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.
[9, 11, 10, 4, 8]) 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. [9]).

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 [?]; 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 [?]. 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. We define our coverage measures relative to these weighted fault models.
Since 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 [9] (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). 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 $\varepsilon$ and $|\sigma|$ denotes the length of a trace $\sigma \in L^*$. We use $L^+ = L^* \setminus \{\varepsilon\}$. For $\sigma, \rho \in L^*$, we say that $\sigma$ is a prefix of $\rho$ and write $\sigma \sqsubseteq \rho$, if $\rho = \sigma \rho'$ for some $\sigma' \in L^*$. If $\sigma$ is a prefix of $\rho$, then $\rho$ is a suffix of $\sigma$. We call $\sigma$ a proper prefix of $\rho$ and $\rho$ a proper suffix of $\sigma$ if $\sigma \sqsubseteq \rho$, but $\sigma \neq \rho$.

We denote by $\mathcal{P}(L)$ the power set of $L$ and for any function $f : L \rightarrow \mathbb{R}$, 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 over $L$ is a function $f : L^* \rightarrow \mathbb{R}^\geq$ such that $0 < \sum_{\sigma \in L^*} f(\sigma) < \infty$.

Thus, a weighted fault model $f$ assigns a non-negative error weight to each trace $\sigma \in L^*$. If $f(\sigma) = 0$, then $\sigma$ represents correct system behavior; if $f(\sigma) > 0$, then $\sigma$ 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 weighted fault model $f$ 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 [6], ioco test theory [9] and FSM testing [?].

Definition 2. Let $f : L^* \rightarrow \mathbb{R}^\geq 0$ 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. $f$ 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 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 $relcov(T, f)$ 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 weighted fault model $f : L^* \rightarrow \mathbb{R}^\geq 0$ if $relcov(T, f) = 1$.

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

Proposition 1. Let $f$ be a weighted fault model 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 case derivation from labeled input-output transition systems, following ioco testing theory [9]. It prepares for the next section that treats an automaton-based formalism for specifying weighted fault models.
3.1 Labeled input-output transition systems

Definition 4. A labeled input-output transition system (LTS) $\mathcal{A}$ is a tuple $(S, s^0, L, \Delta)$, 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 $a^I$ and elements of $L^O$ by $a^O$.
- $\Delta \subseteq S \times L \times S$ is the transition relation. We require $\Delta$ to be deterministic, i.e. if $(s, a, s') \in \Delta$, then $s' = s''$. The input transition relation $\Delta^I$ is the restriction of $\Delta$ to $S \times L^I \times S$ and the output transition relation $\Delta^O$ is the restriction of $\Delta$ 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 $\text{outdeg}(s) = |\Delta(s)|$ the outdegree 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, by adding a self loop $s \xrightarrow{\delta} s$ labeled with a special label $\delta$ to each quiescent state $s$, i.e. each $s$ with $\Delta^O(s) = \emptyset$ and considering $\delta$ as an output action. 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 $s_0 = s^0$ and, for all $1 \leq i \leq n$, we have $(s_{i-1}, a_i, s_i) \in \Delta$. We denote by paths $\pi \subseteq \mathcal{A}$ the set of all paths in $\mathcal{A}$ and by last($\pi$) = $s_n$ the last state of $\pi$.
- The trace of a path $\pi$, $\text{trace}(\pi)$, is the sequence $a_1 a_2 \ldots a_n$ of actions occurring in $\pi$. We write traces $\pi \subseteq \mathcal{A} = \{\text{trace}(\pi) \mid \pi \in \text{paths} \pi \subseteq \mathcal{A}\}$ for the set of all traces in $\mathcal{A}$.
- Let $\sigma \in L^*$ be any trace, not necessarily one from $\mathcal{A}$. We write $\text{reach}_k(\mathcal{A}, \sigma)$ for the set of states that can be reached in $\mathcal{A}$ in exactly $k$ steps by following $\sigma$, i.e. $s' \in \text{reach}_k(\mathcal{A}, \sigma) \text{ if } |\sigma| = k$ and there is a path $\pi \in \text{paths} \pi \subseteq \mathcal{A}$ such that $\text{trace}(\pi) = \sigma$ and last($\pi$) = $s'$. We write $\text{reach}_k(\mathcal{A}, \sigma)$ for the set of states that can be reached via trace $\sigma$ in any number of steps, i.e. $\text{reach}_k(\mathcal{A}, \sigma) = \cup_{k \in \mathbb{N}} \text{reach}_k(\mathcal{A}, \sigma)$; we write $\text{reach}_k(\mathcal{A}, \sigma)$ for the set of states that can be reached in $k$ number of steps, by following any trace, i.e. $\text{reach}_k(\mathcal{A}, \sigma) = \cup_{\sigma \in L^*} \text{reach}_k(\mathcal{A}, \sigma)$; and $\text{reach}(\mathcal{A}, \sigma)$ for the set of all reachable states in $\mathcal{A}$ and $\text{reach}(\sigma)$ contain as most one state, since $\mathcal{A}$ is deterministic.

As before, we leave out the subscript $\mathcal{A}$ if $\mathcal{A}$ 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, \( \text{paths}_{\mathcal{A}[s]} \) denotes the set of paths starting from state \( s \).

