A Semantic Framework for Test Coverage (Extended Version)[⋆]

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

[+] 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 has 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 black box test coverage. Our starting point is a weighted fault model (or WFM), which augments a specification by assigning a weight to each error that may occur 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 WFMs 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.

[⋆] This research has been partially funded by the Netherlands Organization for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.000.505 (MOQS) and grant number 612.064.207 (STRESS), the EU under grant IST-004527 (ARTIST2), and by the DFG/NWO bilateral cooperation program under project number DN 62-600 (VOSS2).
An extended abstract of this paper appears in [BBS06].

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 between an implementation and a specification. Automated generation methods for test suites from specifications [Tre96,TB03,BB04,NH83] have been developed, which have lead to a new generation of powerful test generation and execution tools such as SpecExplorer[CGN$^+$05], TorX[BFS04] and TGV[JJ05].

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. [Tre96]).

In practice, the above notion of exhaustiveness is usually problematic, since exhaustive test suites 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 [Ura92,LY96,NVS$^+$04]; 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 [Mye79,MSBT04,Bal04]. 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 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 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 (WFM) that assigns a weight to each potential error in an implementation. We define our coverage measures relative to these WFMs. Since WFMs are augmented specifications, our coverage framework qualifies as black box.

Since WFMs 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 [Tre96] (recapitulated in Section 3), but their principles apply to a much wider setting.

We provide two ways of deriving WFMs from fault automata, namely the finite depth WFMs (Section 4.1) and the discounted WFMs (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 with respect to a WFM. 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). Related work is discussed in Section 7 and we end by providing conclusions and suggestions for further research (Section 8).

2 Coverage measures in weighted fault models

Preliminaries. Let L be any set. Then L^* denotes the set of all finite 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 $\sigma' \in L^*$. If σ is a prefix of ρ, then ρ is a *suffix* of σ. We call σ a *proper prefix* of ρ and ρ a *proper suffix* of σ if $\sigma \sqsubseteq \rho$, but $\sigma \neq \rho$.

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 by also giving the severity of the erroneous traces. In this section, we work with a fixed action alphabet L.

Definition 1. *A* weighted fault model (WFM) *over L is a function $f : L^* \to \mathbb{R}^{\geq 0}$ such that*

$$0 < \textstyle\sum_{\sigma\in L^*} f(\sigma) < \infty.$$

Thus, a WFM f assigns a non-negative error weight to each trace $\sigma \in L^*$. If $f(\sigma) = 0$, then σ represents correct system behavior; if $f(\sigma) > 0$, then σ represents incorrect behavior and $f(\sigma)$ denotes the severity of the error. So, the higher $f(\sigma)$, the worse the error. We sometimes refer to traces $\sigma \in L^*$ with $f(\sigma) > 0$ as *error traces* and traces with $f(\sigma) = 0$ as *correct traces* in f.

We require the total error weight $\sum_{\sigma \in L^*} f(\sigma)$ to be finite and non-zero, in order to define coverage measures relative to the total error weight.

2.2 Coverage measures

This section abstracts from the exact shape of test cases and test suites. Given a WFM f over action alphabet L, we only use that a test is a trace set, $t \subseteq L^*$; and a test suite is a collection of trace sets, $T \subseteq \mathcal{P}(L^*)$. In this way we define the absolute and relative coverage w.r.t. f of a test and for a test suite. Moreover, our coverage measures apply in all settings where test cases can be characterized as trace sets (in which case test suites can be characterized as collections of trace sets). This is a.o. true for tests in TTCN [ETS03], ioco test theory [Tre96] and FSM testing [Ura92,LY96].

Definition 2. *Let $f : L^* \to \mathbb{R}^{\geq 0}$ be a WFM 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*

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

The coverage of a test suite T w.r.t. f measures the total weight of the errors that can be detected by tests in T. The absolute coverage $\mathit{abscov}(T, f)$ simply accumulates the weights of all error traces in T. Note that the weight of 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 $\mathit{relcov}(T, f)$ yields the error weight in T as a fraction of the weight of all traces in L^*. Since absolute (coverage) numbers have meaning only if they are put in perspective of a maximum or average; we advocate that the relative coverage yields a good indication for the quality of a test suite.

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 WFM $f : L^* \to \mathbb{R}^{\geq 0}$ if $\mathit{relcov}(T, f) = 1$.*

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

Proposition 1. *Let f be a WFM over L and let $T \subseteq \mathcal{P}(L^*)$ be a test suite. Then T 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 Test cases in labeled input-output transition systems

This section recalls some basic theory about test derivation from labeled input-output transition systems, following ioco testing theory [Tre96]. It prepares for the next section that treats an automaton-based formalism for specifying WFMs.

3.1 Labeled input-output transition systems

Definition 4. *A* labeled input-output transition system (LTS) $\mathcal{A}$ *is a tuple* $\langle S, s^0, L, \Delta \rangle$*, where*

- *S is a finite set of states*
- *$s^0 \in S$ is the initial state*
- *L is a finite action alphabet. We assume that $L = L^I \cup L^O$ is partitioned (i.e. $L^I \cap L^O = \emptyset$) 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 $a!$*
- *$\Delta \subseteq S \times L \times S$ 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 transition relation Δ^I is the restriction of Δ to $S \times L^I \times S$ and the output transition relation Δ^O is the restriction of Δ to $S \times L^O \times S$. 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 $outdeg(s) = |\Delta(s)|$ the out-degree of state s, i.e. the number of transitions leaving s*

We denote the components of $\mathcal{A}$ by $S_{\mathcal{A}}$, $s^0_{\mathcal{A}}$, $L_{\mathcal{A}}$, and $\Delta_{\mathcal{A}}$. We omit the subscript $\mathcal{A}$ if it is clear from the context.

We have required $\mathcal{A}$ to be deterministic only for technical simplicity. This is not a real restriction, since we can always determinize $\mathcal{A}$. We can also incorporate quiescence (i.e. the absence of outputs), 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 $\Delta^O(s) = \emptyset$ and considering δ as an output action. But, since quiescence is not preserved under determinization, we must first determinize and then add quiescence.

Example 1. Figure 1 presents a LTS of a MP3 player: if the user pushes the play-button, a song should be played. In b), we see the extension with quiescence. Since δ is not enabled in state s_1, we explicitly forbid the absence of outputs in s_1, i.e. a song must be played. The double circles represent the initial state.

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 $s_0 = s^0$ and, for all $1 \leq i \leq n$, we have $(s_{i-1}, a_i, s_i) \in \Delta$. We denote by $paths_{\mathcal{A}}$ the set of all paths in $\mathcal{A}$. Moreover, we define by $last(\pi) = s_n$ the last state of π and by $|\pi| = n + 1$ the amount of states of π.*
- *The* trace *of a path π, $trace(\pi)$, is the sequence $a_1 a_2 \ldots a_n$ of actions occurring in π. We write $traces_{\mathcal{A}} = \{trace(\pi) | \pi \in paths_{\mathcal{A}}\}$ for the set of all traces in $\mathcal{A}$.*

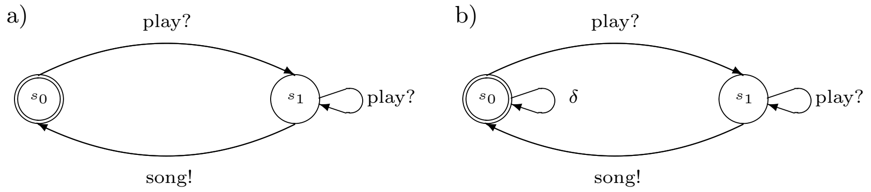

Fig. 1. A label input-output transition system specification of a MP3 player and its extension with quiescence

- *Let $\sigma \in L^*$ be any trace, not necessarily one from $\mathcal{A}$. We write $reach^k_{\mathcal{A}}(\sigma)$ for the set of states that can be reached in $\mathcal{A}$ in exactly k steps by following σ, i.e. $s' \in reach^k_{\mathcal{A}}(\sigma)$ if $|\sigma| = k$ and there is a path $\pi \in paths_{\mathcal{A}}$ such that $trace(\pi) = \sigma$ and $last(\pi) = s'$. We write $reach_{\mathcal{A}}(\sigma)$ for the set of states that can be reached via trace σ in any number of steps, i.e. $reach_{\mathcal{A}}(\sigma) = \cup_{k\in\mathbb{N}} reach^k_{\mathcal{A}}(\sigma)$; we write $reach^k_{\mathcal{A}}$ for the set of states that can be reached in k number of steps, by following any trace, i.e. $reach^k_{\mathcal{A}} = \cup_{\sigma\in L^*} reach^k_{\mathcal{A}}(\sigma)$; and $reach_{\mathcal{A}} = \cup_{\sigma\in L^*} reach_{\mathcal{A}}(\sigma)$ for the set of all reachable states in $\mathcal{A}$.*

As before, we leave out the subscript $\mathcal{A}$ if it is clear from the context.

Definition 6. *Let $\mathcal{A}$ be a LTS and $s \in S$ be a state in $\mathcal{A}$, then $\mathcal{A}[s]$ denotes the LTS $\langle S, s, L, \Delta\rangle$.*

Thus, $\mathcal{A}[s]$ is the same as $\mathcal{A}$, but with s as initial state. This notation allows us to speak of paths, traces, etc, in $\mathcal{A}$ starting from a state that is not the initial state. For instance, $paths_{\mathcal{A}[s]}$ denotes the set of paths starting from state s.

