* \to \mathbb{R}^{\geq 0}$ *be a weighted fault model over* L*, let* $t \subseteq L^*$ *be a trace set and let* $T \subseteq \mathcal{P}(L^*)$ *be a collection of trace sets. We define*

- $abscov(t, f) = \sum_{\sigma \in t} f(\sigma)$ *and* $abscov(T, f) = abscov(\cup_{t \in T}\, t, f)$
- $totcov(f) = abscov(L^*, f)$
- $relcov(t, f) = \frac{abscov(t,f)}{totcov(f)}$ *and* $relcov(T, f) = \frac{abscov(T,f)}{totcov(f)}$

The coverage of a test suite T, w.r.t. a weighted fault model f, measures the total weight of the errors that can be detected by tests in T. The absolute coverage $abscov(T, f)$ simply accumulates the weights of all error traces in T. Note that each trace is counted only once, since one test case is enough to detect the presence of an error trace in an IUT. The relative coverage $relcov(T, f)$ yields the error weight in T as a fraction of the weight of all traces in T. Absolute (coverage) numbers have meaning when they are put in perspective of a maximum, or average. Then, we advocate that the relative coverage is a good measure of the quality of a test suite. Note that the requirement $0 < \sum_{\sigma \in L^*} f(\sigma) < \infty$ is needed to prevent division by 0 and division of ∞ by ∞.

Completeness of a test suite can easily be expressed in terms of coverage.

Definition 3. *A test suite* $T \subseteq \mathcal{P}(L^*)$ *is* complete *w.r.t. a weighted fault model* $f : L^* \to \mathbb{R}^{\geq 0}$ *if* $cov(T, f) = 1$.

The following proposition characterizes the complete test suites. Its proof follows immediately from the definitions.

Proposition 1. *Let* f *be a weighted fault model over* L*. Then a test suite* $T \subseteq \mathcal{P}(L^*)$ *is complete for* f *if and only if for all* $\sigma \in L^*$ *with* $f(\sigma) > 0$*, there exists* $t \in T$ *such that* $\sigma \in t$.

3 Labeled input-output transition systems

This section recalls some basic theory about test case derivation from labeled input-output transition systems, following ioco testing theory [12]. It prepares for the next section that treats an automaton-based formalism for specify weighted fault models.

Definition 4. *A* labeled input-output transition system (LTS) $\mathcal{A}$ *is a tuple* $\langle V, L, \Delta \rangle$*, where*

- V *is a finite set of states.*
- L *is a finite action signature. We assume that* $L = L^I \cup L^O$ *is partitioned into a set* L^I *of* input labels *(also called input actions or inputs) and a set* L^O *of* output labels L^O *(also called output actions or outputs). We denote elements of* L^I *by* $a?$ *and elements of* L^O *by* $a!$.

- $\Delta \subseteq V \times L \times V$ *is the transition relation. We require* Δ *to be deterministic, i.e. if* $(s, a, s'), (s, a, s'') \in \Delta$*, then* $s' = s''$*. The input successor transition relation* Δ^I *is the restriction of* Δ *to* $\Delta^I \subseteq V \times L^I \times V$ *and* Δ^O *is the restriction of* Δ *to* $\Delta^O \subseteq V \times L^O \times V$*. We write* $\Delta(s) = \{(a, s') \mid (s, a, s') \in \Delta\}$ *and similarly for* $\Delta^I(s)$ *and* $\Delta^O(s)$*. We denote by* $outdeg(s) = |\Delta^O(s)|$ *the outdegree of state* s*, i.e. the number of transitions leaving* s*.*

We denote the components of $\mathcal{A}$ *by* $V_{\mathcal{A}}$*,* $L_{\mathcal{A}}$*, and* $\Delta_{\mathcal{A}}$*. We omit the subscript* $\mathcal{A}$ *if it is clear from the context.*

We have asked that $\mathcal{A}$ is deterministic only for technical simplicity. This is not a real restriction, since we can always determinize $\mathcal{A}$. We can also incorporate quiescence, by adding a self loop $s \xrightarrow{\delta} s$ labeled with a special label δ to each quiescent state s, i.e. each s with $\Delta^O(s) = \emptyset$. Since quiescence is not preserved under determinization, we must first determinize and then add quiescence.

We introduce the usual language theoretic concepts for LTSs.

Definition 5. *Let* $\mathcal{A}$ *be a LTS, then*

- *A* path *in* $\mathcal{A}$ *is a finite sequence* $\pi = s_0, a_1, s_1, \ldots s_n$ *such that for all* $1 \leq i \leq n$*, we have* $(s_{i-1}, a_i, s_i) \in \Delta$*. We denote by* $paths(s_0)$ *the set of all paths that start from the state* $s_0 \in V$ *and by* $last(\pi) = s_n$ *the last state of* π*.*
- *The* trace *of* π*,* $trace(\pi)$*, is the sequence* $a_1, a_2 \ldots a_n$ *of actions occurring in* π*. We denote by* $traces(s)$ *the set of all traces that start from state* $s \in V$ *:* $\{trace(\pi) | \pi \in paths(s)\}$*, and by* $traces(\mathcal{A})$ *the set of all traces of* $\mathcal{A}$ *:* $\cup_{s \in V}\ traces(s)$*.*
- *We write* $s \xrightarrow{\sigma} s'$ *if* s' *can be reached from* s *via the trace* σ*, i.e. if there is a path* $\pi \in paths(s)$ *such that* $trace(\pi) = \sigma$ *and* $last(\pi) = s'$*. We write* $s \xrightarrow{\sigma}_k s'$ *if* $s \xrightarrow{\sigma} s'$ *and* $|\sigma| = k$*;* $s \longrightarrow_k s'$ *if* $s \xrightarrow{\sigma}_k s'$ *for some* σ*; and* $s \longrightarrow s'$ *if* $s \xrightarrow{\sigma} s'$ *for some* σ*.*

Test cases for LTSs are based on ioco test theory [12]. As in TTCN, ioco test cases are adaptive. That is, the next action to be performed (observe the IUT, stimulate the IUT or stop the test) may depend on the test history, that is, the trace observed so far. If, after a trace σ, the tester decides to stimulate the IUT with an input $a?$, then the new test history becomes $\sigma a?$; in case of an observation, the test accounts for all possible continuations $\sigma b!$ with $b! \in L^O$ an output action. Ioco theory requires that tests are "fail fast", i.e. stop after the discovery of the first failure, and never fail immediately after an input. If $\sigma \in traces(s)$, but $\sigma a? \notin traces(s)$, then the behavior after $\sigma a?$ is not specified in s, leaving room for implementation freedom. Formally, a test case consists of the set of all possible test histories obtained in this way.

Definition 6.
- *A* test case *(or test)* t *for a LTS* $\mathcal{A}$ *at state* $s \in V$ *is a finite, prefix-closed subset of* $traces(s)$ *such that*
  - *if* $\sigma a? \in t$*, then* $\sigma b \notin t$ *for any* $b \in L$ *with* $a? \neq b$
  - *if* $\sigma a! \in t$*, then* $\sigma b! \in t$ *for all* $b! \in L^O$
  - *if* $f(\sigma) > 0$*, then no proper suffix of* σ *is contained in* t

We denote the set of all tests for $\mathcal{A}$ by $\mathcal{T}(\mathcal{A})$.

- *The* length $|t|$ *of a test case t is the length of the longest trace in t. Thus, $|t| = \max_{\sigma \in t} |\sigma|$. We denote by $\mathcal{T}_k(\mathcal{A})$ the set of all test cases of length k.*

Remark 1. The current definitions allow as a legal test the set $t = \{\varepsilon\}$ containing only the empty sequence. This may seem odd, but does not yield contradictions in the theory. If $f(\varepsilon) = 0$ having this test case yields a pass and does not add any testing power. If $f(\varepsilon) > 0$, then t is the only test case for f, and it yields a fail.