3.2 Test cases

Test cases for LTSs are based on ioco test theory [9]. 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 \( \sigma \), 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 \( \sigma \in \text{traces}_\mathcal{A} \), but \( \sigma a? / \in \text{trace}_\mathcal{A} \), then the behavior after \( \sigma a? \) is not specified in \( \mathcal{A} \), 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 \( \sigma \notin \text{traces}_\mathcal{A} \), then no proper suffix of \( \sigma \) is contained in \( t \)

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

• 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 \( T^k(\mathcal{A}) \) the set of all test cases for \( \mathcal{A} \) with length \( k \).

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 weighted fault model \( f \). However, not all weighted fault models 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 \( f : L^* \rightarrow \mathbb{R}^{\geq 0} \) be a weighted fault model. Then \( f \) is consistent with \( \mathcal{A} \) if \( L = L_\mathcal{A} \) and for all \( \sigma \in L^*_\mathcal{A} \) we have
- If \( \sigma \in \text{traces}_\mathcal{A} \), 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 weighted fault model consistent with \( \mathcal{A} \). Then, the set \( T(\mathcal{A}) \) 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 $t \in T(\mathcal{A})$ 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 $t = \bigcup_{1 \leq i \leq n} X_i$. Since $f$ is consistent with $\mathcal{A}$, the set $t$ is a test in $T(\mathcal{A})$. Clearly, $t$ contains $\sigma$. Now, Proposition 1 yields that $T(\mathcal{A})$ is complete for $f$. $\square$

4 Fault automata

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

**Definition 9.** A fault automaton (FA) $\mathcal{F}$ is a pair $\langle \mathcal{A}, r \rangle$, where $\mathcal{A}$ is a LTS and $r : S \times L^O \rightarrow \mathbb{R}^{\geq 0}$. 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 define $\overline{\mathcal{F}} : S \rightarrow \mathbb{R}^{\geq 0}$ as $\overline{\mathcal{F}}(s) = \sum_{a \in \Delta^O(s)} r(s, a)$. Thus, $r$ accumulates the weight of all the erroneous outputs in a state. We denote the components of $\mathcal{F}$ by $\mathcal{A}_F$ and $r_F$ and leave out the subscripts $\mathcal{F}$ if it is clear from the context. We lift all concepts (e.g. traces, paths, etcetera) and notations that have been defined for LTSs to FAs.

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

We consider two solutions to this problem. First, finite depth weighted 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 $\sigma$ is a trace of $\mathcal{F}$ ending in $s$, but $\sigma a!$ is not a trace in $\mathcal{F}$, then $\sigma a!$ gets weight $r(s, a!)$. 

7
Definition 10. Given a FA $\mathcal{F}$, and a number $k \in \mathbb{N}$, we define the function $f^k_{\mathcal{F}} : L^* \rightarrow \mathbb{R}^{\geq 0}$ by

$$f^k_{\mathcal{F}}(\varepsilon) = 0 \quad f^k_{\mathcal{F}}(\sigma a) = \begin{cases} r(s, a) & \text{if } s \in \text{reach}^k_{\mathcal{F}}(\sigma) \land a \in L_s^0 \\ 0 & \text{otherwise} \end{cases}$$

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

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

Proposition 2. Let $\mathcal{F}$ be a FA, and $k \in \mathbb{N}$. If there is an $i < k$ and a state $s \in \text{reach}^i_{\mathcal{F}}$ with $\tau(s) > 0$, then $f^k_{\mathcal{F}}$ is a weighted fault model 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 weights of all traces. To do so, only finitely many traces may have weight greater than $\varepsilon$, for any $\varepsilon > 0$. 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, a discounted 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 $\alpha$ small enough, then one can easily show that $\sum_{\sigma \in L_s} f(\sigma) < \infty$.

To be precise, we take $\alpha < \frac{1}{d}$, where $d$ is the branching degree of $\mathcal{F}$ (i.e. $d = \max_{s \in S} \text{outdeg}(s)$). 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

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

To obtain more flexibility, we allow the discount to vary per transition. That is, we work with a discount function $\alpha : S \times L \times S \rightarrow \mathbb{R}^{\geq 0}$ 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 $\alpha$ 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 — for these states we have finite total coverage anyway. Thus, if $\text{Inf}_{\mathcal{F}}$ is the set of all states in $\mathcal{F}$ with at least one outgoing infinite path, we require for all states $s$: $\sum_{a \in L, s' \in \text{Inf}_{\mathcal{F}}} \alpha(s, a, s') < 1$. 

8
Definition 11. Let $\mathcal{F}$ be a FA. The set $\text{Inf}_\mathcal{F} \subseteq \mathcal{S}_\mathcal{F}$ of states with at least one infinite path is defined as $\text{Inf}_\mathcal{F} = \{s \in \mathcal{S} \mid \exists \pi \in \text{paths}_\mathcal{F}[s] \cdot |\pi| > |\mathcal{S}|\}$. 