3.2 Test cases

Test cases for LTSs are based on ioco test theory [Tre96]. 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 $\sigma \in traces_{\mathcal{A}}$, but $\sigma a? \notin trace_{\mathcal{A}}$, 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 7. • *A* test case *(or test) t for a LTS $\mathcal{A}$ is a finite, prefix-closed subset of $L^*_{\mathcal{A}}$ 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 $\sigma \notin traces_{\mathcal{A}}$, 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 test t is the length of the longest trace in t, i.e. $|t| = \max_{\sigma \in t} |\sigma|$. We denote by $\mathcal{T}^k(\mathcal{A})$ the set of all tests for $\mathcal{A}$ with length k.*

Example 2. Figure 2 shows two test cases for the MP3 player from Figure 1, represented as trees and augmented with verdicts pass and fail. The prefix closed trace set is obtained by taking all traces in these trees.

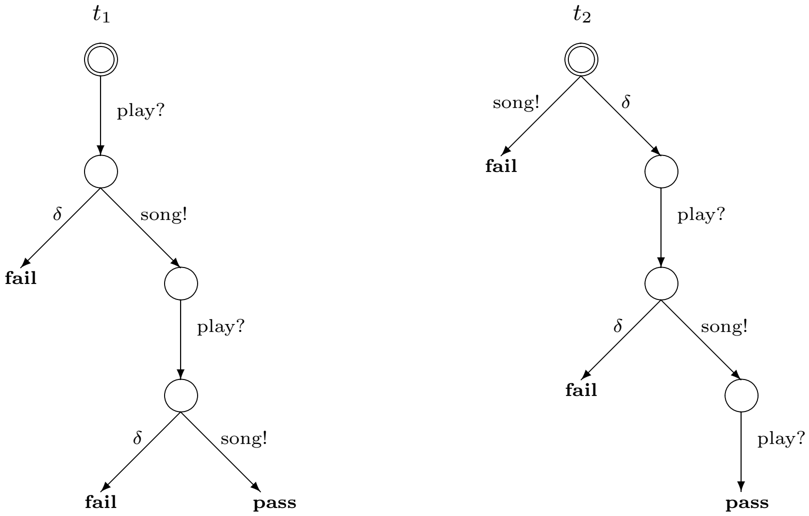

Fig. 2. Two test cases for the LTS from Figure 1, augmented with verdicts **pass** and **fail**

Since each test of $\mathcal{A}$ is a set of traces, we can apply Definition 2 and speak of (absolute, total and relative) coverage of a test case (or a test suite) of $\mathcal{A}$, w.r.t to a WFM f. However, not all WFMs are consistent with the interpretation that traces of $\mathcal{A}$ represent correct system behavior, and that tests are "fail fast" and do not fail after an input.

Definition 8. *Let $\mathcal{A}$ be a LTS and let $f : L^* \to \mathbb{R}^{\geq 0}$ be a WFM. Then f is* consistent with *$\mathcal{A}$ if $L = L_{\mathcal{A}}$ and for all $\sigma \in L^*_{\mathcal{A}}$ we have*

- *If $\sigma \in traces_{\mathcal{A}}$, then $f(\sigma) = 0$ (correct traces have weight 0).*
- *$f(\sigma a?) = 0$ (no failure occurs after an input).*
- *If $f(\sigma) > 0$ then $f(\sigma\rho) = 0$ for all $\rho \in L^+_{\mathcal{A}}$ (at most one failure per trace).*

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

Theorem 1. *Let $\mathcal{A}$ be a LTS and f be a WFM consistent with $\mathcal{A}$. Then, the set $\mathcal{T}(\mathcal{A})$ of all test cases for $\mathcal{A}$ is complete w.r.t. f.*

Proof. For all $\sigma \in L$ with $f(\sigma) > 0$, we build a test $t \in \mathcal{T}(\mathcal{A})$ 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 defined as $t = \cup_{1 \leq i \leq n} X_i$. Since f is consistent with $\mathcal{A}$, the set t is a test in $\mathcal{T}(\mathcal{A})$. Clearly, t contains σ. Now, Proposition 1 yields that $\mathcal{T}(\mathcal{A})$ is complete for f. □

4 Fault automata

Weighted fault models are infinite, semantic objects. This section introduces *fault automata*, which provide a syntactic format for specifying WFMs. 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, as explained below.

Definition 9. *A* fault automaton (FA) *$\mathcal{F}$ is a pair $\langle \mathcal{A}, r \rangle$, where $\mathcal{A} = \langle S, s^0, L, \Delta \rangle$ is a LTS and $r : S \times L^O \to \mathbb{R}^{\geq 0}$. We require that, if $r(s, a!) > 0$, then there is no $a!$-successor of s in $\mathcal{F}$, i.e. there is no $s' \in S$ such that $(s, a!, s') \in \Delta$. We extend r to a function $r : S \times L \to \mathbb{R}^{\geq 0}$ by putting $r(s, a?) = 0$ for $a? \in L^I$ and define $\overline{r} : S \to \mathbb{R}^{\geq 0}$ as $\overline{r}(s) = \sum_{a \in L^O(s)} r(s, a)$. 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 and notations (e.g. traces, paths, etcetera) that have been defined for LTSs to FAs.*

Example 3. Figure 3 presents an FA for our MP3 example. We give error weight 10 if in state s_0 a song is played; and weight 5 if in state s_1 no song occurs.

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

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

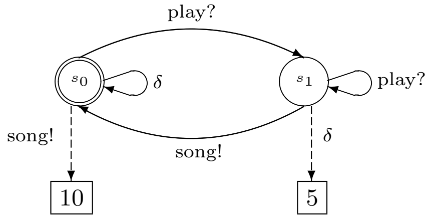

Fig. 3. A FA for the MP3 player

4.1 Finite depth weighted fault models

As said before, the finite depth model derives a WFM from an FA $\mathcal{F}$, for a given $k \in \mathbb{N}$, by ignoring all traces of length longer 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 $\sigma a!$ is not a trace in $\mathcal{F}$, then $\sigma a!$ gets weight $r(s, a!)$.

Definition 10. *Given an FA $\mathcal{F}$, and a number $k \in \mathbb{N}$, we define the function $f^k_{\mathcal{F}} : L^* \to \mathbb{R}^{\geq 0}$ by*

$$f^k_{\mathcal{F}}(\varepsilon) = 0 \qquad f^k_{\mathcal{F}}(\sigma a) = \begin{cases} r(s,a) & \text{if } s \in \mathit{reach}^l_{\mathcal{F}}(\sigma) \wedge a \in L^O \wedge l \leq k \\ 0 & \text{otherwise} \end{cases}$$

Note that this function is uniquely defined because $\mathcal{F}$ is deterministic, so that there is at most one s with $s \in \mathit{reach}^k_{\mathcal{F}}(\sigma)$. Also, if $f^k_{\mathcal{F}}(\sigma a) = r(s, a) > 0$, then $\sigma \in \mathit{traces}_{\mathcal{F}}$, but $\sigma a \notin \mathit{traces}_{\mathcal{F}}$.

The following proposition states that $f^k_{\mathcal{F}}$ is a WFM consistent with $\mathcal{F}$, provided that $\mathcal{F}$ contains as most one state with a positive accumulated weight and that is reachable within k steps.

Proposition 2. *Let $\mathcal{F}$ be an FA, and $k \in \mathbb{N}$. If there is an $i \leq k$ and a state $s \in \mathit{reach}^i_{\mathcal{F}}$ with $\overline{r}(s) > 0$, then $f^k_{\mathcal{F}}$ is a WFM consistent with $\mathcal{F}$.*

Example 4. Given the FA $\mathcal{F}$ from Figure 3 , Figure 4 shows the function $f^k_{\mathcal{F}}$ for $k = 3$. Using the tests t_1 and t_2 presented in Figure 2, we obtain $\mathit{abscov}(t_1, f^k_{\mathcal{F}}) = 5$ and $\mathit{abscov}(t_2, f^k_{\mathcal{F}}) = 15$. Moreover, if $T = \{t_1, t_2\}$ then $\mathit{abscov}(T, f^k_{\mathcal{F}}) = 20$.

4.2 Discounted weighted fault models

While finite depth WFMs achieve finite total coverage by considering finitely many traces, discounted WFMs take into account the error weights 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 lower weights.

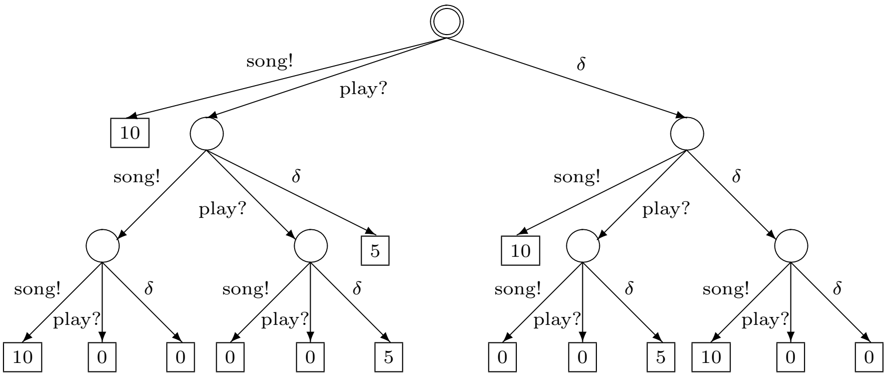

Fig. 4. Function $f^k_{\mathcal{F}}$, with $k = 3$, consistent with $\mathcal{F}$ from Figure 3

In its basic form, a discounted WFM 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, then one can easily show that $\sum_{\sigma \in L^*} f(\sigma) < \infty$. To be precise, we take $\alpha < \frac{1}{d}$, where d is the branching degree of $\mathcal{F}$ (i.e. $d = \max_{s \in S} outdeg(s)$). Indeed, let $\alpha d < 1$ and $M = \max_s r(s, a)/\alpha$. Then $f(\sigma) \leq \alpha^{|\sigma|}M$. Since there are at most d^k traces of length k in $\mathcal{F}$, it follows that

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

To obtain more flexibility, we allow the discount to vary per transition. That is, we work with a discount function $\alpha : S \times L \times S \to \mathbb{R}^{\geq 0}$ 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:

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

for each s. We can even be more flexible and, in the sum above, do not range over states in which all paths are finite, as in these states we have finite total coverage 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 11. *Let $\mathcal{F}$ be an FA. The set $\mathit{Inf}_{\mathcal{F}} \subseteq S_{\mathcal{F}}$ of states with at least one infinite path is defined as $\mathit{Inf}_{\mathcal{F}} = \{s \in S_{\mathcal{F}} \mid \exists \pi \in paths_{\mathcal{F}[s]} : |\pi| > |S_{\mathcal{F}}|\}$.*

The following proposition states that the set $\mathit{Inf}_{\mathcal{F}}$ is closed under taking the predecessors of a state.