Since a test is a set of traces, we can apply Definition 2 and speak of the (absolute, total and relative) coverage of a test case or a test suite, relative to a weighted fault model f. However, not all weighted fault models are consistent with the interpretation that traces of f represent correct system behavior, and that tests are fail fast and do not fail after an input.

Definition 7. *A weighted fault model $f : L^* \to \mathbb{R}^{\geq 0}$ is* consistent with *the LTS $\mathcal{A}$ at state $s \in V_\mathcal{A}$ if we have: $L = L_\mathcal{A}$, for all $\sigma \in L^*_\mathcal{A}$ and $a? \in L^I$*

- *If $\sigma \in traces(s)$, then $f(\sigma) = 0$.*
- *$f(\sigma a?) = 0$ (no failure occurs after an input).*
- *If $f(\sigma) > 0$ then $f(\sigma\rho) = 0$ for all $\rho \in L^+_\mathcal{A}$ (failures are counted only once).*

The following result states that the set containing all possible test cases has complete coverage.

Theorem 1. *The set of all test cases $\mathcal{T}(\mathcal{A})$ is complete for any weighted fault model f consistent with $\mathcal{A}$.*

Proof. For all $\sigma \in L$ with $f(\sigma) > 0$, we build a test $t \in \mathcal{T}(f)$ with $\sigma \in t$. Write $\sigma = a_1, a_2, \ldots, a_n$. For $1 \leq i \leq n$, define a set X_i by

$$X_i = \begin{cases} \{a_1 \ldots a_i\} & \text{if } a_i \in L^I \\ \{a_1 \ldots a_{i-1} b \mid b \in L^O\} & \text{if } a_i \in L^O \end{cases}$$

The set t is obtained by $\cup_{1 \leq i \leq n} X_i$. Clearly, t is a test containing σ.

4 Fault automata

Weighted fault models are infinite, semantic objects. This section introduces fault automata, which provide a syntactic format for specifying fault models. A fault automaton is a LTS $\mathcal{A}$ augmented with a state weight function r. The LTS $\mathcal{A}$ is the behavioral specification of the system, i.e. its traces represent the correct system behaviors. Hence, these traces will be assigned error weight 0; traces not in $\mathcal{A}$ are erroneous and get an error weight through r.

Definition 8. *A* fault automaton (FA) $\mathcal{F}$ *is a pair* $\langle \mathcal{A}, r \rangle$*, where* $\mathcal{A}$ *is a LTS and* $r : V \times L^O \to \mathbb{R}^{\geq 0}$*. We require that, if* $r(s, a!) > 0$*, then there is no* $a!$*-successor of* s *in* $\mathcal{F}$*, i.e. there is no* $s' \in V$ *such that* $(s, a!, s') \in \Delta$*. We define* $\overline{r} : V \to \mathbb{R}^{\geq 0}$ *as* $\overline{r}(s) = \sum_{a \in \Delta^O(s)} r(s, a)$*. Thus,* $\overline{r}$ *accumulates the weight of all the erroneous outputs in a state. We denote the components of* $\mathcal{F}$ *by* $\mathcal{A}_{\mathcal{F}}$ *and* $r_{\mathcal{F}}$ *and leave out the subscripts* $\mathcal{F}$ *if it is clear from the context. We lift all concepts (e.g. traces, paths,...) that have been defined for traces to FA.*

We wish to construct a fault model f from and FA $\mathcal{F}$, using r to assign weights to traces not in $\mathcal{F}$. If there is no outgoing $b!$-transition in s, then the idea is that, for a trace σ ending in s, the (incorrect) trace $\sigma b!$ gets weight $r(s, b!)$. However, doing so, the total error weight $totcov(f)$ could be infinite.

We consider two solutions to this problem. First, finite depth fault models (Section 4.1) consider, for a given $k \in \mathbb{N}$, only faults in traces of length k or smaller. Second, discounted weighted fault models (Section 4.2) obtain finite total coverage through discounting, while considering error weight in all traces. The solution presented here are only two potential solutions, there are many other ways to derive a weighted fault model from a fault automaton.

4.1 Finite depth weighted fault models

As said before, the finite depth model derives a weighted fault model from a FA $\mathcal{F}$, for a given $k \in \mathbb{N}$, by ignoring all traces of length larger than k, i.e. by putting their error weight to 0. For all other traces, the weight is obtained via the function r. If σ is a trace of $\mathcal{F}$ ending in s, but $\sigma b!$ is not a trace in $\mathcal{F}$, then $\sigma b!$ gets weight $r(s, b!)$.

Definition 9. *Given a FA* $\mathcal{F}$*, a state* $s \in V$*, and a number* $k \in \mathbb{N}$*, we define the function* $f_{(\mathcal{F},s,k)} : L^* \to \mathbb{R}$ *by*

$$f_{(\mathcal{F},s,k)}(\varepsilon) = 0$$

$$f_{(\mathcal{F},s,k)}(\sigma a) = \begin{cases} r(s', a) & \textit{if } s \xrightarrow{\sigma}_k s' \wedge a \in L^O \\ 0 & \textit{otherwise} \end{cases}$$

Note that this function is uniquely defined because $\mathcal{F}$ is deterministic, so that there is at most one s' with $s \xrightarrow{\sigma}_k s'$. Also, if $f_{(\mathcal{F},s,k)}(\sigma a) = r(s, a) > 0$, then $\sigma \in \textit{traces}(s)$, but $\sigma a \notin \textit{traces}(s)$.

Proposition 2. *Let* $\mathcal{F}$ *be a FA,* $s \in V$ *and* $k \in \mathbb{N}$ *and assume that there exists a state* s' *in* $\mathcal{F}$ *such that* $s \longrightarrow_k s'$ *and* $\overline{r}(s') > 0$*. Then* $f_{(\mathcal{F},s,k)}$ *is a fault model that is consistent with* $\mathcal{F}$*.*

4.2 Discounted weighted fault models

While finite depth weighted fault models achieve finite total coverage by considering finitely many traces, discounted weighted fault models take into account the error weight of all traces. To do so, only finitely many traces may have weight greater than ϵ, for any $\epsilon > 0$. One way to do this is by discounting: lowering the weight of a trace proportional to its length. The rationale behind this is that errors in the near future are worse than errors in the far future, and hence, the latter should have a higher error weights.

In its basic form, this means that the weighted fault model f for an FA $\mathcal{F}$ sets the weight of a trace $\sigma a!$ to $\alpha^{|\sigma|} r(s, a!)$, for some discount factor $\alpha \in (0, 1)$. If we take α small enough, to be precise, smaller than $\frac{1}{d}$, where d is the branching degree of $\mathcal{F}$ (i.e. $d = \max_{s \in V} \mathit{outdeg}(s)$), one can easily show that $\sum_{\sigma \in L^*} f(\sigma) < \infty$. Indeed, since there are at most d^k traces of length k in $\mathcal{F}$, and writing $M = \max_{s,a} r(s, a)$ and assuming that $\alpha d < 1$, it follows that

$$\sum_{\sigma \in L^*} f(\sigma) = \sum_{k \in \mathbb{N}} \sum_{\sigma \in L^k} \alpha^k r(s, a) \leq \sum_{k \in \mathbb{N}} \sum_{\sigma \in L^k} \alpha^k M \leq \sum_{k \in \mathbb{N}} d^k \alpha^k M = \frac{M}{1 - d\alpha}$$