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

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

Definition 12. Let $\mathcal{F}$ be a FA. Then a discount function for $\mathcal{F}$ is a function $\alpha : \mathcal{S}_\mathcal{F} \times \mathcal{L}_\mathcal{F} \times \mathcal{S}_\mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ such that

- For all $s, s' \in \mathcal{S}_\mathcal{F}$, and $a \in \mathcal{L}_\mathcal{F}$ we have $\alpha(s, a, s') = 0$ iff $(s, a, s') \notin \Delta_\mathcal{F}$.
- For all $s \in \mathcal{S}_\mathcal{F}$, we have: $\sum_{a \in \mathcal{L}_\mathcal{F}, s' \in \text{Inf}_\mathcal{F}} \alpha(s, a, s') < 1$.

Definition 13. Let $\alpha$ be a discount function for the FA $\mathcal{F}$. Given a path $\pi = s_0a_1\ldots s_k$ 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 a FA, a state $s \in \mathcal{S}$, and a discount function $\alpha$ for $\mathcal{F}$. We define the function $f^\alpha_{\mathcal{F}} : \mathcal{L}^* \rightarrow \mathbb{R}_{\geq 0}$ by

$$f^\alpha_{\mathcal{F}}(\varepsilon) = 0$$

$$f^\alpha_{\mathcal{F}}(\sigma a) = \begin{cases} \alpha(\pi) \cdot r(s, a) & \text{if } s \in \text{reach}_\mathcal{F}(\sigma) \land a \in \mathcal{L}_O \land \text{trace}(\pi) = \sigma \\ 0 & \text{otherwise} \end{cases}$$

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

The following proposition states that $f^\alpha_{\mathcal{F}}$ is a weighted fault model 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 a FA and $\alpha$ a discount function for $\mathcal{F}$. If there is a state $s \in \text{reach}_\mathcal{F}$ with $r(s) > 0$, then $f^\alpha_{\mathcal{F}}$ is a weighted fault model consistent with $\mathcal{F}$.

Remark 1. 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 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 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 $F$ and any $u > 0$, 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 $1 - u$. That is, we show that for any given $k$ and $u$, there exists a discount function $\alpha$ such that the relative coverage of all test cases of length $k$ or shorter is less than $u$. This means that, to get coverage higher than $1 - u$, one needs test cases longer than $k$.

For technical reasons, we need the weight assignment function in the FA to be fair, i.e. all states in $\text{Inf}$ must be able to reach some state with a positive weight.

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

**Theorem 2.** Let $F$ be a FA with fair weight assignment. Then there exists a family of discount functions $\alpha_u$ for $F$ such that for all $k \in \mathbb{N}$

$$\lim_{u \downarrow 0} \text{relecov}(T_k(f^\alpha_u)_F, f^\alpha_u) = 0$$

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

$$\alpha_u : S \times L \times S \to (0, 1)$$

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

Here $\text{OutInf}(s) = \{(a, s') \in \Delta(s) | s' \in \text{Inf}_F\}$.

## 5 Algorithms

This section represents various algorithms for computing and optimizing coverage for a given FA, interpreted under the finite depth or discounted fault model.

In particular, Section 5.1 presents an algorithm 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 5.3 provides two optimization algorithms. The first one finds a test case of length $k$ with maximal coverage; the second one finds a test suite with $n$ test cases of length $k$ and maximal coverage.

In this section, we use the following notation. Recall that the $F[s]$ denotes the FA that is the same as $F$, but with $s$ as initial state. When $F$ is clear from the context, we write respectively $f^k_s$ and $f^\alpha_s$ for the weighted fault models $f^k_{F[s]}$ and $f^\alpha_{F[s]}$ derived from $F$.

Given a FA $F = \langle A, r \rangle$, we write $A_F$ for the multi-adjacency matrix of $A$, containing at position $(s, s')$ the number of edges between $s$ and $s'$, i.e. $(A_F)_{ss'} =$
\[
\sum_{(s,a,s') \in A} 1. \text{ If } \alpha \text{ is a discount function for } F, \text{ then } A^\alpha_F \text{ is a weighted version of } A_F, \text{ i.e. } (A^\alpha_F)_{ss'} = \sum_{a \in T} \alpha(s,a,s'). \text{ We omit the subscript } F \text{ if it is clear from the context.}
\]

5.1 Absolute coverage in a test suite

Given a FA \( F \), a discounting function \( \alpha \) for \( F \) and a test suite \( T = \{t_1, \ldots, t_k\} \), we desire to compute, as in Definition 2,

\[
\text{abscon}(T, f) = \text{abscon}(\cup_{t \in T} t, f)
\]

To this end, we write \( at \) for \( \{a\sigma \mid \sigma \in t\} \) where \( t \) is a test and \( a \) an action; and \( t + t' \) as the union \( t \cup t' \) of tests \( t \) and \( t' \). In this way we can write a test as: \( t = \varepsilon \) or \( t = at_1 \) in case \( a \) is an input or \( t = b_1t_1 + \cdots + b_nt_n \) when \( b_1, \ldots, b_n \) are output actions of \( F \). We call super-test (Stest) to \( t' = a_1t'_1 + \cdots + a_kt'_k + b_1t''_1 + \cdots + b_nt''_n \)
where \( a_i \) are inputs and \( b_i \) are all outputs, in this way any test is a Stest.

To compute the union: \( \cup_{t \in T} t \), we merge the test using the infix operator \( \uplus \). Then we add the error weight of all traces in \( \cup_{t \in T} t \) via the function \( ac \).