Proposition 3. *Let $\mathcal{F}$ be an FA. If $(s, a, s') \in \Delta$ and $s' \in \mathit{Inf}_{\mathcal{F}}$, then $s \in \mathit{Inf}_{\mathcal{F}}$.*

Definition 12. *Let $\mathcal{F}$ be an FA. Then a* discount function *for $\mathcal{F}$ is a function $\alpha : S_{\mathcal{F}} \times L_{\mathcal{F}} \times S_{\mathcal{F}} \to \mathbb{R}^{\geq 0}$ such that*

- *For all $s, s' \in S_{\mathcal{F}}$, and $a \in L_{\mathcal{F}}$ we have $\alpha(s,a,s') = 0$ iff $(s,a,s') \notin \Delta_{\mathcal{F}}$.*
- *For all $s \in S_{\mathcal{F}}$, we have: $\sum_{a \in L_{\mathcal{F}}, s' \in Inf_{\mathcal{F}}} \alpha(s,a,s') < 1$.*

Definition 13. *Let α be a discount function for the FA $\mathcal{F}$. Given a path $\pi = s_0 a_1 \ldots s_n$ in $\mathcal{F}$, we define $\alpha(\pi)$ as $\prod_{i=1}^{n} \alpha(s_{i-1}, a_i, s_i)$.*

Definition 14. *Let $\mathcal{F}$ be an FA, $s \in S$, and α a discount function for $\mathcal{F}$. We define the function $f_{\mathcal{F}}^{\alpha} : L^* \to \mathbb{R}^{\geq 0}$ by*

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

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

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

The following proposition states that $f_{\mathcal{F}}^{\alpha}$ is a WFM consistent with $\mathcal{F}$, provided that $\mathcal{F}$ contains as most one reachable state with a positive accumulated weight.

Proposition 4. *Let $\mathcal{F}$ be an FA and α a discount function for $\mathcal{F}$. If there is a state $s \in \textit{reach}_{\mathcal{F}}$ with $\overline{r}(s) > 0$, then $f_{\mathcal{F}}^{\alpha}$ is a WFM consistent with $\mathcal{F}$.*

Example 5. Figure 5 presents function $f_{\mathcal{F}}^{\alpha}$ for $\mathcal{F}$ from Figure 3 and $\alpha(s,a,s') = \gamma$ for every transition $(s,a,s') \in \Delta$. Using the tests t_1 and t_2 presented in Figure 2, we obtain $\textit{abscov}(t_1, f_{\mathcal{F}}^{\alpha}) = \gamma 5$ and $\textit{abscov}(t_2, f_{\mathcal{F}}^{\alpha}) = 10 + \gamma^2 5$. Moreover, if $T = \{t_1, t_2\}$ then $\textit{abscov}(T, f_{\mathcal{F}}^{\alpha}) = 10 + \gamma 5 + \gamma^2 5$.

4.3 Properties

Calibration of the discount function. 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, for any FA $\mathcal{F}$, any given k and $u > 0$, there exists a discount function α such that the relative coverage of all test cases of length k or shorter is less than u. This means that by choosing the right α we can always make the contribution of short tests (i.e. smaller than the given k) arbitrarily small (i.e. $< u$).

For technical reasons, the weight assignment function of an FA have to be fair, i.e. all states in *Inf* must be able to reach some state with a positive weight.

Definition 15. *A FA $\mathcal{F}$ has* fair weight assignment *if for all $s \in Inf_{\mathcal{F}}$, there exists state $s' \in \textit{reach}_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$.*

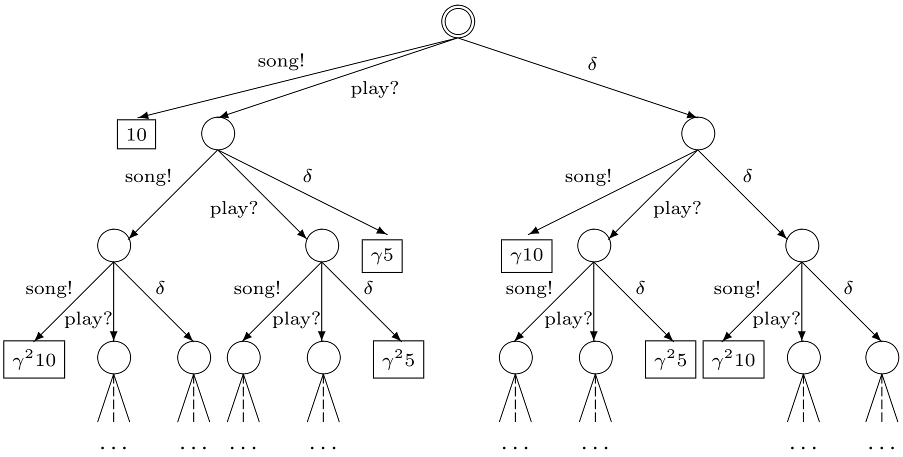

Fig. 5. Function $f_{\mathcal{F}}^{\alpha}$ for $\mathcal{F}$ from Figure 3 with $\alpha(s, a, s') = \gamma$

Theorem 2. *Let $\mathcal{F}$ be an FA with fair weight assignment. Then there exists a family of discount functions $\{\alpha_u\}_{u \in (0,1)}$ for $\mathcal{F}$ such that for all $k \in \mathbb{N}$*

$$\lim_{u \downarrow 0} relcov(\mathcal{T}^k(f_{\mathcal{F}}^{\alpha_u}), f_{\mathcal{F}}^{\alpha_u}) = 0$$

The proof of this result is given in the Appendix, page 24. Here, we only mention that the family of discount functions $\alpha_u : S \times L \times S \to (0, 1)$ with $u \in (0, 1)$ is given by

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

where $OutInf(s) = \{(a, s') \in \Delta(s) \mid s' \in Inf_{\mathcal{F}}\}$.

Invariance under bisimilarity. It is not difficult to see that our coverage notions are truly semantic in that they are invariant under r-preserving bisimilarity, and α-preserving bisimilarity.

Definition 16. *Let $\mathcal{F}$ be an FA and let $R \subseteq S \times S$ be an equivalence relation on the state space of $\mathcal{F}$. Then R is a r-*preserving bisimulation *on $\mathcal{F}$ if for all $(s, s') \in R$, $a \in L$, we have*

- $\forall\ b \in L^O : r(s, b) = r(s', b)$
- *if $s \xrightarrow{a} s_1$ then there is a transition $s' \xrightarrow{a} s_1'$ with $(s_1, s_1') \in R$.*

Theorem 3. *Let $\mathcal{F}$ be an FA, R be a r-preserving bisimulation on $\mathcal{F}$ and $(s, s') \in R$. Then $f_{\mathcal{F}[s]}^{k} = f_{\mathcal{F}[s']}^{k}$ for all k.*

Proof.
We prove that $f^k_{\mathcal{F}[s]}(\sigma \cdot b) = f^k_{\mathcal{F}[s']}(\sigma \cdot b)$.

Let $f^k_{\mathcal{F}[s]}(\sigma \cdot b) = r(s_1, b)$ then by definition we know that $s_1 \in \mathit{reach}_{f^k_{\mathcal{F}[s]}}(\sigma) \wedge b \in L^O$ then because R is an r-preserving bisimulation on $\mathcal{F}$ then $\exists\ s'_1 \in \mathit{reach}_{f^k_{\mathcal{F}[s']}}(\sigma)$ such that $(s_1, s'_1) \in R$ and moreover for $b \in L^O$ we have $f^k_{\mathcal{F}[s']}(\sigma \cdot b) = r(s'_1, b)$. Now, again because $(s_1, s'_1) \in R$ we know that $r(s_1, b) = r(s'_1, b)$. So, $f^k_{\mathcal{F}[s]}(\sigma \cdot b) = f^k_{\mathcal{F}[s']}(\sigma \cdot b)$.

Definition 17. *Let $\mathcal{F}$ be an FA and let $R \subseteq S \times S$ be an equivalence relation on the state space of $\mathcal{F}$. Then R is a* α-preserving bisimulation on $\mathcal{F}$ *if for all $(s, s') \in R$, $a \in L$, we have*

- $\forall\ b \in L^O : \alpha(s, a, s_1) = \alpha(s', a, s'_1)$
- *if $s \xrightarrow{a} s_1$ then there is a transition $s' \xrightarrow{a} s'_1$ with $(s_1, s'_1) \in R$.*

Theorem 4. *Let $\mathcal{F}$ be an FA, R be a r-preserving bisimulation and α-preserving bisimulation on $\mathcal{F}$. Then, if $(s, s') \in R$. Then $f^\alpha_{\mathcal{F}[s]} = f^\alpha_{\mathcal{F}[s']}$.*

Proof.
We prove that $f^\alpha_{\mathcal{F}[s]}(\sigma \cdot b) = f^\alpha_{\mathcal{F}[s']}(\sigma \cdot b)$.