To obtain more flexibility, we allow the discount to vary per transition. That is, we work with a discount function $\alpha : V \times L \times V \to \mathbb{R}^{\geq 0}$, that assigns a positive weight to each transition of $\mathcal{F}$. Then we discount the trace $a_1, \ldots, a_k$ obtained from the path $s_0, a_1, s_1, \ldots s_k$ by $\alpha(s_0, a_1, s_1)\alpha(s_1, a_2, s_2), \cdots, \alpha(s_{k-1}, a_k, s_k)$. The requirement that α is small enough now becomes:

$$\sum_{a \in L, s' \in V} \alpha(s, a, s') < 1$$

for each s. We can even be more flexible and in the sum above, we do not range over states in which all paths are finite, because we obtain finite coverage in these states anyway. Thus, if $\mathit{Inf}_{\mathcal{F}}$ is the set of all states in $\mathcal{F}$ with at least one outgoing infinite path, we require for all states s:

$$\sum_{a \in L, s' \in \mathit{Inf}_{\mathcal{F}}} \alpha(s, a, s') < 1$$

Definition 10. *Let $\mathcal{F}$ be a FA. Then a* discount function *for $\mathcal{F}$ is a function $\alpha : V_{\mathcal{F}} \times L_{\mathcal{F}} \times V_{\mathcal{F}} \to \mathbb{R}^{\geq 0}$ such that*

- *For all $s, s' \in V$, and $a \in L$ we have $\alpha(s, a, s') = 0$ iff $(s, a, s') \notin \Delta$.*
- *For all $s \in V_{\mathcal{F}}$, we have:*

$$\textstyle\sum_{a \in L, s' \in \mathit{Inf}} \alpha(s, a, s') < 1$$

Definition 11. *Let α be a discount function for the FA $\mathcal{F}$. Given a path $\pi = s_0, a_1, \ldots s_k$ in $\mathcal{F}$, we define*

$$\alpha(\pi) = \textstyle\prod_{i=1}^{n} \alpha(s_{i-1}, a_i, s_i)$$

Definition 12. *Let be given a FA $\mathcal{F}$, a state $s \in V$, and a discount function α for $\mathcal{F}$. We define the function $f_{(\mathcal{F},s,\alpha)} : L^* \to \mathbb{R}^{\geq 0}$ by*

$$f_{(\mathcal{F},s,\alpha)}(\varepsilon) = 0$$

$$f_{(\mathcal{F},s,\alpha)}(\sigma a) = \begin{cases} \alpha(\pi) \cdot r(s', a) & \text{if } s \xrightarrow{\sigma} s' \wedge a \in L^O \wedge \mathit{trace}(\pi) = \sigma \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathcal{F}$ is deterministic, there is at most one π with $trace(\pi) = \sigma$, so the function above is uniquely defined.

Definition 13. *A FA $\mathcal{F} = \langle \mathcal{A}, r \rangle$ has a* fair *weight assignment r if for all $s \in Inf_{\mathcal{F}}$ there exists an $s' \in V$ that is* reachable *from s with $\overline{r}(s') > 0$.*

Proposition 3. *Let $\mathcal{F}$ be a FA, $s \in V$ be a state and α be a discount function for $\mathcal{F}$. If $\mathcal{F}$ has fair weight assignment, then $f_{(\mathcal{F},s,\alpha)}$ is a weighted fault model that is consistent with $\mathcal{F}$.*

Remark 2. We like to stress that the finite depth and discounted models are just two examples for deriving weighted fault models from fault automata, but there are many more possibilities. For instance, one may combine the two and not discount the weights of traces of length less than some k or less, and only discount traces longer than k. Alternatively, one may let the discount factor depend on the length of the trace, etcetera. We claim that the methods and algorithms we present in this paper can easily adapted for weighted fault models with such variations.

4.3 Calibration

Discounting weighs errors in short traces more than in long traces. Thus, if we discount too much, we may obtain very high test coverage just with a few short test cases. The calibration result (Theorem 2) presented in this section shows that, in any FA $\mathcal{F}$ and any $\epsilon > 0$, we can choose the discounting function in such a way that test cases of a given length k or longer are needed to achieve test coverage higher than a coverage bound $1-\epsilon$. That is, we show that for any given k and ϵ, there exists a discount function α such that the relative coverage of all test cases of length k or shorter is less than ϵ. This means that, to get coverage higher than $1 - \epsilon$, one needs test cases longer than k.

Theorem 2. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be a FA with fair weight assignment. Then there exists a family of discount functions α_u for $\mathcal{F}$ such that for all $k \in \mathbb{N}$ and states $s \in V$*

$$\lim_{u \to 0} cov(\mathcal{T}_k(f_{(\mathcal{F},s,\alpha_u)}), f_{(\mathcal{F},s,\alpha_u)}) = 0$$

The rest of this section is concerned with the proof from the theorem above.

Given a FA $\mathcal{F} = \langle \mathcal{A}, r \rangle$, we write $A_{\mathcal{F}}$ for the multi-adjacency matrix of $\mathcal{A}$, containing at position (s, s') the number of edges between s and s', i.e. $(A_{\mathcal{F}})_{ss'} = \sum_{a:(s,a,s') \in \Delta} 1$. If α is a discount function for $\mathcal{F}$, then $A^{\alpha}_{\mathcal{F}}$ is a weighted version of $A_{\mathcal{F}}$, i.e. $(A^{\alpha}_{\mathcal{F}})_{ss'} = \sum_{a \in L} \alpha(s, a, s')$. We omit the subscript $\mathcal{F}$ if it is clear from the context.

Definition 14. *Given an FA $\mathcal{F}$, we define a discount function $\alpha_u : V \times L \times V \to (0, 1)$ by*

$$\alpha_u(s, a, s') = \begin{cases} \frac{(1-\epsilon)}{|OutInf\,(s)|} & \text{if } (s, a, s') \in \Delta \text{ and } s' \in Inf_{\mathcal{F}} \\ > 0 & \text{if } (s, a, s') \in \Delta \text{ and } s' \in V \setminus Inf_{\mathcal{F}} \\ 0 & \text{otherwise} \end{cases}$$

Where $OutInf(s) = \{(a, s') \in \Delta(s) | s' \in \mathit{Inf}_{\mathcal{F}}\}$. *We usually write* A_u *for the matrix* A^{α_u}.

Definition 15. *Given a FA* $\mathcal{F}$, *we define the vector* $\mathbf{1}_{\mathit{Inf}}$ *indexed by* $s \in V$ *by*

$$\mathbf{1}_{\mathit{Inf}}(s) = \begin{cases} 1 & \text{if } s \in \mathit{Inf}_{\mathcal{F}} \\ 0 & \text{otherwise} \end{cases}$$

Proposition 4. $\mathbf{1}_{\mathit{Inf}}$ *is an eigenvector of* A_u *with eigenvalue* $1 - \epsilon$, *i.e.*

$$A_u \cdot \mathbf{1}_{\mathit{Inf}} = (1 - \epsilon) \cdot \mathbf{1}_{\mathit{Inf}}$$

Proof. First, consider $s \in \mathit{Inf}_{\mathcal{F}}$:

$$\begin{aligned}
(A_u \cdot \mathbf{1}_{\mathit{Inf}})_s &= \sum_{s' \in V} (A_u)_{ss'} \cdot \mathbf{1}_{\mathit{Inf}}(s') \\
&= \sum_{s' \in \mathit{Inf}} (A_u)_{ss'} \cdot \mathbf{1}_{\mathit{Inf}}(s') \\
&= \sum_{s' \in \mathit{Inf}} \sum_{a \in L} \alpha_u(s, a, s') \\
&= \sum_{(a,s') \in \mathit{OutInf}(s)} \frac{(1-\epsilon)}{|\mathit{OutInf}(s)|} \\
&= |\mathit{OutInf}(s)| \cdot \frac{(1-\epsilon)}{|\mathit{OutInf}(s)|} \\
&= 1 - \epsilon
\end{aligned}$$

For $s \in V \backslash \mathit{Inf}_{\mathcal{F}}$ we get, using the closure property of Inf:

$$\begin{aligned}
(A_u \cdot \mathbf{1}_{\mathit{Inf}})_s &= \sum_{s' \in V} (A_u)_{ss'} \cdot \mathbf{1}_{\mathit{Inf}}(s') \\
&= \sum_{s' \in \mathit{Inf}} (A_u)_{ss'} \cdot \mathbf{1}_{\mathit{Inf}}(s') \\
&= \sum_{s' \in \mathit{Inf}} \sum_{a \in L} \alpha_u(s, a, s') \\
&= \sum_{s' \in \mathit{Inf}} \sum_{a \in L} 0 \\
&= 0
\end{aligned}$$

Corollary 1. $(A_u)^n \cdot \mathbf{1}_{\mathit{Inf}} = (1 - \epsilon)^n \cdot \mathbf{1}_{\mathit{Inf}}$.