Merge set of tests. Given a set of tests \( \{t_1, \ldots, t_k\} \) merge is a function \( \uplus \): Stest \( \times \) test \( \rightarrow \) Stest. Let \( t' \) be a Stest, \( t' = a_1t'_1 + \cdots + a_kt'_k + b_1t''_1 + \cdots + b_nt''_n \)
and \( t \) be a test, then \( t = \varepsilon \) or \( t = at_1 \) or \( t = b_1t'_1 + \cdots + b_nt'_n \)

\[
t' \uplus t =
\begin{cases}
   a_1t'_1 + \cdots + a_j(t'_j \uplus t_1) + \cdots + a_kt'_k + b_1t''_1 + \cdots + b_nt''_n & \text{if } t = at_1 \wedge a = a_j \\
   a_1t'_1 + \cdots + a_kt'_k + b_1(t''_1 \uplus t_1) + \cdots + b_n(t''_n \uplus t_n) & \text{if } t = b_1t'_1 + \cdots + b_nt'_n \\
   t' + t & \text{otherwise}
\end{cases}
\]

Absolute coverage in a Stest. Given a Stest \( t \) of \( F \) and a state \( s \) on \( F \), then

\[
\text{ac}(\varepsilon, s) = 0 \quad \text{ac}(t, s) = \sum_{i=1}^{n} \text{aux}(a_it_i, s)
\]

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

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

Theorem 3. Given a FA \( F \), a state \( s \in V \), a number \( k \in \mathbb{N} \) and \( T \) a set of tests, then

- \( \text{abscon}(T, f^s) = \text{ac}(\varepsilon T, s) \)
- If \( k > \max \left| t \right| \) and \( \alpha(s, a, s') = 1 \) then \( \text{abscon}(T, f^s) = \text{ac}(\varepsilon T, s) \).
5.2 Total coverage algorithms

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

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

$$
tc(s) = \tau(s) + \sum_{a \in L, s' \in S} \alpha(s, a, s') \cdot tc(s') = \tau(s) + \sum_{s' \in S} A_s^\alpha \cdot tc(s') \quad (*)
$$

These equations express that the total coverage in state $s$ equals the weight $\tau(s)$ 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 Proposition 9 in the Appendix.

Proposition 10 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 4. Let $\mathcal{F}$ be a FA such that for all $s \in S$ there exists a state $s' \in \text{reach}_{\mathcal{F}[s]}$ with $\tau(s') > 0$, and let $\alpha$ be a discount function for $\mathcal{F}$. Then

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

Complexity. The complexity of the method above is dominated by matrix inversion, which can be computed in $O(|S|^3)$ with Gaussian elimination, $O(|S|\log_2^2)$ with Strassen’s method or even faster with more sophisticated techniques.

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

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

$$
tc_0(s) = 0
$$

$$
tc_{k+1}(s) = \tau(s) + \sum_{(a, s') \in \Delta(s)} tc_k(s') = \tau(s) + \sum_{a \in L, s' \in S} A_s^\alpha \cdot tc_k(s')
$$

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

$$
tc_0 = 0 \quad tc_{k+1} = \tau + Atc_k \quad (*)
$$

Theorem 5. Let be given a FA $\mathcal{F}$, a state $s \in S$ and a number $k \in \mathbb{N}$. If $\forall s \in S : \exists s' \in \text{reach}_{\mathcal{F}[s]} : \tau(s') > 0$, then
\[ t_{ck} = \sum_{i=0}^{k-1} A^i \tau \]

Note that for a state \( s \) in an arbitrary \( \mathcal{F} \), there exists a state \( s' \in \text{reach}_{\mathcal{F}_F}^k[\mathcal{s}] \) with \( \tau(s') > 0 \) iff \( (\sum_{i=0}^{k-1} A^i \tau)_s > 0 \).

**Complexity.** By using Theorem 5 with sparse matrix multiplication, or by iterating the equations just above it, \( t_{ck} \) can be computed in time \( O(k \cdot |\Delta| + |S|) \).

**Remark 2.** 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. \( \text{abs cov}(T_k, f_s^\alpha) \), where \( T_k \) is the set of all tests of length \( k \) in \( \mathcal{F} \). Writing \( t_{cdk}(s) = \text{abs cov}(T_k, f_s^\alpha) \), the recursive equations become

\[
t_{cd0}(s) = 0 \\
t_{cdk+1}(s) = \tau(s) + \sum_{a \in L, s' \in S} t_{cdk}(s') = \tau(s) + \sum_{a \in L, s' \in S} A_{s,s'}^\alpha \cdot t_{cdk}(s')
\]

and the analogon of Theorem 5 becomes

\[
t_{cdk} = \sum_{i=0}^{k-1} (A_{\alpha})^i \tau = (I - A^\alpha)^{-1} \cdot (I - (A^\alpha)^k) \cdot \tau
\]

The latter equality holds because \( I - A^\alpha \) is invertible. Thus, the computing \( t_{cdk} \) 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 \( 1 \) whose entries are all equal to 1 that \( A1 = 1 \). Hence, 1 is in the kernel of \( I - A \), so \( I - A \) is not invertible.

**Relative Coverage**

### 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 \( \Delta \) and having \( \alpha(s, a, s') = 0 \) otherwise. We call a function \( \alpha \) 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 \( \text{acopt}_k(s) = \max_{t \in \mathcal{T}} \{ \text{abs cov}(t, s) \} \). 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 \( \alpha(s, a?, s') \cdot \text{acopt}_k(s') \).