Let $f^\alpha_{\mathcal{F}[s]}(\sigma b) = \alpha(\pi) r(s_1, b)$ then by definition we know that $s_1 \in \mathit{reach}_{f^\alpha_{\mathcal{F}[s]}}(\sigma) \wedge b \in L^O \wedge \sigma(\pi) = \sigma$ then because R is an r-preserving bisimulation on $\mathcal{F}$ then $\exists\ s'_1 \in \mathit{reach}_{f^\alpha_{\mathcal{F}[s']}}(\sigma) \wedge \exists\ \pi' : \sigma(\pi) = \sigma'$ such that $(s_1, s'_1) \in R$ and moreover for $b \in L^O$ we have $f^\alpha_{\mathcal{F}[s']}(\sigma \cdot b) = \alpha(\pi') \cdot r(s'_1, b)$. Now, again because $(s_1, s'_1) \in R$ using that R is α-preserving bisimulation on $\mathcal{F}$ then $\alpha(\pi) = \alpha(\pi')$. So, $f^\alpha_{\mathcal{F}[s]}(\sigma \cdot b) = f^\alpha_{\mathcal{F}[s']}(\sigma \cdot b)$.

More weighted fault models from fault automata. We like to stress that the finite depth and discounted models are just two examples for deriving WFMs from fault automata, but there are many more possibilities. For instance, one may combine the two and do not discount the weights of traces of length less than some k, 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 be adapted for WFMs with such variations.

5 Algorithms

This section presents various algorithms for computing and optimizing coverage for a given FA, interpreted under the finite depth or discounted weighted fault model. In particular, Section 5.1 presents algorithms to calculate the absolute coverage in a test suite of a given FA. In Section 5.2 we give algorithms that yield the total coverage in a weighted fault model derived from a FA. Section 5.4 provides three optimization algorithms for tests with length k. The first one finds a test case with maximal coverage; the second one finds the n test cases with maximal coverage; and the third one finds a test suite with n test cases with maximal coverage (i.e. the best n test cases with minimum overlap).

We use the following notation. Recall that $\mathcal{F}[s]$ denotes the FA that is the same as $\mathcal{F}$, but with s as initial state. When $\mathcal{F}$ is clear from the context, we write respectively f_s^k and f_s^α for the weighted fault models $f_{\mathcal{F}[s]}^k$ and $f_{\mathcal{F}[s]}^\alpha$ derived from $\mathcal{F}$. Moreover, given an FA $\mathcal{F} = \langle \mathcal{A}, r \rangle$, we write $A_\mathcal{F}$ for the multi-adjacency matrix of $\mathcal{A}$. That is, $A_\mathcal{F}$ contains at position (s, s') the number of edges between s and s', i.e. $(A_\mathcal{F})_{ss'} = \sum_{a:(s,a,s') \in \varDelta} 1$. If α is a discount function for $\mathcal{F}$, then $A_\mathcal{F}^\alpha$ is a weighted version of $A_\mathcal{F}$, i.e. $(A_\mathcal{F}^\alpha)_{ss'} = \sum_{a \in L} \alpha(s, a, s')$. We omit the subscript $\mathcal{F}$ if it is clear from the context.

5.1 Absolute coverage in a test suite

Given test suite T, an FA $\mathcal{F}$, and either a discounting function α for $\mathcal{F}$ or a number k, we desire to compute

$$abscov(T, f) = abscov(\bigcup_{t \in T} t, f)$$

where $f = f_\mathcal{F}^k$ or $f_\mathcal{F}^\alpha$. Given two tests t and t' and an action a, we write at for $\{a\sigma \mid \sigma \in t\}$ and $t + t'$ for the union $t \cup t'$. We call a super-test the union of any number of tests.

Now, we can write each test as $t = \varepsilon$; 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 all output actions of $\mathcal{F}$. Each super-test can be written as $a_1 t_1' + \cdots + a_k t_k' + b_1 t_1'' + \cdots + b_n t_n''$ where a_i are inputs and b_i are all outputs and t_i', t_j'' are super-tests.

To compute the union $\cup_{t \in T} t$, we recursively merge all tests in T into a super-test using the infix operator $\uplus$, Then we add the error weights of all traces in $\cup_{t \in T} t$ via the function ac defined below.

Merging of tests. Let $t' = a_1 t_1' + \cdots + a_k t_k' + b_1 t_1'' + \cdots + b_n t_n''$ be a super-test and t be a test. Then $t = \varepsilon$ or $t = at_1$ or $t = b_1 t_1' + \cdots + b_n t_n'$. Then we define

$$t' \uplus t =$$
$$\begin{cases} a_1 t_1' + \cdot\cdot + a_j (t_j' \uplus t_1) + \cdot\cdot + a_k t_k' + b_1 t_1'' + \cdot\cdot + b_n t_n'' & \text{if } t = at_1 \wedge a = a_j \\ a_1 t_1' + \cdot\cdot + a_k t_k' + b_1 (t_1'' \uplus t_1) + \cdot\cdot + b_n (t_n'' \uplus t_n) & \text{if } t = b_1 t_1' + \cdot\cdot + b_n t_n' \\ t' + t & \text{otherwise} \end{cases}$$

Absolute coverage in a super-test. Given a super-test t of $\mathcal{F}$ and a state s on $\mathcal{F}$, then

$$\begin{aligned} ac(\varepsilon, s) &= 0 \\ ac(t, s) &= \textstyle\sum_{i=1}^n aux(a_i t_i, s) \end{aligned}$$

where

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

The correctness of this algorithm is stated in the following theorem.

Theorem 5. *Given an FA $\mathcal{F}$, a state $s \in V$, a number $k \in \mathbb{N}$, a function $\alpha : S \times L \times S \to [0,1]$ and T a test suite, then*

- *If α is a discount function for $\mathcal{F}$, then $abscov(T, f_s^\alpha) = ac(\uplus T, s)$*
- *If $k \geq \max_{t\in T} |t|$ and $\alpha(s,a,s') = 1$ for all transitions (s,a,s') in $\mathcal{F}$, then $abscov(T, f_s^k) = ac(\uplus T, s)$.*

We write $\uplus\{t_1, t_2, \ldots t_n\}$ for $t_1 \uplus t_2 \uplus \ldots t_n$.

5.2 Total coverage

Total coverage in discounted FA. Given an FA $\mathcal{F}$, a state $s \in S$ and a discounting function α for $\mathcal{F}$, we desire to calculate $totcov(f_s^\alpha) = \sum_{\sigma\in L^*} f_s^\alpha(\sigma)$. We assume that from each state in $\mathcal{F}$ we can reach at least one error state (i.e. $\forall s \in S : \exists s' \in reach_{\mathcal{F}[s]} : \overline{r}(s) > 0$). In this way, f_s^α is a WFM for every s.

The basic idea behind the computation method is that the function $tc : S \to [0,1]$ ("total coverage") given by $s \mapsto totcov(f_s^\alpha)$ satisfies the following set of equations.

$$tc(s) = \overline{r}(s) + \sum_{a\in L, s'\in S} \alpha(s,a,s') \cdot tc(s') = \overline{r}(s) + \sum_{s'\in S} A^\alpha_{ss'} \cdot tc(s') \qquad (*)$$

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 $\sum_{a\in L} \alpha(s,a,s')$. Their correctness is shown in Proposition 9 in the Appendix. Using matrix-vector notation, we obtain

$$tc = \overline{r} + A^\alpha \cdot tc$$

Proposition 10 on page 27 of the Appendix states that the matrix $I - A^\alpha$ is invertible. Thus, we obtain the following result; in particular, tc is the unique solution of the equations (*) above.

Theorem 6. *Let $\mathcal{F}$ be an FA such that for all $s \in S$ there exists a state $s' \in reach_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$, and let α be a discount function for $\mathcal{F}$. Then*

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

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

Example 6. Given our FA $\mathcal{F}$ from Figure 3 and a discount function $\alpha = \{(s_0, \delta, s_0, \frac{1}{5}), (s_0, \text{play?}, s_1, \frac{1}{3}), (s_1, \text{play?}, s_1, \frac{1}{4}), (s_1, \text{song!}, s_0, \frac{1}{2})\}$. Then,

$$\begin{aligned} tc(s_0) &= \overline{r}(s_0) + \alpha(s_0, \delta, s_0) \cdot tc(s_0) \qquad + \alpha(s_0, \text{play?}, s_1) \cdot tc(s_1) \\ tc(s_1) &= \overline{r}(s_1) + \alpha(s_1, \text{song!}, s_0) \cdot tc(s_0) + \alpha(s_1, \text{play?}, s_1) \cdot tc(s_1) \end{aligned}$$

In matrix notation

$$\begin{aligned} tc &= \overline{r} + (A^{\alpha} \cdot tc) \\ tc - (A^{\alpha} \cdot tc) &= \overline{r} \\ (I - A^{\alpha}) \cdot tc &= \overline{r} \\ tc &= (I - A^{\alpha})^{-1} \cdot \overline{r} \end{aligned}$$

Then, with matrix A^{α}, matrix $I - A^{\alpha}$ and $(I - A^{\alpha})^{-1}$ equal to:

$$A^{\alpha} = \begin{bmatrix} \frac{1}{5} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix}, \; I - A^{\alpha} = \begin{bmatrix} \frac{4}{5} & \frac{-1}{3} \\ \frac{-1}{2} & \frac{3}{4} \end{bmatrix}, \; (I - A^{\alpha})^{-1} = \begin{bmatrix} \frac{45}{26} & \frac{10}{13} \\ \frac{15}{26} & \frac{24}{13} \end{bmatrix}$$

and $\overline{r}$ as the one given in Figure 3: $\overline{r} = [10, 5]$, we obtain $tc = 21.15384616$.