Proof. By induction on n.

Proposition 5. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be a FA with a fair weight assignment r then there exists an $N \geq 0$ such that $(\sum_{i=0}^{N} A_u^i \cdot \overline{r})_s > 0$ for every $s \in \mathit{Inf}_{\mathcal{F}}$.*

Proof. Note that $(A_u^i)_{ss'} > 0$ implies that s' can be reached from s in i transitions. As $\mathcal{F}$ is based on a FA every state s is at most $|V| - 1$ transitions removed any of the states s' that can be reached from it, so that there is an $N < |V|$ with $(\sum_{i=0}^{N} A_u^i)_{ss'} > 0$ for any pair of such $s, s' \in V$. By the definition of fair weight assignment all states $s \in \mathit{Inf}_{\mathcal{F}}$ can reach an $s' \in V$ with $\overline{r}(s') > 0$. Thus we get $(\sum_{i=0}^{N} A_u^i \cdot \overline{r})_s = \sum_{s' \in V} (\sum_{i=0}^{N} A_u^i)_{ss'} \cdot \overline{r}(s') > 0$.

Lemma 1. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be a FA with fair weight assignment. Then for every $s \in \mathit{Inf}_{\mathcal{F}}$*

$$\lim_{u \to 0} \mathit{cov}(\mathcal{T}_k(\mathcal{F}, \alpha_u, s), (\mathcal{F}, \alpha_u, s)) = 0$$

Proof. We first observe that

$$\mathit{relcov}(\mathcal{T}_k(\mathcal{F}, \alpha_u, s), (\mathcal{F}, \alpha_u, s)) = \frac{\mathit{abscov}(\mathcal{T}_k(\mathcal{F}, \alpha_u, s), (\mathcal{F}, \alpha_u, s))}{\mathit{totcov}((\mathcal{F}, \alpha_u, s))}$$

As $\mathit{abscov}(\mathcal{T}_k(\mathcal{F}, \alpha_u, s), (\mathcal{F}, \alpha_u, s))$ is always finite, it suffices to show that $\lim_{u \to 0} \mathit{totcov}(\mathcal{F}, \alpha_u, s) = \infty$.

This can be shown as follows:

$$\begin{aligned}
\mathit{totcov}(\mathcal{F}, \alpha_u, s) &= (\sum_{i=0}^{\infty} A_u^i \cdot \overline{r})_s \\
&= ((\sum_{i=0}^{N} A_u^i + \sum_{i=N+1}^{\infty} A_u^i) \cdot \overline{r})_s \\
&= ((\sum_{i=0}^{N} A_u^i + A_u^{N+1} \cdot (\sum_{i=0}^{\infty} A_u^i)) \cdot \overline{r})_s \\
&= (\sum_{j=0}^{\infty} A_u^{j(N+1)} \cdot (\sum_{i=0}^{N} A_u^i) \cdot \overline{r})_s \\
&\text{and according to the above proposition} \\
&\text{for some } \overline{r}' \text{ with } \overline{r}'(s) > 0 \text{ for } s \in \mathit{Inf}_{\mathcal{F}}
\end{aligned}$$

$$= (\sum_{j=0}^{\infty} A_u^{j(N+1)} \cdot \overline{r}')_s$$

defining $r_{min} = \min_{s \in Inf} \overline{r}'(s)$ we get

$$> r_{min} \cdot (\sum_{j=0}^{\infty} A_u^{j(N+1)} \cdot \mathbf{1}_{Inf})_s$$

$$= r_{min} \cdot (\sum_{j=0}^{\infty} (1-\epsilon)^{j(N+1)} \cdot \mathbf{1}_{Inf})_s$$

$$= \frac{r_{min}}{(1-(1-\epsilon)^{N+1})}$$

As $1-(1-\epsilon)^{N+1}$ is of the order $\mathcal{O}(\epsilon)$ we get $\lim_{u \to 0} (\sum_{i=0}^{\infty} A_u^i \cdot \overline{r})_s = \infty$.

5 Algorithms

5.1 Absolute coverage in a test suite

To make the notation simpler for a test t and an action a we write at for $\{a\sigma \mid \forall \sigma \in t\}$. Moreover if t' is also a test, then $t + t' = \{\sigma \mid \forall \sigma \in t\} \cup \{\sigma' \mid \forall \sigma' \in t'\}$. In this way we can write a test as: $t = \epsilon$ or $t = at_1$ in case a is an input or $t = b_1 t_1 + \cdots + b_n t_n$ when $b_1, \cdots, b_n$ are the output actions of the system. We called super-test (Stest) in case $t' = a_1 t'_1 + \cdots + a_k t'_k + b_1 t''_1 + \cdots + b_n t''_n$ where a_i are inputs and b_i are all the outputs.

Given an FA $\mathcal{F}$, a discounting function α for $\mathcal{F}$ and a test suite $T = \{t_1, \cdots, t_k\}$. To compute the absolute coverage of T, using Definition 2, we have to compute:$abscov(T, \mathcal{F}) = abscov(\cup_{t \in T} t, \mathcal{F})$. Then, we have to compute the union and then compute the absolute coverage of the union. To do the union we use the merge function form a test t and a Stest t' to a Stest.

Merge set of tests. Given a set of test $\{t_1, \cdots, t_k\}$ merge is a function mg: Stest $\times$ test $\rightarrow$ Stest. Let t' be a Stest. (Note that any test is a Stest.) Let t be a test, then $t = \epsilon$ or $t = at_1$ or $t = b_1 t'_1 + \cdots + b_n t'_n$

$$mg(t', t) =$$

$$\begin{cases} a_1 t'_1 + \cdots + a_j mg(t'_j, t_1) + \cdots + a_k t'_k + b_1 t''_1 + \cdots + b_n t''_n & \text{if } t = at_1 \wedge a = a_j \\ a_1 t'_1 + \cdots + a_k t'_k + b_1 mg(t''_1, t_1) + \cdots + b_n mg(t''_n, t_n) & \text{if } t = b_1 t'_1 + \cdots + b_n t'_n \\ t' + t & \text{otherwise} \end{cases}$$

Now we can compute the absolute coverage of a Stest, given a state $s \in V$, then

$$tc(\varepsilon, s) = 0$$

$$tc(t, s) = \sum_{i=1}^{n} aux(a_i t_i, s)$$

$$aux(a_i t_i, s) = \begin{cases} \alpha(s, a_i, \delta(s, a_i)) tc(t_i, \delta(s, a_i)) & \text{if } a_i \in \delta(s) \\ r(a_i, s) & \text{otherwise} \end{cases}$$

Then to compute the absolute coverage of a Stest t it is enough with $tc(t, s_0)$.