The best coverage that we can obtain by stimulating the IUT is given by \( \max_{(a', s') \in \Delta^2(s)} \alpha(s, a?, s') \cdot \text{acopt}_k(s') \).
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 \( \mathcal{F}(s) + \sum_{(b, s') \in \Delta^O(s)} \alpha(s, b, s') \cdot \text{acopt}_k(s') \). The optimal test \( \text{acopt}(s) \) of length \( k + 1 \) is obtained from by \( \text{acopt}_k \) 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 \( \alpha \), and test length \( k \in \mathbb{N} \). Then \( \text{acopt}_k \) satisfies the following recursive equations.

\[
\begin{align*}
\text{acopt}_0(s) &= 0 \\
\text{acopt}_{k+1}(s) &= \max \left( \mathcal{F}(s) + \sum_{(b, s') \in \Delta^O(s)} \alpha(s, b, s') \cdot \text{acopt}_k(s'), \ \max_{(a?, s') \in \Delta'(s)} \alpha(s, a?, s') \cdot \text{acopt}_k(s') \right)
\end{align*}
\]

The proof of this theorem follows from Proposition 5.

**Complexity.** Based on Theorem 6, we can compute \( \text{acopt}_k \) in time \( O(k \cdot (|S| + |\Delta|)) \).

**Proposition 5.** Let \( s \) be a state, let \((a?, s') \in \Delta'(s)\), and let \( t' \) be a test case in states \( s' \). Write \( t \) for the test case \( t = \{a? \sigma | \sigma \in t'\} \). Then

\[
\text{abskov}(t, s) = \alpha(s, a, s') \cdot \text{abskov}(t', s')
\]

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. Also, write \( \mathcal{L}^O \setminus \{b_1!, \ldots, b_n!\} = \{c_1, c_2, \ldots, c_m\} \). Let \( t_1, t_2, \ldots, t_n \) test cases in states \( s_1 \ldots s_n \) respectively. Write \( t \) for the test case \( t = \{b_i! \sigma | \sigma \in t_i\} \cup\{c_1, c_2, \ldots, c_m\} \). Then

\[
\text{abskov}(t, s) = \mathcal{F}(s) + \sum_{i=1}^{n} \alpha(s, b_i!, s_i) \cdot \text{abskov}(t_i, s_i)
\]

**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 \( \text{acopt}_k(s) > c \).

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 \( \text{stw}(s) = \text{sup}_{t \in \mathcal{F}} \text{abskov}(t, s) \), i.e. the maximal absolute weight of a single test case. Then \( \text{stw} \) is again characterized by a set of equations.
Theorem 7. Let \( F \) be a FA, and \( \alpha \) be a discount function for \( F \). Then \( stw \) is the unique solution of the following set of equations.

\[
stw(s) = \max_{(a?, s') \in \Delta^I(s)} \alpha(s, a?, s') \cdot stw(s'), \tau(s) + \sum_{(b?, s') \in \Delta^O(s)} \alpha(s, b?, s') \cdot stw(s')
\]

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

Theorem 8. Let \( F \) be a FA, and \( \alpha \) be a discount function. Then \( stw \) is the optimal solution of the following LP problem.

\[
\begin{align*}
\text{minimize} & \sum_{s \in S} stw(s) \\
\text{subject to} & \quad stw(s) \geq \alpha(s, a?, s') \cdot stw(s'), \quad (a?, s') \in \Delta^I(s) \\
& \quad stw(s) \geq r(s) + \sum_{(b?, s') \in \Delta^O(s)} \alpha(s, b?, s') \cdot stw(s') \quad s \in S
\end{align*}
\]

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 [?]. 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 \( T \), we can also derive the best \( n \) tests of length \( k \). The idea is as follows.

We write \( acptk_\mu(s) \) for the list \( \{abs cov(t, s)\} \), 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 \( acptk_\mu \) recursively.

Assume that all input actions are given by \( a_1, a_2, \ldots, a_m \). Consider a test suite \( T = \{t_1, t_2, \ldots, t_m\} \) consisting of \( km \) tests, which all have length \( k + 1 \) and start in state \( s \). Assume that each test \( t_i \) applies input \( a_i \), leading successor state \( s_i \). Let \( \{t_i, t_i', \ldots, t_i''\} \) be the \( n \) best 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 \( b_1, b_2, \ldots, b_l \). Now, consider the test suite \( T = \{t_1, t_2, \ldots, t_l\} \) consisting of \( l \) tests, of length \( k + 1 \) starting in state \( s \). Assume that each test \( t_i \) observes the IUT. If action \( b^O \) occurs, then successor state \( s_j \) is reached, applies the optimal test of length \( k \). The coverage of this test case is \( \tau(s) + \sum_{(b, s') \in \Delta^O(s)} \alpha(s, b, s') \cdot acptk_\mu(s') \). The optimal test \( acpt(\tau) \) of length \( k + 1 \) is obtained from by \( acptk_\mu \) by selecting from these options (i.e. inputting 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 $F$, a discount function $\alpha$ for $F$, a test length $k \in \mathbb{N}$, and a number $n \in \mathbb{N}$. Then $tc_k$ satisfies the following equations.