Total coverage in finite depth FA. Given an FA $\mathcal{F}$, a state $s \in S$ and a depth $k \in \mathbb{N}$, we desire to compute $totcov(f_s^k) = \sum_{\sigma \in L^*} f_s^k(\sigma)$. We assume that from each state, there is at least one error reachable in k steps (i.e. $\forall s \in S : \exists s' \in reach^k_{\mathcal{F}[s]} : \overline{r}(s') > 0$). This makes that f_s^k is a weighted fault model for any s.

The basic idea behind the computation method is that the function $tc_k : S \to [0,1]$ given by $s \mapsto totcov(f_s^k)$ satisfies the following recursive equations.

$$\begin{aligned} &tc_0(s) = 0 \\ &tc_{k+1}(s) = \overline{r}(s) + \sum_{(a,s') \in \Delta(s)} tc_k(s') = \overline{r}(s) + \sum_{a \in L, s' \in S} A_{s,s'} \cdot tc_k(s') \end{aligned}$$

The correctness of these equations follows from Proposition 12, stated on page 28 in the Appendix. In matrix-vector notation, we have

$$\begin{aligned} &tc_0 = 0 \\ &tc_{k+1} = \overline{r} + A \cdot tc_k \end{aligned} \tag{**}$$

Theorem 7. *Let $\mathcal{F}$ be an FA, a state $s \in S$ and $k \in \mathbb{N}$. If $\forall s \in S : \exists s' \in reach^k_{\mathcal{F}[s]} : \overline{r}(s') > 0$, then*

$$tc_k = \textstyle\sum_{i=0}^{k-1} A^i \cdot \overline{r}$$

Note that for a state s in an arbitrary $\mathcal{F}$, there exists a state $s' \in reach^k_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$ if and only if $(\sum_{i=0}^{k-1} A^i \cdot \overline{r})_s > 0$.

Complexity. Using Theorem 7 with sparse matrix multiplication, or iterating the equations just above it, tc_k can be computed in time $O(k \cdot |\Delta| + |S|)$.

Example 7. Given our FA $\mathcal{F}$ from Figure 3 and $k = 2$ the matrix A, becomes:

$$A = \begin{bmatrix} 1 \, 1 \\ 1 \, 1 \end{bmatrix} \quad \text{and} \quad tc_k = \begin{bmatrix} 10 \\ 5 \end{bmatrix} + \begin{bmatrix} 1 \, 1 \\ 1 \, 1 \end{bmatrix} \cdot \begin{bmatrix} 10 \\ 5 \end{bmatrix}$$

Then, we calculate $tc_k = 25$.

Remark 1. A similar method to the one above can be used to compute the weight of all tests of length k in the discounted weighted fault model, i.e. $abscov(T^k, f_s^\alpha)$, where T^k is the set of all tests of length k in $\mathcal{F}$.

Writing $tcd_k(s) = abscov(T^k, f_s^\alpha)$ ("total coverage discounted"), the recursive equations become

$$\begin{aligned} tcd_0(s) &= 0 \\ tcd_{k+1}(s) &= \overline{r}(s) + \sum_{a \in L, s' \in S} tcd_k(s') = \overline{r}(s) + \sum_{a \in L, s' \in S} A^\alpha_{ss'} \cdot tcd_k(s') \end{aligned}$$

and the analogon of Theorem 7 becomes

$$tcd_k = \textstyle\sum_{i=0}^{k-1} (A^\alpha)^i \cdot \overline{r} = (I - A^\alpha)^{-1} \cdot (I - (A^\alpha)^k) \cdot \overline{r}$$

The latter equality holds because $I - A^\alpha$ is invertible. Thus, the computing tcd_k requires one matrix inversion and, using the power method, $\log_2(k)$ matrix multiplications, yielding time complexity in $O(|S|^{\log_2 7} + |S|^{\log_2(k)})$ with Strassen's method.

If $(I - A^\alpha)$ can be put in diagonal form, the problem can be solved in $O(|S|^3 + \log_2 n)$. 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.

Example 8. Given our FA $\mathcal{F}$ from Figure 3, $k = 2$, matrix A^α, matrix $I - A^\alpha$, $(I - A^\alpha)^{-1}$, and $(A^\alpha)^k$ are equal to:

$$A^\alpha = \begin{bmatrix} \frac{1}{5} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{4} \end{bmatrix}, \; I - A^\alpha = \begin{bmatrix} \frac{4}{5} & \frac{-1}{3} \\ \frac{-1}{2} & \frac{3}{4} \end{bmatrix}, \; (I - A^\alpha)^{-1} = \begin{bmatrix} \frac{45}{26} & \frac{10}{13} \\ \frac{15}{26} & \frac{24}{13} \end{bmatrix}, \; (A^\alpha)^k = \begin{bmatrix} \frac{31}{150} & \frac{3}{20} \\ \frac{9}{40} & \frac{11}{48} \end{bmatrix}$$

Here, we calculate $tcd_k = 13.66666667$.

5.3 Relative coverage

Combining the algorithms for computing total and absolute coverage from the previous sections, one easily computes $relcov(T, f) = \frac{abscov(T,f)}{totcov(f)}$ for a test suite T and $f = f_s^k$ or $f = f_s^\alpha$.

5.4 Optimization

Optimal coverage in a test case. Given an FA $\mathcal{F}$ and a length k, we compute the best test case with length k (i.e. the one with highest coverage). We treat the finite depth and discounted model at once by fixing, in the finite depth model $\alpha(s, a, s') = 1$ if (s, a, s') is a transition in Δ and $\alpha(s, a, s') = 0$ otherwise. A function α is call *extended discount function* if it is a discount function or it is obtained from a finite depth model.

The optimization method is again based on recursive equations. We write $acopt_k(s) = \max_{t \in T^k} \{abscov(t, f_s^\alpha)\}$ ("optimal absolute coverage"). To understand the recursive characterization of $acopt_k$, we consider two situations. First, 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 $\alpha(s, a?, s') \cdot acopt_k(s')$. The best coverage that we can obtain by stimulating the IUT is given by $\max_{(a?,s')\in\Delta^I(s)} \alpha(s, a?, s') \cdot acopt_k(s')$.

Second, 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 $\overline{r}(s) + \sum_{(b!,s')\in\Delta^O(s)} \alpha(s, b!, s') \cdot acopt_k(s')$. Now, the optimal test $acopt(s)$ of length $k+1$ is obtained from $acopt_k$ by selecting from these options (i.e. inputing an action $a?$ or observing) the one with the highest coverage. Thus, we obtain the following result, which follows from Proposition 13 on page 28.

Theorem 8. *Let $\mathcal{F}$ be an FA, α be an extended discount function, and $k \in \mathbb{N}$ test length. Then $acopt_k$ satisfies the following recursive equations.*

$$\begin{aligned} acopt_0(s) &= 0 \\ acopt_{k+1}(s) &= \max\Big(\overline{r}(s) + \sum_{(b!,s')\in\Delta^O(s)} \alpha(s, b!, s') \cdot acopt_k(s'), \\ &\qquad \max_{(a?,s')\in\Delta^I(s)} \alpha(s, a?, s') \cdot acopt_k(s')\Big) \end{aligned}$$

Complexity. Based on Theorem 8, we can compute $acopt_k$ in time $O(k(|S|+|\Delta|))$.

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. We iterate the equations in Theorem 8 and stop as soon as we achieve coverage higher than c, i.e. at the first n with $acopt_n(s) > c$.

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

Theorem 9. *Let $\mathcal{F}$ be an FA, and α be a discount function for $\mathcal{F}$. Then mw is the unique solution of the following set of equations.*

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

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

Theorem 10. *Let $\mathcal{F}$ be an FA, and α be a discount function. Then mw is the optimal solution of the following LP problem.*

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

Complexity. The above LP problem contains $|S|$ variables and $|S|+|\Delta^I|$ inequalities. Thus, solving this problem is polynomial in $|S|$, $|S| + |\Delta^I|$ and the length of the binary encoding of the coefficients [Tar85]. In practice, the exponential time simplex method outperforms existing polynomial time algorithms.

Optimal coverage in n test cases. The first algorithm in this section for computing the best test case of length k can be extended to a method for computing the best n test cases with optimal coverage: the previous algorithm picks the best test case with length k. To pick the second best test case, we apply the same procedure, except that we exclude the first choice from all possible options; for the third best choice, we exclude the previous two, etc.

Optimal coverage in a test suite (with n test cases). Differently from the previous algorithm where the n chosen tests may overlap, we now present an algorithm to compute the best coverage in a test suite with n tests. In this test suite, we avoid test overlapping. Avoiding overlapping after we combine the tests in a super-test we obtain the test suite with optimal coverage. The idea is the following.

We write $\mathit{acopt}^n_k(s) = \underset{t_1,t_2,\dots t_n\in\mathcal{T}^k}{\mathit{maxn}}\{\mathit{abscov}(t, s)\}$, for the ordered list $[l_1, l_2, \dots l_n]$, where l_i is the coverage of the i^{th} best test of length k. We characterize acopt^n_k recursively. To do this we start dividing our reasoning in two sets: a test suite for inputs and a test suite for outputs, and later we combine them.