Theorem 3. *Given a FA $\mathcal{F}$, a state $s \in V$, a number $k \in \mathbb{N}$ and T a set of test, then*

- $abscov(T, f_{(\mathcal{F},s,\alpha)}) = tc(mg(T), s)$
- *If* $k > \max_{t \in T} |t|$ *and* $\alpha(s, a, s') = 1$ *then*
$$abscov(T, f_{(\mathcal{F},s,k)}) = tc(mg(T), s)$$

5.2 Total coverage algorithms

Total coverage in discounted FA. Given a FA $\mathcal{F}$, a state $s \in V$ and a discounting function α for $\mathcal{F}$, we desire to calculate $totcov(f_{(\mathcal{F},s,\alpha)}) = \sum_{\sigma \in L^*} f_{(\mathcal{F},s,\alpha)}(\sigma)$. The basic idea behind the computation method is that the function $tw : V \to [0, 1]$ given by $s \mapsto totcov(f_{(\mathcal{F},s,\alpha)})$ satisfies the following set of equations.

$$tw(s) = \overline{r}(s) + \sum_{a \in L, s' \in V} \alpha(s, a, s') tw(s') \tag{1}$$

$$= \overline{r}(s) + \sum_{s' \in V} A^{\alpha}_{s,s'} \cdot tw(s') \tag{*}$$

These equations express that the total coverage in state s equals the weight $\overline{r}(s)$ of all immediate errors in s, plus the weights in all successors s' in s, discounted by:

$$\textstyle\sum_{a \in L} \alpha(s, a, s')$$

Proof.

$$\begin{aligned} tw(s) &= \sum_{\sigma \in L^*} f_{(\mathcal{F},\alpha,s)}(\sigma) \\ &= f_{(\mathcal{F},\alpha,s)}(\varepsilon) + \sum_{a \in \Delta(s), \sigma \in L^*} f_{(\mathcal{F},\alpha,s)}(\sigma) + \sum_{a \notin \Delta(s), \sigma \in L^*} f_{(\mathcal{F},\alpha,s)}(\sigma) \quad \text{(Proposition ??)} \\ &= 0 + \sum_{a \in \Delta(s), \sigma \in L^*} \alpha(s, a, s') f_{(\mathcal{F},\alpha,s')}(\sigma) + \sum_{a \notin \Delta(s)} r(s, a) \\ &= \sum_{a \in L} \alpha(s, a, s') tw(s) + \overline{r}(s) \end{aligned}$$

In matrix-vector notation, we obtain:

$$tw = \overline{r} + A^{\alpha} tw$$

Since the matrix $I - A^{\alpha}$ is invertible (cf. [3]), we obtain the following result. In particular, tw is the unique solution of the equations (*) above.

Theorem 4. *Let $\mathcal{F}$ be a FA, and α be a discount function for $\mathcal{F}$, then*

$$tw = (I - A^{\alpha})^{-1} \cdot \overline{r}$$

Complexity. The complexity of the method above is dominated by matrix inversion, which can be computed in $O(|V|^3)$ with Gaussian elimination, $O(|V|^{log_2 7})$ with Strassen's method or even faster with more sophisticated techniques.

Proposition 6. *The matrix $I - A^\alpha$ is invertible.*

Proof. By reordering the states we can obtain $\mathit{Inf}_{\mathcal{F}} = \{s_1, \ldots, s_{n_1}\}$ and $V_{\mathcal{F}} \backslash \mathit{Inf}_{\mathcal{F}} = \{s_{n_1+1}, \ldots, s_{n_1+n_2}\}$ with $n_1 + n_2 = n = |V_{\mathcal{F}}|$. Without loss of generality we may therefore assume that A^α is of the form

$$\begin{pmatrix} B & C \\ 0 & D \end{pmatrix}$$

with B the $n_1 \times n_1$ matrix that is the restriction of A^α to $\mathit{Inf}_{\mathcal{F}}$, and D the restriction of A^α to $V_{\mathcal{F}} \backslash \mathit{Inf}_{\mathcal{F}}$. It follows that $I_{(n)} - A^\alpha$ is invertible iff $I_{(n_1)} - B$ and $I_{(n_2)} - D$ are invertible.

We first show that $\|Bv\|_\infty < \|v\|_\infty$ for all $v \neq 0$, where $\|v\|_\infty = max_i(v_i)$ denotes the supremum norm of v.

Assume $v \neq 0$ and consider the i^{th} component $(Bv)_i$ of the vector Bv.

$$\begin{aligned} (Bv)_i &= \sum_{j \leq n_1} B_{ij} v_j \\ &\leq \sum_{j \leq n_1} B_{ij} \|v\|_\infty \\ &= \|v\|_\infty \cdot \sum_{(j,a) \in \mathit{OutInf}(i)} \alpha(i, a, j) \qquad \text{(Def of discount function)} \\ &< \|v\|_\infty \end{aligned}$$

Hence, $\|Bv\|_\infty < \|v\|_\infty$. Therefore $Bv \neq v$, so $(I - B)v \neq 0$ for $v \neq 0$, which yields that $I - B$ is invertible.

Without loss of generality we can also assume that the states have been numbered such that for $i, j \in V_{\mathcal{F}} \backslash \mathit{Inf}_{\mathcal{F}}$ $(i, a, j) \in \delta_{\mathcal{F}}$ implies $i < j$. It follows that $D_{ij} = 0$ for all $1 < j \leq i < n_2$, and that $(I - D)_{ij} = 0$ for all $1 < j < i < n_2$ with $(I - D)_{ii} = 1$ for all $1 < i < n_2$. We can conclude that $\det(I - D) = 1 \neq 0$, and thus that $I - D$ is invertible.

Remark 3. It is easy to see that $v \mapsto \overline{r} + Bv$ is a contraction on $\mathit{Inf}_{\mathcal{F}} \to \mathbb{R}^{\geq 0}$. While it is known that contractions have unique fixpoints on compact spaces, we cannot immediately apply this argument here, since $\mathit{Inf}_{\mathcal{F}} \to \mathbb{R}^{\geq 0}$ is not compact.

Total coverage in finite depth FA. Given a FA $\mathcal{F}$, a state $s \in V$ and a depth $k \in \mathbb{N}$, we desire to compute $totcov(f_{(\mathcal{F},s,k)}) = \sum_{\sigma \in L^*} f_{(\mathcal{F},s,k)}(\sigma)$. The basic idea behind the computation method is that the function $tw_k : V \to [0, 1]$

given by $s \mapsto totcov(f_{(\mathcal{F},s,k)})$ satisfies the following recursive equations.

$$\begin{aligned} tw_0(s) &= 0 \\ tw_{k+1}(s) &= \overline{r}(s) + \sum_{(a,s') \in \Delta(s)} tw_k(s') \\ &= \overline{r}(s) + \sum_{a \in L, s' \in V} A_{s,s'} \cdot tw_n(s') \end{aligned}$$

The correctness of these equations follows from Proposition **??**. Or, in matrix-vector notation we have

$$tw_0 = 0 \tag{2}$$
$$tw_{k+1} = \overline{r} + A tw_k \tag{**}$$

Thus, we have the following.

Theorem 5. *Let be given a FA $\mathcal{F}$, a state $s \in V$ and a number $k \in \mathbb{N}$, then*

$$tw_k = \sum_{i=0}^{k-1} A^i \overline{r}$$

Complexity. By using Theorem 5 with sparse matrix multiplication, or by iterating the equations just above it, tw_k can be computed in time $O(k \cdot |\Delta| + |V|)$.