$$v_0(s) = [0, 0, \ldots, 0]$$

$$v_{j+1}(s) = \max n \{ \alpha(s, a, s') \cdot v | (a?, s') \in \Delta^k(s), v \leftarrow v_j(s') \} + r(s) \oplus \text{sumlist}(b!, s') \in \Delta(s) \alpha(s, b!, s') \otimes v_j(s') \}$$

(j < k)

Here, $x \oplus l$ adds the number $x \in \mathbb{R}$ to each element of the list $l$, i.e.,

$$x \oplus [e_1, e_2, \ldots, e_n] = [e_1 + x, e_2 + x, \ldots, e_n + x].$$

Similarly, $x \otimes l$ multiplies each list element with $x$. The operator sumlist yields the point wise summation of all the lists $l_i$. Thus, for $l_i = [e_{i1}, e_{i2}, \ldots, e_{in}]$, we have, $\text{sumlist}_{l_i} = [\sum e_{i1}, \sum e_{i2}, \ldots, \sum e_{in}]$.

Note that all the lists to which we apply this operator length $k$.

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 $c < \text{size}(s)$. 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, a chat protocol. This protocol was previously used as the conference protocol in [?].

The chat protocol provides a multi-cast service to users engaged in a 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. To be identified 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 1 displays an LTS modeling the chat protocol. This model considers two chat sessions and two users.

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 consider three different discount functions, \( \alpha_1, \alpha_2 \) and \( \alpha_3 \). Given a transition in the FA leaving from a state with outdegree \( n \), \( \alpha_1 \) assigns value \( \frac{1}{n} \) to this transition; \( \alpha_2 \) assigns \( \left( \frac{1}{n} - \frac{1}{100} \right) \) to it and \( \alpha_3 \) assigns \( \left( \frac{1}{n} - \frac{1}{10000} \right) \). 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 \( \alpha_2 \). For 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 \( t^k_1, \ldots t^k_{10} \) 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 \( t^k_i \) 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 Related work

There is a vast literature on syntactic test coverage criteria. [?].

Test coverage and optimization are well studied for (extended) finite state machines [?]. 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 [?]. 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 [?] 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 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 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

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

Theorem 10. Let $\mathcal{F} = \langle A, r \rangle$ be an FA with fair weight assignment. Then there exists a family of discount functions $\alpha_u$ for $\mathcal{F}$ such that for all $k \in \mathbb{N}$

$$\lim_{u \downarrow 0} \text{relcov}(T_k(f^{\alpha_u}_\mathcal{F}), f^{\alpha_u}_\mathcal{F}) = 0$$

Definition 18. Given an FA $\mathcal{F}$ and a number $u \in (0, 1)$, we define a discount function $\alpha_u : S \times L \times S \rightarrow (0, 1)$ by

$$\alpha_u(s, a, s') = \begin{cases} (1-u) & \text{if } (s, a, s') \in \Delta \text{ and } s' \in \text{Inf} \\ > 0 & \text{if } (s, a, s') \in \Delta \text{ and } s' \in S \setminus \text{Inf} \\ 0 & \text{otherwise} \end{cases}$$

Here $\text{OutInf}(s) = \{(a, s') \in \Delta(s) | s' \in \text{Inf}\}$. We usually write $A_u$ for the matrix $A^{\alpha_u}$.

Definition 19. Given an FA $\mathcal{F}$, we define the vector $1_{\text{Inf}}$ indexed by $s \in S$ by

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

Proposition 6. $1_{\text{Inf}}$ is an eigenvector of $A_u$ with eigenvalue $1 - u$, i.e.

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

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

$$(A_u \cdot 1_{\text{Inf}})_s = \sum_{s' \in S} (A_u)_{ss'} \cdot 1_{\text{Inf}}(s')$$

$$= \sum_{s' \in \text{Inf}_{\mathcal{F}}} (A_u)_{ss'}$$

$$= \sum_{s' \in \text{Inf}_{\mathcal{F}}} \sum_{a \in L} \alpha_u(s, a, s')$$

$$= \sum_{(a, s') \in \text{OutInf}(s)} \frac{(1-u)}{|\text{OutInf}(s)|}$$

$$= |\text{OutInf}(s)| \cdot \frac{(1-u)}{|\text{OutInf}(s)|}$$

$$= 1 - u$$

20
For $s \in S \setminus \text{Inf } F$ we get, using Proposition 3,
\[
(A_u \cdot 1_{\text{Inf}})_s = \sum_{s' \in S} (A_u)_{ss'} \cdot 1_{\text{Inf}}(s') \\
= \sum_{s' \in \text{Inf}} (A_u)_{ss'} \\
= \sum_{s' \in \text{Inf}} \sum_{a \in L} \alpha_u(s, a, s') \\
= \sum_{s' \in \text{Inf}} 0 \\
= 0
\]
\[\square\]

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

**Proof.** By induction on $n$. \[\square\]

**Proposition 7.** Let $F = (A, r)$ be an FA with a fair weight assignment $r$ then
\[
(\sum_{i=0}^{\lfloor |S|/2 \rfloor} A_i^t) \cdot \pi > 0 \quad \text{for every } s \in \text{Inf } F.
\]