Assume the input actions are: $a_1, a_2, \dots a_m$. Let T be a test suite started in state s such that: $T = \{a_1T_1, \cdots, a_mT_m\}$. To compute $\mathit{acopt}^n_k(s)$ of the test suite T we assume that exists the set of n best test cases that start on s_i : $\mathit{acopt}^n_{k-1}(s_i)$ for all $0 \leq i \leq m$ (where s_i is the state reachable from s after the input action a_i). Then, let $\mathit{acopt}^n_{k-1}(s_i)$ be equal to $[l'_1, l'_2, \dots l'_n]$ where l'_1 is the optimal coverage of a test started in s_i, l'_2 is the second optimal coverage of a test started form s_i, etc. Then,

$$\begin{aligned}
\mathit{acopt}^n_{k-1}(s_i) &= [l'_1, l'_2, \dots l'_n] \\
&= [(\mathit{acopt}^n_{k-1}(s_i))_1, \cdots, (\mathit{acopt}^n_{k-1}(s_i))_n]
\end{aligned}$$

$$\begin{aligned}
acopt^n_k(s) &= maxn\ [\alpha(s,a_1,s_1)(acopt^n_{k-1}(s_1))_1, \cdots, \alpha(s,a_1,s_1)(acopt^n_{k-1}(s_1))_n, \cdots \\
&\qquad \alpha(s,a_m,s_m)(acopt^n_{k-1}(s_m))_1, \cdots, \alpha(s,a_m,s_m)(acopt^n_{k-1}(s_m))_n] \\
&= maxn\ [\alpha(s,a,s') \cdot l \mid (a?,s') \in \Delta^I(s) \wedge l \leftarrow (acopt^n_{k-1}(s'))_j \\
&\qquad \wedge\ 0 \leq j \leq n]
\end{aligned}$$

For the case of outputs. Assume the outputs actions are: $b_1, b_2, \ldots b_p$. Let T be a test suite started in state s such that $T = \{b_1T_1, \cdots, b_pT_p\}$. To compute $acopt^n_k(s)$ of the test suite T,again , we assume that exists the set of n best test cases that start on s_i: $acopt^n_{k-1}(s_i)$ for all $0 \leq i \leq m$ (where s_i is the state reachable from s after an output action b_i). Also, let $acopt^n_{k-1}(s_i)$ be equal to $[l'_1, l'_2, \ldots l'_n]$ where l'_1 is the optimal coverage of a test started in s_i, etc. Then,

$$\begin{aligned}
acopt^n_{k-1}(s_i) &= [l'_1, l'_2, \ldots l'_n] \\
&\quad [(acopt^n_{k-1}(s_i))_1, \cdots, (acopt^n_{k-1}(s_i))_n]
\end{aligned}$$

$$\begin{aligned}
acopt^n_k(s) &= [\overline{r}(s) + \alpha(s,b_1,s_1)(acopt^n_{k-1}(s_1))_1 + \cdots + \alpha(s,b_p,s_p)(acopt^n_{k-1}(s_n))_n, \cdots \\
&\qquad \overline{r}(s) + \alpha(s,b_1,s_1)(acopt^n_{k-1}(s_1))_n + \cdots + \alpha(s,b_p,s_p)(acopt^n_{k-1}(s_p))_n] \\
&= \overline{r}(s) \oplus [\sum_{(b!,s') \in \Delta^O(s)} \alpha(s,b,s') \cdot l \mid l \leftarrow (acopt^n_{k-1}(s'))_j \wedge 0 \leq j \leq n]
\end{aligned}$$

Here, $x \oplus l$ adds the number $x \in \mathbb{R}$ to each element of the list l (i.e., $x \oplus [e_1, e_2, \ldots e_n] = [e_1 + x, e_2 + x, \ldots e_n + x]$). 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.

Theorem 11. *Let be given an 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 tc_k satisfies the following equations.*

$$\begin{aligned}
v_0(s) =& [0, 0, .., 0] \\
v_{k+1}(s) =& maxn\{[\alpha(s,a,s') \cdot v \mid (a?,s') \in \Delta^I(s), v \leftarrow (v_k(s'))_j \wedge 0 \leq j \leq n] \\
& \quad ++ r(s) \oplus [\sum_{(b!,s') \in \Delta^O(s)} \alpha(s,b,s') \cdot l \mid l \leftarrow (v_k(s'))_j \wedge 0 \leq j \leq n]
\end{aligned}$$

Algorithm 18 *(Variation on the theme) Rather than computing in algorithm 5.4 the best test case of a fixed length k, we can compute the best test case with coverage c, for some $c < mw(s)$. That is, we compute v_0, v_1, $v_2 \ldots v_k$, until we find $v_k(s) \leq c$.*

6 Application: a chat protocol

This section applies our theory to a practical example, namely a chat protocol, also used as a conference protocol [BFV$^+$99]. This protocol provides a multi-cast service to users engaged in a chat session. Each user can send messages to and receive messages from all other partners participating in the same chat session.

The chat participants are dynamic, as the chat service allows them to join and leave the chat at any moment in time. Different chats can exist at the same time, but each user can only participate in at most one chat at a time.

The protocol specifies the data units, the underlying service and the chat service. 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 [BFV$^+$99]. The chat service is explained as follows. Each chat session has a name. 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 session; the user title identifies a user in a chat;
- *datareq*: a user sends a message to all other users participating in its session;
- *dataind*: a user receives a message from another user participating in its session;
- *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 joining, 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 that each datareq causes corresponding dataind's in all other participants in the chat. 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.

Figure 6 displays an LTS modeling the chat protocol. This model considers two chat sessions and three users. We consider different weights values per error, depending on the gravity of the error, their values can be found in Figure 7.

In the transition weight function r, we consider absence of required answers as the worse errors with weight 10; inappropriate answers as less serious with weight 7 and inappropriate joins or leaves as the least severe with weight 3. We interpret $\mathcal{F}$ as a discounted WSM under different discount functions, α_1, α_2 and α_3. If $\theta = (s, a, s')$ is a transition in $\mathcal{F}$ leaving from state s with out-degree d, we use $\alpha_1(\theta) = \frac{1}{8}$; $\alpha_2(\theta) = \frac{1}{d} - \frac{1}{100}$; and $\alpha_3(\theta) = \frac{1}{d} - \frac{1}{10000}$.

States error values, out-degree, and the different values of discount function can be found in Figure 8.

Figure 9 gives the total coverage in $\mathcal{F}$ (table on the left) and the absolute (table on the middle) and relative (table on the right) coverage of the test suites containing all tests of length k, for $k = 2, 4, 50$ and $\alpha_1, \alpha_2, \alpha_3$. These results were obtained by applying the first (total coverage in discounted FA) from Section 5.1 second algorithm from Section 5.2 and third algorithm (Remark 1) from Section 5.3. We used Maple 9.5 to resolve the matrix equations in these algorithms.

Figure 10 displays the relative coverage for test suites that have been generated automatically with TorX, using discount function α_2. For test lengths $k = 30, 35, 40, 45, 50$, TorX has generated a test suite T^k, consisting of 10 tests $t_1^k, \ldots t_{10}^k$ of length k. We used Algorithm 5.1 to calculate the relative coverage of T^k. Figure 10, also, 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. Notice the influence of the discount factor and the test length on the coverage numbers.

7 Related work

There is a vast literature on syntactic test coverage criteria. [Bal04]. Test coverage and optimization are well studied for (extended) finite state machines [Ura92,LY96]. Most works consider syntactic coverage measures and optimize preset tests, i.e. find the shortest sequence of inputs to the IUT that achieves a certain coverage.

Test optimization in the adaptive setting is also considered in [NVS$^+$04]. Their specification models are Markov Decision Processes, i.e. the tester chooses an input to the IUT and the IUT makes a probabilistic choice among all possible outputs, and assigns a cost to each transition to be executed. This paper provides optimization techniques for deriving test suites with maximal expected coverage for (final) states and transitions at minimal expected cost. Thus, their coverage criteria are syntactic.

The work [BdJV$^+$] optimize the order in which a test suite is executed, such that the impact (i.e. the probability that a certain error occurs times its weight) is maximized against total duration, cost and produced quality.

8 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 presented semantic coverage notions based on WFMs. We introduced fault automata, FA, to syntactically represent (a subset of) WFMs and provided algorithms to compute and optimize test coverage. This approach is purely semantic since replacing an FA with a semantically equivalent one (i.e. $\overline{r}$-preserving bisimilar) leaves the coverage unchanged. Our experiments with the chat protocol indicate that our approach is feasible for small protocols. Larger case studies should evaluate the applicability of this framework for more complex systems.

Our weighted fault models are based on (adaptive) ioco test theory. We expect to be 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 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 both feasible and useful.

References

[Bal04] Thomas Ball. A theory of predicate-complete test coverage and generation. In *Proceedings of FMCO'04*, pages 1–22, 2004.

[BB04] L. Brandán Briones and E. Brinksma. A test generation framework for *quiescent* real-time systems. In *FATES'04*, pages 64–78, 2004.

[BBS06] E. Brinksma, L. Brandán Briones, and M.I.A. Stoelinga. A semantic framework for test coverage. In S. Graf and W. Zhang, editors, *Proceedings of the fourth international symposium on Automated Technology for Verification and Analysis (ATVA'06)*, LNCS. Springer, 2006.

[BdJV$^+$] R. Boumen, I.S.M. de Jong, J.W.H. Vermunt, J.M. van de Mortel-Fronczak, and J.E. Rooda.

[BFS04] A. Belinfante, L. Frantzen, and C. Schallhart. Tools for test case generation. In *Model-Based Testing of Reactive Systems*, pages 391–438, 2004.

[BFV$^+$99] A. Belinfante, J. Feenstra, R.de Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In *Int. Workshop on Testing of Communicating Systems 12*, pages 179–196, 1999.

[CGN$^+$05] C. Campbell, W. Grieskamp, L. Nachmanson, W. Schulte, N. Tillmann, and M. Veanes. Model-based testing of object-oriented reactive systems. Technical Report MSR-TR-2005-59, 2005.

[ETS03] 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.

[JJ05] C. Jard and T. Jéron. TGV: theory, principles and algorithms. *STTT*, 7(4):297–315, 2005.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - A survey. In *Proceedings of the IEEE*, volume 84, pages 1090–1126, 1996.