Remark 4. A similar method to the one above can be used to compute the weight of all tests of length k in the discounted fault model, i.e. $abscov(T_k, f_{(\mathcal{F},s,\alpha)})$, where T_k is the set of all tests of length k in $\mathcal{F}$. Writing $twd_k(s) = abscov(T_k, f_{(\mathcal{F},s,\alpha)})$, the recursive equations become

$$\begin{aligned} twd_0(s) &= 0 \\ twd_{k+1}(s) &= \overline{r}(s) + \sum_{a \in L, s' \in V} tw_k(s') \\ &= \overline{r}(s) + \sum_{a \in L, s' \in V} A^{\alpha}_{s,s'} \cdot twd_k(s') \end{aligned}$$

and the analogon of Theorem 5 becomes

$$\begin{aligned} twd_k &= \sum_{i=0}^{k-1} (A_\alpha)^i \overline{r} \\ &= (I - A^\alpha)^{-1} \cdot (I - (A^\alpha)^k) \cdot \overline{r} \end{aligned}$$

The latter equality holds because $I - A^\alpha$ is invertible. Thus, the computing twd_k requires one matrix inversion and, using the power method, $\log_2(k)$ matrix multiplications, yielding time complexity in $O(|V|^{\log_2 7} + |V|^{\log_2(k)})$ with Strassen's method. If $(I - A^\alpha)$ can be put in diagonal form, the problem can be solved in $O(|V|^3 + \log_2 n)$. These tricks cannot be applied in the finite depth model, because $I - A$ is not invertible. Since A has row sum 1, we have for the vector $\mathbf{1}$ whose entries are all equal to 1 that $A\mathbf{1} = \mathbf{1}$. Hence, $\mathbf{1}$ is in the kernel of $I - A$, so $I - A$ is not invertible.

5.3 Optimization

Optimal coverage in a single test case. This section presents an algorithm to compute, for a given FA $\mathcal{F}$, and a length k, the best test case with length k, that is, the one with highest coverage. We treat the finite depth and discounted model at once by putting, in the finite depth model $\alpha(s,a,s')=1$ if (s,a,s') is a transition in Δ and having $\alpha(s,a,s')=0$ otherwise. We call a function α that is either obtained from a finite depth model in this way, or that is a discount function, an *extended discount function*.

The optimization method is again based on recursive equations. We write $tcopt_k(s) = \max_{t\in\mathcal{T}_k}\{abscov(t,s)\}$. Consider a test case of length $k+1$ that in state s applies an input $a?$ and in the successor state s' applies the optimal test of length k. The (absolute) coverage of this test case is $\alpha(s,a?,s')\cdot tcopt_k(s')$. The best coverage that we can obtain by stimulating the IUT is given by $\max_{(a',s')\in\Delta^I(s)}\alpha(s,a?,s')\cdot tcopt_k(s')$.

Now, consider the test case of length $k+1$ that in state s observes the IUT and in each successor state s' applies the optimal test of length k. The coverage of this test case is $\overline{r}(s)+\sum_{(b!,s')\in\Delta^O(s)}\alpha(s,b!,s')\cdot tcopt_k(s')$. The optimal test $tcopt(s)$ of length $k+1$ is obtained from by $tcopt_k$ by selecting from these options (i.e. inputing an action $a?$ or observing) the one with the highest coverage. Thus, we have the following result.

Theorem 6. *Let be given a FA $\mathcal{F}$, an extended discount function α, and test length $k\in\mathbb{N}$. Then $tcopt_k$ satisfies the following recursive equations.*

$$\begin{aligned}
tcopt_0(s) &= 0\\
tcopt_{k+1}(s) &=\\
\max\Big(\overline{r}(s) + &\sum_{(b!,s')\in\Delta^O(s)}\alpha(s,b!,s')tcopt_k(s'),\ \max_{(a?,s')\in\Delta^I(s)}\alpha(s,a?,s')tcopt_k(s')\Big)
\end{aligned}$$

The proof of this theorem follows from Proposition 7.

Complexity. Based on Theorem 6, we can compute $tcopt_k$ in time $O(k\cdot(|V|+|\Delta|))$.

Proposition 7.
- *Let s be a state, let $(a?,s')\in\Delta^I(s)$, and let t' be a test case in states s'. Write t for the test case $t=\{a?\sigma|\sigma\in t'\}$. Then*

$$abscov(t,s)=\alpha(s,a,s')\cdot abscov(t',s')$$

- *Let s be a state and $\Delta^O(s)=\{(b_1!,s_1),(b_2!,s_2)\ldots(b_n!,s_n)\}$, where the $b_i!$'s are all distinct. Let $t_1,t_2,\ldots t_n$ test cases in states $s_1\ldots s_n$ respectively. Write t for the test case $t=\{b_i!\sigma \mid \sigma\in t_i\}$. Then*

$$abscov(t,s)=\overline{r}(s)+\sum_{i=1}^{n}\alpha(s,b_i!,s_i)\cdot abscov(t_i,s_i)$$

Shortest test case with high coverage. We can use the above method not only to compute the test case of a fixed length k with optimal coverage, but also to derive the shortest test case with coverage higher than a given bound c. That is, we iterate the equations in Theorem 6 and stop as soon as we achieve coverage higher than c, i.e. at the first n with $tcopt_k(s) > c$.

We have to take care that the bound c is not too high, i.e. higher than what is achievable with a single test case. In the finite depth model, this is easy: if the test length is the same as c then we can stop, since this is the longest test we can have. In the discounted model, however, we have to ensure that c is strictly smaller than the supremum of the coverage of all tests in single test case.

Let $stw(s) = supp_{t \in \mathcal{T}} abscov(t, s)$, i.e. the maximal absolute weight of a single test case. Then stw is again characterized by a set of equations.

Theorem 7. *Let $\mathcal{F}$ be a FA, and α be a discount function for $\mathcal{F}$. Then stw is the unique solution of the following set of equations.*

$$stw(s) =$$
$$\max\left(\max_{(a?,s')\in\Delta^I(s)} \alpha(s,a?,s')\cdot stw(s'), \overline{r}(s) + \sum_{(b!,s')\in\Delta^O(s)} \alpha(s,b!,s')\cdot stw(s')\right)$$

The solution of these equations can be found by linear programming (LP).

Theorem 8. *Let $\mathcal{F}$ be a FA, and α be a discount function. Then stw is the optimal solution of the following LP problem.*

$$\begin{aligned}
&\textit{minimize} \sum_{s\in V} stw(s) \textit{ subject to} && \\
&stw(s) \geq \alpha(s,a?,s')\cdot stw(s'), && (a?,s')\in\Delta^I(s)\\
&stw(s) \geq r(s) + \sum_{(b!,s')\in\Delta^O(s)} \alpha(s,b!,s')\cdot stw(s') && s\in V
\end{aligned}$$

Complexity. The above LP problem contains $|V|$ variables and $|V| + |\Delta^I|$ inequalities. Thus, solving this problem is polynomial in $|V|$, $|V| + |\Delta^I|$ and the length of the binary encoding of the coefficients [11]. In practice, the exponential time simplex method outperforms existing polynomial time algorithms.

Best coverage test suites Just as we can ask for the best test of length k, we can also derive the best n tests of length k. The idea is as follows.

We write $tcoptk_k(s) = \underset{t_1,t_2,\ldots t_n\in\mathcal{T}_n}{maxn} \{abscov(t,s)\}$, for the list $[l_1, l_2, \ldots l_n]$, where l_i the coverage of the i^{th} best test of length k. We characterize $tcoptk_k$ recursively.

Assume that all input actions are given by $a_1, a_2, \ldots a_m$. Consider a test suite $T = \{t_1^1, t_1^2, \ldots t_1^n, t_2^1, t_2^2, \ldots t_2^n, \ldots t_m^1, t_m^2, \ldots t_m^n\}$ consisting of km tests, which all have length $k+1$ and start in state s. Assume that each test t_i^j applies input $a_i?$, leading successor state s_i'. Let $\{t_i^1, t_i^2, \ldots t_1^n\}$ be the best n tests at state i. Then the best test suite at s that can be achieved by stimulating the IUT is obtained by picking the best n tests from T.