**Proof.** Note that $(A_i^t)_{ss'} > 0$ implies that $s'$ can be reached from $s$ in $i$ transitions. As $F$ 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 $(A_i^t)_{ss'} > 0$. Hence $(\sum_{i=0}^{N} A_i^t)_{ss'} > 0$ for any pair of such $s, s' \in S$. By the definition of fair weight assignment all states $s \in \text{Inf } F$ can reach an $s' \in S$ with $\pi(s') > 0$. Thus we get $(\sum_{i=0}^{N} A_i^t) \cdot \pi = \sum_{s' \in S} (\sum_{i=0}^{N} A_i^t)_{ss'} \cdot \pi(s') > 0$. \[\square\]

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

**Proposition 8.** Let $F = (A, r)$ be an FA with fair weight assignment. Then for every $s \in \text{Inf } F$
\[
\lim_{u \downarrow 0} \text{relcov}(\mathcal{T}_k(f^{\alpha_u}_{f^2_{\mathcal{F}}}), f^{\alpha_u}_{f^2_{\mathcal{F}}}) = 0.
\]

**Proof.** Recall that
\[
\text{relcov}(\mathcal{T}_k(f^{\alpha_u}_{f^2_{\mathcal{F}}}), f^{\alpha_u}_{f^2_{\mathcal{F}}}) = \frac{\text{abscov}(\mathcal{T}_k(f^{\alpha_u}_{f^2_{\mathcal{F}}}), f^{\alpha_u}_{f^2_{\mathcal{F}}})}{\text{totcov}(f^{\alpha_u}_{f^2_{\mathcal{F}}})}
\]

As $\text{abscov}(\mathcal{T}_k(f^{\alpha_u}_{f^2_{\mathcal{F}}}), f^{\alpha_u}_{f^2_{\mathcal{F}}})$ is always finite, it suffices to show that $\lim_{u \downarrow 0} \text{totcov}(f^{\alpha_u}_{f^2_{\mathcal{F}}}) = \infty$. This can be shown as follows.

Define $r_{min} = \min_{s' \in \text{Inf }} \sum_{i=0}^{\lfloor |S|/2 \rfloor} A_i^t \cdot \pi_{s'}$. Then Proposition 7 yields that $r_{min} > 0$. Moreover, we have for all $s' \in S$ that
\[
\sum_{i=0}^{\lfloor |S|/2 \rfloor} A_i^t \cdot \pi_{s'} \geq r_{min} \cdot 1_{\text{Inf}}(s').
\]

21
Therefore,
\[ \text{totcov}(f^\alpha_s) = (\sum_{i=0}^{\infty} A^i_u \cdot \tau)_s \]
\[ = (\sum_{j=0}^{\infty} A^j_u |S| \cdot \sum_{i=0}^{-1} A^i_u \cdot \tau)_s \]
\[ \geq r_{\text{min}} \cdot \left( \sum_{j=0}^{\infty} A^j_u |S| \cdot 1_{\text{Inf}} \right)_s \]
\[ = r_{\text{min}} \cdot \left( \sum_{j=0}^{\infty} (1 - u)^j |S| \cdot 1_{\text{Inf}} \right)_s \]
\[ \geq r_{\text{min}} \cdot \frac{1 - (1 - u)^n}{1 - (1 - u)|S|} \]

As \( r_{\text{min}} > 0 \) and \( 1 - (1 - u)^n \) is of the order \( O(u) \), we get \( \lim_{u \downarrow 0} (\sum_{i=0}^{\infty} A^i_u \cdot \tau)_s = \infty \).

\[ \square \]

A.2 Correctness proofs

Proposition 9. Let \( F \) be an FA. Then the function \( tc : S \to [0, 1], s \mapsto \text{totcov}(f^\alpha_s) \) satisfies the following set of equations.

\[ tc(s) = \tau(s) + \sum_{a \in L, s' \in S} \alpha(s, a, s') tc(s') \]

Proof.
\[ tc(s) = \sum_{\sigma \in L^*} f^\alpha_s(\sigma) \]
\[ = f^\alpha_s(\varepsilon) + \sum_{a \in \Delta(s), \sigma \in L^*} f^\alpha_s(\sigma) + \sum_{a \notin \Delta(s), \sigma \in L^*} f^\alpha_s(\sigma) \quad \text{(Proposition 8)} \]
\[ = 0 + \sum_{a \in \Delta(s), \sigma \in L^*} \alpha(s, a, s') f^\alpha_s(\sigma) + \sum_{a \notin \Delta(s)} r(s, a) \]
\[ = \sum_{a \in L} \alpha(s, a, s') tc(s') + \tau(s) \]

\[ \square \]

Proposition 10. Let \( F \) be an FA such that for all states \( s \in S \) there is a state \( s' \in \text{reach}_F[s] \) with \( \tau(s) > 0 \). Let \( \alpha \) be a discount function for \( F \). Then, the matrix \( I - A^\alpha \) is invertible.