[MSBT04] G. Myers, C. Sandler, T. Badgett, and T. Thomas. *The Art of Software Testing.* Wiley & Sons, 2004.

[Mye79] G. Myers. *The Art of Software Testing.* Wiley & Sons, 1979.

[NH83] R.de Nicola and M. Hennessy. Testing equivalences for processes. In *Proceedings ICALP*, volume 154, 1983.

[NVS$^+$04] L. Nachmanson, M. Veanes, W. Schulte, N. Tillmann, and W. Grieskamp. Optimal strategies for testing nondeterministic systems. In *International Symposium on Software Testing and Analysis*, pages 55–64. ACM Press, 2004.

[Tar85] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[TB03] J. Tretmans and E. Brinksma. TorX: Automated model-based testing. In *First European Conference on Model-Driven Software Engineering*, 2003.

[Tre96] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software-Concepts and Tools*, 17(3):103–120, 1996.

[Ura92] H. Ural. Formal methods for test sequence generation. *Computer Communications Journal*, 15(5):311–325, 1992.

ewpage

A Appendix

A.1 Calibration Theorem

This section is concerned with the proof of the Calibration result, stated in Theorem 2. We first recall this result, as well as the notion of fair weight assignment.

Definition 19. *An FA $\mathcal{F}$ has* fair weight assignment *if for all $s \in \mathit{Inf}$, there exists state $s' \in \mathit{reach}_{\mathcal{F}[s]}$ with $\overline{r}(s') > 0$.*

Theorem 12. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be an FA with fair weight assignment. Then there exists a family of discount functions α_u for $\mathcal{F}$ such that for all $k \in \mathbb{N}$*

$$\lim_{u \downarrow 0} \mathit{relcov}(\mathcal{T}_k(f_{\mathcal{F}}^{\alpha_u}), f_{\mathcal{F}}^{\alpha_u}) = 0$$

Definition 20. *Given an FA $\mathcal{F}$ and a number $u \in (0,1)$, we define a discount function $\alpha_u : S \times L \times S \to (0,1)$ by*

$$\alpha_u(s,a,s') = \begin{cases} \frac{(1-u)}{|\mathit{OutInf}(s)|} & \text{if } (s,a,s') \in \Delta \text{ and } s' \in \mathit{Inf}_{\mathcal{F}} \\ > 0 & \text{if } (s,a,s') \in \Delta \text{ and } s' \in S \backslash \mathit{Inf}_{\mathcal{F}} \\ 0 & \text{otherwise} \end{cases}$$

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

Definition 21. *Given an FA $\mathcal{F}$, we define the vector $\mathbf{1}_{\mathit{Inf}}$ indexed by $s \in S$ by*

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

Proposition 5. *$\mathbf{1}_{\mathit{Inf}}$ is an eigenvector of A_u with eigenvalue $1-u$, i.e.*

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

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

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

For $s \in S \backslash \mathit{Inf}_{\mathcal{F}}$ we get, using Proposition 3,

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

□

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

Proof. By induction on n. □

Proposition 6. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be an FA with a fair weight assignment r. Then $(\sum_{i=0}^{|S|-1} A_u^i \cdot \overline{r})_s > 0$ 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 an FA every state s is at most $|S|-1$ transitions removed any of the states s' that can be reached from it, so that there is an $i < |S|$ with $(A_u^i)_{ss'} > 0$. Hence $(\sum_{i=0}^{|S|-1} A_u^i)_{ss'} > 0$ for any pair of such $s, s' \in S$. By the definition of fair weight assignment all states $s \in \mathit{Inf}_{\mathcal{F}}$ can reach an $s' \in S$ with $\overline{r}(s') > 0$. Thus we get $(\sum_{i=0}^{|S|-1} A_u^i \cdot \overline{r})_s = \sum_{s' \in S} \sum_{i=0}^{|S|-1} (A_u^i)_{ss'} \cdot \overline{r}(s') > 0$. □

Now we are ready to show that the family of discounted functions $\{\alpha_u\}_{u \in (0,1)}$ has the desired properties.

Proposition 7. *Let $\mathcal{F} = \langle \mathcal{A}, r \rangle$ be an FA with fair weight assignment. Then for every $s \in Inf_{\mathcal{F}}$*

$$\lim_{u \downarrow 0} relcov(\mathcal{T}_k(f_{\mathcal{F}}^{\alpha_u}), f_{\mathcal{F}}^{\alpha_u}) = 0$$

Proof. Recall that

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

As $abscov(\mathcal{T}_k(f_{\mathcal{F}}^{\alpha_u}), f_{\mathcal{F}}^{\alpha_u})$ is always finite, it suffices to show that $\lim_{u \downarrow 0} totcov(f_{\mathcal{F}}^{\alpha_u}) = \infty$. This can be shown as follows.

Define $r_{min} = \min_{s' \in Inf} (\sum_{i=0}^{|S|-1} A_u^i \cdot \overline{r})_{s'}$. Then Proposition 6 yields that $r_{min} > 0$. Moreover, we have for all $s' \in S$ that

$$(\sum_{i=0}^{|S|-1} A_u^i \cdot \overline{r})_{s'} \geq r_{min} \cdot \mathbf{1}_{Inf}(s').$$

Therefore,

$$\begin{aligned} totcov(f_{\mathcal{F}}^{\alpha_u}) &= \big(\sum_{i=0}^{\infty} A_u^i \cdot \overline{r}\big)_s \\ &= \big(\sum_{j=0}^{\infty} A_u^{j|S|} \cdot \sum_{i=0}^{|S|-1} A_u^i \cdot \overline{r}\big)_s \\ &\geq r_{min} \cdot \big(\sum_{j=0}^{\infty} A_u^{j|S|} \cdot \mathbf{1}_{Inf}\big)_s \\ &= r_{min} \cdot \big(\sum_{j=0}^{\infty} (1-u)^{j|S|} \cdot \mathbf{1}_{Inf}\big)_s \\ &= \frac{r_{min}}{(1-(1-u)^{|S|})} \end{aligned}$$

As $r_{min} > 0$ and $1-(1-u)^{|S|}$ is of the order $\mathcal{O}(u)$, we get $\lim_{u \downarrow 0} \big(\sum_{i=0}^{\infty} A_u^i \cdot \overline{r}\big)_s = \infty$. □

A.2 Correctness proofs of the algorithms

Total coverage in discounted model. The following proposition gives an alternative recursive characterization of $\mathcal{F}$.

Proposition 8. *Let $\mathcal{F}$ be an FA and α a discount function for $\mathcal{F}$. Then*

$$f_s^{\alpha}(a\sigma) = \begin{cases} \sum_{s' \in S} \alpha(s, a, s') \cdot f_{s'}^{\alpha}(\sigma) & \text{if } a \in \Delta(s), \\ r(s, a) & \text{if } a \notin \Delta(s), \sigma = \varepsilon, \\ 0 & \text{otherwise} \end{cases}$$

where $a \in \Delta(s)$ means that $\exists\ s' : (a, s') \in \Delta(s)$.

Proposition 9. *Let $\mathcal{F}$ be an FA and α a discount function for $\mathcal{F}$. Then the function $tc : S \to [0,1], s \mapsto totcov(f_s^\alpha)$ satisfies the following set of equations.*

$$tc(s) = \overline{r}(s) + \sum_{a \in L, s' \in S} \alpha(s,a,s') \cdot tc(s')$$

Proof.

$$\begin{aligned}
tc(s) &= \sum_{\sigma \in L^*} f_s^\alpha(\sigma) \\
&= f_s^\alpha(\varepsilon) + \sum_{a \notin \Delta(s)} f_s^\alpha(a) + \sum_{a \notin \Delta(s), \sigma \in L^+} f_s^\alpha(a\sigma) + \sum_{a \in \Delta(s), \sigma \in L^*} f_s^\alpha(a\sigma) \\
&\text{(Proposition 8)} \\
&= 0 + \sum_{a \notin \Delta(s)} r(s,a) + 0 + \sum_{a \in \Delta(s), s' \in S, \sigma \in L^*} \alpha(s,a,s') \cdot f_{s'}^\alpha(\sigma) \\
&= \overline{r}(s) + \sum_{a \in L, s' \in S} \alpha(s,a,s') \cdot tc(s')
\end{aligned}$$

□

Proposition 10. *Let $\mathcal{F}$ be an FA such that for all states $s \in S$ there is a state $s' \in reach_{\mathcal{F}[s]}$ with $\overline{r}(s) > 0$. Let α be a discount function for $\mathcal{F}$. Then, the matrix $I - A^\alpha$ is invertible.*

Proof. By reordering the states we can obtain $Inf_{\mathcal{F}} = \{s_1, \ldots, s_{n_1}\}$ and $V_{\mathcal{F}} \backslash Inf_{\mathcal{F}} = \{s_{n_1+1}, \ldots, s_{n_1+n_2}\}$ 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 $Inf_{\mathcal{F}}$, and D the restriction of A^α to $V_{\mathcal{F}} \backslash Inf_{\mathcal{F}}$. 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.

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

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 $i, j \in V_{\mathcal{F}} \backslash Inf_{\mathcal{F}}$ $(i, a, j) \in \delta_{\mathcal{F}}$ 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. □

Total coverage in finite depth model. The following proposition gives an alternative recursive characterization of f_s^k; it is the analogon in the finite depth model of Proposition 8.

Proposition 11. *Let $\mathcal{F}$ be an FA and $k \in \mathbb{N}$ be a number. Then*

$$f_s^k(a\sigma) = \begin{cases} \sum_{s' : (a, s') \in \Delta(s)} f_{s'}^{k-1}(\sigma) & \text{if } a \in \Delta(s), |\sigma| \leq k, \\ r(s, a) & \text{if } a \notin \Delta(s), \sigma = \varepsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 12. *Let $\mathcal{F}$ be an FA and α a discount function for $\mathcal{F}$. Then the function $tc_k : S \to [0, 1], s \mapsto totcov(f_s^k)$ satisfies the following set of equations.*

$$tc_k(s) = \overline{r}(s) + \sum_{(a, s') \in \Delta(s)} tc_{k-1}(s')$$

Proof. As the proof of Proposition 9.