Assume that all output actions are given by $\{b_1, b_2, \dots b_l\}$. Now, consider the test suite $T = \{t_1, t_2, \dots t_l\}$ consisting of l tests, of length $k+1$ starting in state s. Assume that each test t_i observes the IUT. If action b^j occurs, then successor state s'_j is reached. applies the optimal test of length k. The coverage of this test case is $\overline{r}(s) + \sum_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \cdot tcopt_k(s')$. The optimal test $tcopt(s)$ of length $k+1$ is obtained from by $tcopt_k$ by selecting from these options (i.e. inputing an action $a?$ or observing) the one with the highest coverage.

We find n test cases of length k with maximal coverage.

If we observe, the best n test cases are $sumlist[]$

Theorem 9. *Let be given a FA $\mathcal{F}$, a discount function α for $\mathcal{F}$, a test length $k \in \mathbb{N}$, and a number $n \in \mathbb{N}$. Then tw_k satisfies the following equations.*

$$\begin{aligned} v_0(s) =& [0, 0, .., 0] \\ v_{j+1}(s) =& maxn\{ \big[\alpha(s, a, s') \cdot v \mid (a?, s') \in \Delta^I(s), v \leftarrow v_j(s')\big] ++ \\ & r(s) \oplus sumlist_{(b!,s') \in \Delta^O(s)} \alpha(s, b!, s') \otimes v_j(s')\} \qquad (j < k) \end{aligned}$$

Here, $x \oplus l$ adds the number $x \in \mathbb{R}$ to each element of the list l, i.e., $x \oplus [e_1, e_2, \dots e_n] = [e_1 + x, e_2 + x, \dots e_n + x]$. Similarly, $x \otimes l$ multiplies each list element with x. The operator $sumlist_i l_i$ yields the point wise summation of all the lists l_i. Thus, for $l_i = [e^i_1, e^i_2, \dots e^i_n]$, we have. $sumlist_i l_i = [\sum_i e^i_1, \sum_i e^i_2, \dots \sum_i e^i_n]$. Note that all the lists to which we apply this operator length k.

Here $maxn$ yields the n maximal elements in a list. By keeping the lists sorted (largest element first) we can efficiently implement the algorithm. To do so, it suffices that $maxn$ returns a sorted list and that the list that $sumlist$ also preserves the order.

Algorithm 16 *(Variation on the theme) Rather than computing in algorithm 5.3 the best test case of are fixed length k, we can compute the best test case with coverage c, for some $c < stw(s)$. That is, we compute v_0, v_1, $v_2 \dots v_k$, until we find $v_k(s) \leq c$.*

6 Application: a chat protocol

As a practical example we present the chat protocol (also know as conference protocol [1]). The protocol is specified as the chat service, the protocol data, the underlying service and the behavior of the protocol entities.

The protocol data units describes the format of the data units that are used by the protocol entities to communicate with peer entities, the underlying service describes the service of the underlying communication medium through which these data units have to be communicated between peer entities and the behavior of the protocol entities. Details of all this services can be found in [1]. The service provided by a chat protocol, called the chat service is explained as follows.

The chat service provides a multi-cast service to users participating in a chat. A chat is a group of users that can exchange messages. Every user in a chat can send messages to all other chat partners participating in that chat, and it can receive messages from every other participant. The participants in a chat can change dynamically, as the chat service allows its users to join and leave a chat. Different chats can exist at the same time, but each user can only participate in at most one chat at a time. The chat service has the following service primitives (called CSPs), which can be performed at the chat service access points (CSAPs):

- join: a user joins a named chat and defines its user title in this chat; the user title identifies a user in a chat;
- datareq: a user sends a message to all other users participating in the chat;
- dataind: a user receives a message from another user participating in the chat;
- leave: a user leaves the chat; since a user can only participate in one chat at a time, there is no need to identify the chat in this primitive.

The service primitives join and leave are used for chat control. The service primitives datareq and dataind are used for data transfer. Initially, a user is only allowed to perform a join to a chat. After this, the user is allowed to send messages, by performing datareq's, or to receive messages, by performing dataind's. In order to stop its participation in the chat, a user issues a leave at any time after it has issued a join.

Data transfer is multi-cast, which means each datareq causes corresponding dataind's in all other participants in the chat, i.e., all other users who have performed a join to the chat the sending user belongs to, and have not performed a leave after that. Data transfer in the chat service is not reliable: messages may get lost, but they never get corrupted; corrupted messages are discarded. Also, the sequence delivery of messages is not guaranteed.

We have created a FA for this protocol. This automaton considers two chat sessions and two users. It has 39 states and 95 transitions. A formalization of the protocol in LTS can be found in Figure 1.

We consider different weights values per error, depending on the gravity of the error, their values can be found in Figure 2.

The state weight function r in the FA assigns different weights per state, depending on the possible errors from that state. We, also, consider three different discount functions, α_1, α_2 and α_3. Given a transition in the FA leaving from a state with outdegree n, α_1 assigns value $\frac{1}{8}$ to this transition; α_2 assigns $(\frac{1}{n} - \frac{1}{100})$ to it and α_3 assigns $(\frac{1}{n} - \frac{1}{10000})$.States error values, outdegree, and the different values of discount function can be found in Figure 3.

Figure 4 gives the total coverage in the FA (column 1) and the absolute coverage of the test suites containing all tests of length k (columns 2, 3, 4), for $k = 2, 4, 50$, for the various discount functions. These results have been obtained by applying Algorithm 5.2 (total coverage) and Algorithm 5.2 (relative coverage). We have used Maple 9.5 to resolve the matrix equations in these algorithms.

Figure 5 displays the relative coverage for test suites that have been generated automatically with TorX. For each test we use the discount function α_2. For

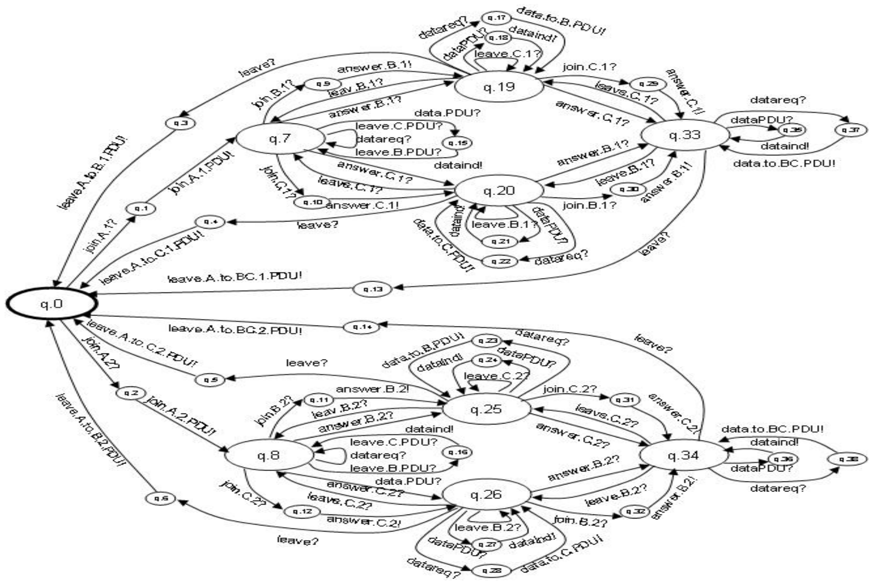

Fig. 1. Chat Protocol with two chats

given test lengths $k = 30$, $k = 35$, $k = 40$, $k = 45$ and $k = 50$, TorX has generated a test suite T^k, consisting of 10 tests $t_1^k, \dots t_{10}^k$ of length k. We have used Algorithm 5.1 to calculate the relative coverage of T^k. Figure 5 lists the coverage of each individual test t_i^k as well as for the test suites T^k. The running times of all computations were very small, in the order of a few seconds.