Proof. By reordering the states we can obtain \( \text{Inf}_F = \{ s_1, \ldots, s_{n_1} \} \) and \( V_F \setminus \text{Inf}_F = \{ s_{n_1+1}, \ldots, s_{n_1+n_2} \} \) with \( n_1 + n_2 = n = |V_F| \). Without loss of generality we may therefore assume that \( A^\alpha \) is of the form

22
\[
\begin{pmatrix}
B & C \\
0 & D
\end{pmatrix}
\]

with \( B \) the \( n_1 \times n_1 \) matrix that is the restriction of \( A^\alpha \) to \( \text{Inf} x \), and \( D \) the restriction of \( A^\alpha \) to \( V_F \setminus \text{Inf} x \). 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 \( Bv \).

\[
(Bv)_i = \sum_{j \leq n_1} B_{ij} v_j \\
\leq \sum_{j \leq n_1} B_{ij} \|v\|_\infty \\
= \|v\|_\infty \cdot \sum_{(j,a) \in \text{OutInf}(i)} \alpha(i,a,j) \quad \text{(Def of discount function)} \\
< \|v\|_\infty
\]

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 \( i,j \in V_F \setminus \text{Inf} x \) \((i,a,j) \in \delta x \) 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. \( \square \)
Fig. 1. Chat Protocol with two chats. L = leave, J = join, D = data and W = answer

<table>
<thead>
<tr>
<th>name of error</th>
<th>value</th>
<th>name of error</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>join.A.1.PDU!</td>
<td>3</td>
<td>leave.A.to.C.2.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>join.A.2.PDU!</td>
<td>3</td>
<td>leave.A.to.BC.1.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>answer.B.1!</td>
<td>7</td>
<td>leave.A.to.BC.2.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>answer.B.2!</td>
<td>7</td>
<td>dataind!</td>
<td>3</td>
</tr>
<tr>
<td>answer.C.1!</td>
<td>7</td>
<td>data.to.B.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>answer.C.2!</td>
<td>7</td>
<td>data.to.C.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>leave.A.to.B.1.PDU!</td>
<td>3</td>
<td>data.to.BC.PDU!</td>
<td>3</td>
</tr>
<tr>
<td>leave.A.to.B.2.PDU!</td>
<td>3</td>
<td>quiescent!</td>
<td>10</td>
</tr>
<tr>
<td>leave.A.to.C.1.PDU!</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. Error name and errors values
<table>
<thead>
<tr>
<th>state</th>
<th>tck</th>
<th>σ</th>
<th>tck</th>
<th>σ</th>
<th>tck</th>
<th>σ</th>
<th>tck</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>state0</td>
<td>64</td>
<td>1/8</td>
<td>0.323</td>
<td>0.333</td>
<td>state20</td>
<td>64</td>
<td>1/8</td>
<td>0.115</td>
</tr>
<tr>
<td>state1</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state21</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state2</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state22</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state3</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state23</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state4</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state24</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state5</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state25</td>
<td>64</td>
<td>1/8</td>
<td>0.115</td>
</tr>
<tr>
<td>state6</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state26</td>
<td>64</td>
<td>1/8</td>
<td>0.115</td>
</tr>
<tr>
<td>state7</td>
<td>64</td>
<td>9/16</td>
<td>0.101</td>
<td>0.111</td>
<td>state27</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state8</td>
<td>64</td>
<td>9/16</td>
<td>0.101</td>
<td>0.111</td>
<td>state28</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state9</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state29</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state10</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state30</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state11</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state31</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state12</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state32</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state13</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state33</td>
<td>64</td>
<td>1/8</td>
<td>0.156</td>
</tr>
<tr>
<td>state14</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state34</td>
<td>64</td>
<td>1/8</td>
<td>0.156</td>
</tr>
<tr>
<td>state15</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state35</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state16</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state36</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state17</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state37</td>
<td>67</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state18</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
<td>0.999</td>
<td>state38</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
<tr>
<td>state19</td>
<td>64</td>
<td>8/16</td>
<td>0.115</td>
<td>0.124</td>
<td>state39</td>
<td>71</td>
<td>1/8</td>
<td>0.990</td>
</tr>
</tbody>
</table>

**Fig. 3.** Value of \( r \), outdegree (od) and \( \sigma \) per state

<table>
<thead>
<tr>
<th>tc</th>
<th>tck, ( k = 2 )</th>
<th>tck, ( k = 4 )</th>
<th>tck, ( k = 50 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha_1 )</td>
<td>99.134</td>
<td>89.750</td>
<td>97.171</td>
</tr>
<tr>
<td>( \alpha_2 )</td>
<td>511.369</td>
<td>130.607</td>
<td>239.025</td>
</tr>
<tr>
<td>( \alpha_3 )</td>
<td>743.432</td>
<td>132.652</td>
<td>249.329</td>
</tr>
</tbody>
</table>

**Fig. 4.** Total coverage and maximal coverage of test with length \( k \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>test ( t^1 )</th>
<th>test ( t^2 )</th>
<th>test ( t^3 )</th>
<th>test ( t^4 )</th>
<th>test ( t^5 )</th>
<th>test ( t^6 )</th>
<th>test ( t^7 )</th>
<th>test ( t^8 )</th>
<th>tck, suite ( T^\alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>14.100</td>
<td>15.275</td>
<td>15.263</td>
<td>8.537</td>
<td>8.579</td>
<td>5.348</td>
<td>15.275</td>
<td>8.536</td>
<td>8.495</td>
</tr>
</tbody>
</table>

**Fig. 5.** Relative coverage, as a percentage, of tests with length \( k \) using \( \alpha_2 \).