Optimal coverage.

Proposition 13. *Let $\mathcal{F}$ be an FA and let α be a discount function for $\mathcal{F}$ and $k \in \mathbb{N}$.*

- *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*

$$\begin{aligned} abscov(t, f_s^\alpha) &= \alpha(s, a, s') \cdot abscov(t', f_{s'}^\alpha) \\ abscov(t, f_s^k) &= abscov(t', f_{s'}^{k-1}) & \text{if } |t| \leq k + 1 \end{aligned}$$

- *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. Also, write $L^O \backslash \{b_1, \ldots b_n\} = \{c_1, c_2, \ldots c_m\}$. Let $t_1, t_2, \ldots t_n$ be test cases in states $s_1 \ldots s_n$ respectively. Write t for the test case $t = \{b_i!\sigma \mid \sigma \in t_i\} \cup \{c_1, c_2, \ldots c_m\}$. Then*

$$\begin{aligned} abscov(t, f_s^\alpha) &= \overline{r}(s) + \sum_{i=1}^{n} \alpha(s, b_i!, s_i) \cdot abscov(t_i, f_{s_i}^\alpha) \\ abscov(t, f_s^k) &= \overline{r}(s) + \sum_{i=1}^{n} abscov(t_i, f_{s_i}^k) & \text{if } |t| \leq k + 1 \end{aligned}$$

Proof. We give the proof for f_s^α; the one for f_s^k is similar.

- $$\begin{aligned} abscov(t, f_s^\alpha) &= \sum_{\sigma \in t} f_s^\alpha(\sigma) \\ &= \sum_{\sigma' \in t'} f_s^\alpha(a\sigma') && \text{(Proposition 8)} \\ &= \sum_{\sigma' \in t'} \alpha(s, a, s') \cdot f_{s'}^\alpha(\sigma') \\ &= \alpha(s, a, s') \cdot abscov(t', f_{s'}^\alpha) \end{aligned}$$

- $$\begin{aligned} abscov(t, f_s^\alpha) &= \sum_{\sigma \in t} f_s^\alpha(\sigma) \\ &= \sum_{c \notin \Delta(s)} f_s^\alpha(c) + \sum_{i=1}^{n} \sum_{\sigma' \in t_i} f_s^\alpha(b_i\sigma') && \text{(Proposition 8)} \\ &= \overline{r}(s) + \sum_{i=1}^{n} \sum_{\sigma' \in t_i} \alpha(s, b_i, s_i) \cdot f_{s_i}^\alpha(\sigma') \\ &= \overline{r}(s) + \sum_{i=1}^{n} \alpha(s, b_i, s_i) \cdot abscov(t_i, f_{s_i}^\alpha) \end{aligned}$$

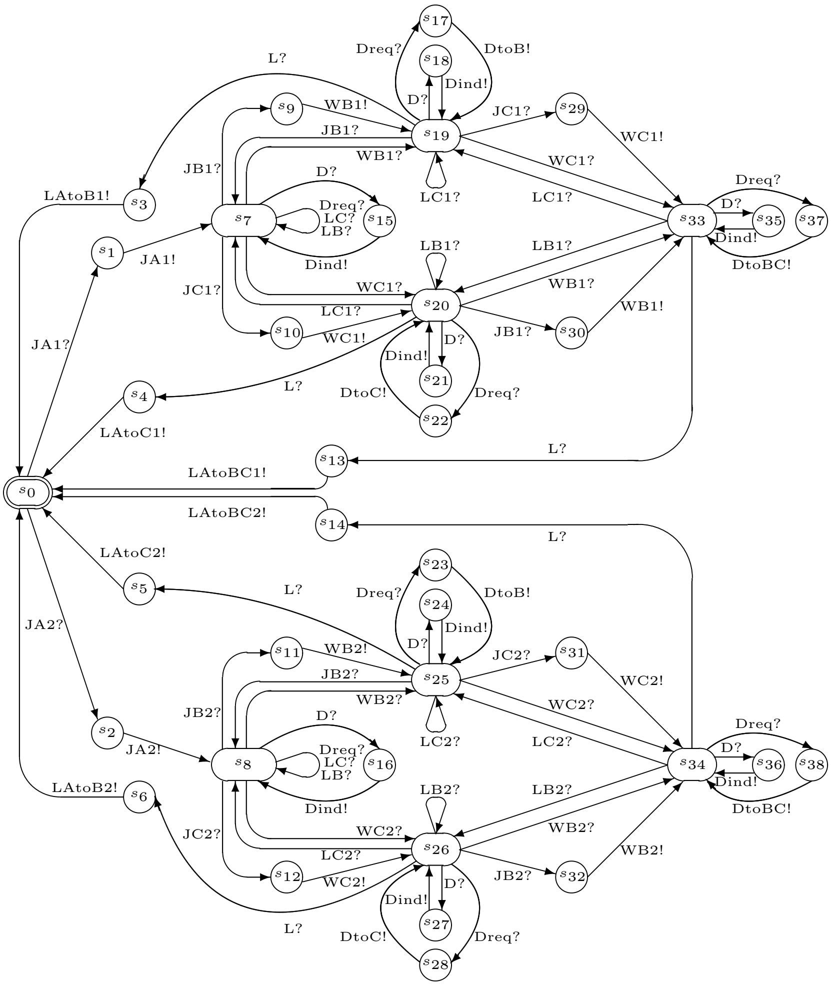

Fig. 6. Chat Protocol with two chats. L = leave, J = join, D = data and W = answer

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

Fig. 7. Error names and errors values

state	$\overline{r}$	d	α_1	α_2	α_3	state	$\overline{r}$	d	α_1	α_2	α_3
s_0	64	3	1/8	0.323	0.333	s_{20}	64	8	1/8	0.115	0.124
s_1	71	1	1/8	0.990	0.999	s_{21}	71	1	1/8	0.990	0.999
s_2	71	1	1/8	0.990	0.999	s_{22}	71	1	1/8	0.990	0.999
s_3	71	1	1/8	0.990	0.999	s_{23}	71	1	1/8	0.990	0.999
s_4	71	1	1/8	0.990	0.999	s_{24}	71	1	1/8	0.990	0.999
s_5	71	1	1/8	0.990	0.999	s_{25}	64	8	1/8	0.115	0.124
s_6	71	1	1/8	0.990	0.999	s_{26}	64	8	1/8	0.115	0.124
s_7	64	9	1/8	0.101	0.111	s_{27}	71	1	1/8	0.990	0.999
s_8	64	9	1/8	0.101	0.111	s_{28}	71	1	1/8	0.990	0.999
s_9	67	1	1/8	0.990	0.999	s_{29}	67	1	1/8	0.990	0.999
s_{10}	67	1	1/8	0.990	0.999	s_{30}	71	1	1/8	0.990	0.999
s_{11}	67	1	1/8	0.990	0.999	s_{31}	67	1	1/8	0.990	0.999
s_{12}	67	1	1/8	0.990	0.999	s_{32}	67	1	1/8	0.990	0.999
s_{13}	71	1	1/8	0.990	0.999	s_{33}	64	6	1/8	0.156	0.166
s_{14}	71	1	1/8	0.990	0.999	s_{34}	64	6	1/8	0.156	0.166
s_{15}	71	1	1/8	0.990	0.999	s_{35}	71	1	1/8	0.990	0.999
s_{16}	71	1	1/8	0.990	0.999	s_{36}	71	1	1/8	0.990	0.999
s_{17}	71	1	1/8	0.990	0.999	s_{37}	67	1	1/8	0.990	0.999
s_{18}	71	1	1/8	0.990	0.999	s_{38}	71	1	1/8	0.990	0.999
s_{19}	64	8	1/8	0.115	0.124	s_{39}	71	1	1/8	0.990	0.999

Fig. 8. The accumulated weights of erroneous outputs ($\overline{r}$), the out-degree (d) and the different discounts α per state

	tc
α_1	99.134
α_2	511.369
α_3	743.432

ack	$k = 2$	$k = 4$	$k = 50$
α_1	89.750	97.171	99.134
α_2	130.607	239.025	510.768
α_3	132.652	249.320	733.540

rck	$k = 2$	$k = 4$	$k = 50$
α_1	91%	98%	100%
α_2	25%	47%	100%
α_3	18%	34%	99%

Fig. 9. Total coverage (tc) for different discount functions; absolute (ack) and relative (rck) coverage of the test suite containing all tests of length k

	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
$k = 30$	15.3%	4.6%	14.0%	5.3%	15.3%	4.6%	14.2%	8.5%	15.3%	4.9%
$k = 35$	14.1%	15.3%	15.3%	8.5%	8.6%	5.3%	15.3%	8.5%	8.5%	4.9%
$k = 40$	5.3%	14.0%	14.2%	15.3%	5.3%	14.1%	15.3%	5.3%	14.0%	15.3%
$k = 45$	5.0%	8.5%	14.0%	5.0%	8.5%	15.3%	4.9%	15.3%	4.5%	14.2%
$k = 50$	5.3%	14.2%	5.3%	4.9%	14.0%	5.3%	14.2%	5.3%	14.0%	15.3%

	$k = 30$	$k = 35$	$k = 40$	$k = 45$	$k = 50$
test suite T^k	63.1%	69.1%	72.8%	47.2%	54.2%

Fig. 10. Relative coverage, as a percentage, of tests generated by TorX and the test suites of them, using α_2