In the figures it is possible to appreciate who important the discount factor is, and who it influences in the coverage metrics.

7 Conclusions and future research

Semantic notions of test coverage have long been overdue, while they are much needed in the selection, generation and optimization of test suites. In this paper, we have presented semantic coverage notions based on weighted fault models. We have introduced fault automata, FA, to syntactically represent (a subset of) weighted fault models and provided algorithms to compute and optimize test coverage. This approach is purely semantic since replacing a FA with a semantically equivalent one leaves the coverage unchanged. Our experiments with the chat example indicate that our approach is feasible for small protocols. Larger case studies should evaluate the applicability of this framework for more complex systems.

Our fault models are based on (adaptive) ioco test theory. We expect that it is easy to adapt our approach to different settings, such as FSM testing or on-the-fly testing. Furthermore, our optimization techniques use test length as

neme of error	value	neme of error	value
join.A.1.PDU!out	3	leave.A.to.C.2.PDU!out	3
join.A.2.PDU!out	3	leave.A.to.BC.1.PDU!out	3
answer.B.1!out	7	leave.A.to.BC.2.PDU!out	3
answer.B.2!out	7	dataind!out	3
answer.C.1!out	7	data.to.B.PDUout	3
answer.C.2!out	7	data.to.C.PDU!out	3
leave.A.to.B.1.PDU!out	3	data.to.BC.PDU!out	3
leave.A.to.B.2.PDU!out	3	quiescent!out	10
leave.A.to.C.1.PDU!out	3		

Fig. 2. Error name and errors values

an optimality criterion. To accommodate more complex resource constraints (e.g time, costs, risks/probability) occurring in practice, it is relevant to extend our techniques with these attributes. Since these fit naturally within our model and optimization problems subject to costs, time and probability are well-studied, we expect that such extensions are feasible and useful.

References

1. Belinfante, A., Feenstra, J., Vries, R., Tretmans, J., Goga, N., Feijs, L., Mauw, S., and Heerink, L. Formal test automation: A simple experiment. In *Int. Workshop on Testing of Communicating Systems 12* (1999), G.Csopaki, S.Dibuz, and K.Tarnay, Eds., Kluwer, pp. 179–196.
2. Belinfante, A., Frantzen, L., and Schallhart, C. Tools for test case generation. In *Model-Based Testing of Reactive Systems* (2004), pp. 391–438.
3. Brinksma, E., Stoelinga, M., and Briones, L. B. A semantic framework for test coverage (extended version). In *Technical Report* (2006), TR-CTIT-06-24.
4. Briones, L. B., and Brinksma, E. A test generation framework for *quiescent* real-time systems. In *FATES'04. Also in http://fmt.cs.utwente.nl/research/testing/files/BBB04.ps.gz* (2004), pp. 64–78.
5. Campbell, C., Grieskamp, W., Nachmanson, L., Schulte, W., Tillmann, N., and Veanes, M. Model-based testing of object-oriented reactive systems. In *Technical Report* (2005), MSR-TR-2005-59.
6. ETSI. Es 201 873-6 v1.1.1 (2003-02). methods for testing and specification (mts). In *The Testing and Test Control Notation version 3: TTCN-3 Control Interface (TCI). ETSI Standard* (2003).
7. Jard, C., and Jéron, T. TGV: theory, principles and algorithms. *STTT 7*, 4 (2005), 297–315.
8. Myers, G. *The Art of Software Testing*. Wiley & Sons, 1979.
9. Myers, G., Sandler, C., Badgett, T., and Thomas, T. *The Art of Software Testing*. Wiley & Sons, 2004.
10. Nicola, R., and Hennessy, M. Testing equivalences for processes. In *ICALP83* (1983), vol. 154.
11. Tardos, E. A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5*, 3 (1985), 247–255.

state	r	outdegree	σ_1	σ_2	σ_3	state	r	outdegree	σ_1	σ_2	σ_3
state0	64	3	1/8	0.3233	0.3332	state20	64	8	1/8	0.115	0.1249
state1	71	1	1/8	0.99	0.999	state21	71	1	1/8	0.99	0.999
state2	71	1	1/8	0.99	0.999	state22	71	1	1/8	0.99	0.999
state3	71	1	1/8	0.99	0.999	state23	71	1	1/8	0.99	0.999
state4	71	1	1/8	0.99	0.999	state24	71	1	1/8	0.99	0.999
state5	71	1	1/8	0.99	0.999	state25	64	8	1/8	0.115	0.1249
state6	71	1	1/8	0.99	0.999	state26	64	8	1/8	0.115	0.1249
state7	64	9	1/8	0.1011	0.1110	state27	71	1	1/8	0.99	0.999
state8	64	9	1/8	0.1011	0.1110	state28	71	1	1/8	0.99	0.999
state9	67	1	1/8	0.99	0.999	state29	67	1	1/8	0.99	0.999
state10	67	1	1/8	0.99	0.999	state30	71	1	1/8	0.99	0.999
state11	67	1	1/8	0.99	0.999	state31	67	1	1/8	0.99	0.999
state12	67	1	1/8	0.99	0.999	state32	67	1	1/8	0.99	0.999
state13	71	1	1/8	0.99	0.999	state33	64	6	1/8	0.1566	0.1665
state14	71	1	1/8	0.99	0.999	state34	64	6	1/8	0.1566	0.1665
state15	71	1	1/8	0.99	0.999	state35	71	1	1/8	0.99	0.999
state16	71	1	1/8	0.99	0.999	state36	71	1	1/8	0.99	0.999
state17	71	1	1/8	0.99	0.999	state37	67	1	1/8	0.99	0.999
state18	71	1	1/8	0.99	0.999	state38	71	1	1/8	0.99	0.999
state19	64	8	1/8	0.115	0.1249	state39	71	1	1/8	0.99	0.999

Fig. 3. Value of r, outdegree and σ per state

	tw	twk, $k=2$	twk, $k=4$	twk, $k=50$
α_1	99.134	89.750	97.171	99.135
α_2	511.369	130.607	239.025	510.768
α_3	743.432	132.652	249.320	733.540

Fig. 4. Total coverage and maximal coverage of test with length k

12. Tretmans, J. Test generation with inputs, outputs and repetitive quiescence. In *Software-Concepts and Tools, 17(3)* (1996), Also: Technical Report N0. 96-26, Center for Telematics and Information Technology, University of Twente, The Netherlands, pp. 103–120.
13. Tretmans, J., and Brinksma, E. Torx: Automated model-based testing. In *First European Conference on Model-Driven Software Engineering, Nuremberg* (2003), A.Hartmann and K.Dussa-Ziegler.
14. Ural, H. Formal methods for test sequence generation. *Computer Communications Journal 15*, 5 (1992), 311–325.
15. van der Bijl, M., Rensink, A., and Tretmans, J. Compositional testing with ioco. In *FATES* (2003), pp. 86–100.

	test t_1^k	test t_2^k	test t_3^k	test t_4^k	test t_5^k	test t_6^k	test t_7^k	test t_8^k	test t_9^k	test t_{10}^k	suite T^k
$k = 30$	15.275	4.573	13.983	5.322	15.278	4.5877	14.235	8.502	15.265	4.898	63.052
$k = 35$	14.100	15.275	15.263	8.537	8.579	5.348	15.275	8.536	8.495	4.900	69.146
$k = 40$	5.325	13.968	14.237	15.276	5.343	14.130	15.275	5.314	13.980	15.276	72.848
$k = 45$	5.021	8.536	13.969	4.969	8.548	15.275	4.894	15.263	4.532	14.235	47.153
$k = 50$	5.320	72.802	5.326	4.898	13.982	5.319	14.233	5.320	13.968	15.289	54.204

Fig. 5. Relative coverage, as a percentage, of tests with length k using α_2.