TEST SELECTION, TRACE DISTANCE AND HEURISTICS

L.M.G. Feijs, N. Goga, S. Mauw*
Eindhoven University of Technology, P.O. Box 513, NL–5600 MB Eindhoven
feijs, goga, sjouke@win.tue.nl

J. Tretmans
University of Twente, P.O. Box 217, NL–7500 AE Enschede, The Netherlands
tretmans@cs.utwente.nl

Abstract

Two heuristic principles for test selection are proposed: *the reduction heuristic* and *the cycling heuristic*. The first assumes that few outgoing transitions of a state show essentially different behaviour. The second assumes that the probability to detect erroneous behaviour in a loop decreases after each correct execution of the loop behaviour. We formalize these heuristic principles and we define a coverage function which serves as a measure for the error-detecting capability of a test suite. For this purpose we introduce the notion of a *marked trace* and a distance function on such marked traces.

Keywords: Test selection, test coverage, trace distance, heuristics, edit distance.

1. INTRODUCTION

The selection of an appropriate set of tests from all possible ones (usually infinitely many test cases), is not a trivial task. We refer to this task as *test selection*. Traditionally, test selection is based on a number of heuristic criteria. Well-known heuristics include equivalence partitioning, boundary value analysis, and use of code-coverage criteria like statement-, decision- and path-coverage [9]. Although these criteria

*This research was supported by the Dutch Technology Foundation STW under project STW TIF.4111: *Côte de Resyste* – COnformance TEsting of REactive SYSTEms; URL: http://fmt.cs.utwente.nl/CdR.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35497-2_31

I. Schieferdecker et al. (eds.), *Testing of Communicating Systems XIV*

provide some heuristics for selecting test cases, they are rather informal and they do not allow to measure the error-detecting capability of a test suite.

If test cases are derived from a formal specification, in particular if it is done algorithmically using tools for automatic test generation, e.g., AUTOLINK [10], TGV [8] or TORX [3], then the test selection problem is even more apparent. These test tools can generate a large number of test cases, when given a specification in the appropriate formalism, without much user intervention. All these generated test cases can detect potential errors in implementations, and errors detected with these test cases indeed indicate that an implementation is not correct with respect to its specification. However, the number of candidate test cases may be very large, or even infinite. In order to control and get insight in the selection of the tests, and by that get confidence in the correctness of an IUT that passes the tests, it is important that the selection process is formally described and based on a well-defined strategy.

This paper approaches the problem of test selection by making assumptions in an automata-based, or labelled transition system-based formalism. Two different kinds of assumptions are considered in this paper, starting with the ideas of [5]. The first one, called *reduction heuristic*, assumes that few outgoing transitions of a state show essentially different behaviour. The second one, referred to as *cycling heuristic*, assumes that the probability to detect erroneous behaviour in a loop decreases after each correct execution of the loop behaviour.

Section 2 introduces the labeled transition systems and automata. After that we propose a mathematical framework, defining a heuristic as a function on the set of behaviours (traces). This is done in Section 3. When we want to make the two heuristics more precise, defining them as functions according to the definition from Section 3, we observe that an appropriate behaviour representation for them is needed. Therefore in Section 4 we define the *marked trace* representation. After these preparations the definitions of the heuristics as functions on marked traces are easy (Section 5). Subsequently, the notion of isolation and closeness of errors is formalized in Section 6 by defining a *distance* function between behaviours. This idea is taken from [1, 2] and extended to *marked traces*. The trace distance implements the considered heuristics in the sense that the traces which are selected by the heuristics are remote from each other. Every trace which is excluded by the heuristics is close to one of the selected traces. A *coverage function* which may serve as a measure for the error-detecting capability of a test suite is defined based on the maximum distance between selected and non-selected behaviours and a formula for approximating the coverage is given in Section 7.

2. PRELIMINARIES

The basic formalism for our discussion about test selection is the labelled transition system. A labelled transition system provides means to specify, model, analyze and reason about (concurrent) system behaviour.

A *labelled transition system* is a 4-tuple $\langle Q, L, T, q_0 \rangle$, where Q is a non-empty set of *states*, L is a set of *labels*, $T \subseteq Q \times L \times Q$ is the *transition relation*, and $q_0 \in Q$ is the *initial state*. The labels in L represent the actions of a system. An action $a \in L$ is executable in state $q \in Q$ if $(q, a, q') \in T$ for some state $q' \in Q$, which is said to be the *new state* after execution of a; we also write $q \xrightarrow{a} q'$. A finite sequence of pairs $\langle$*state, action*$\rangle$ ending into a state is called a *path*. Similarly, a finite sequence of actions is called a *trace*. The set of all traces over L is denoted by L^*, with ϵ denoting the empty sequence. Abusing notation, we will use p to denote both the labelled transition system and the current (or initial) state of the system.

The traces of a labelled transition system p are all sequences of actions that p can execute from its initial state q_0: $traces(p) =_{\text{def}} \{ \sigma \in L^* \mid q_0 \xrightarrow{\sigma} \}$. Here we use the following additional definitions:

$$q \xrightarrow{a_1 \cdot \ldots \cdot a_n} q' \quad =_{\text{def}} \quad \exists q_0, \ldots, q_n : \; q = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} q_n = q'$$

$$q \xrightarrow{\sigma} \quad =_{\text{def}} \quad \exists q' : \; q \xrightarrow{\sigma} q'$$

For our presentation and formalization we use *minimal, deterministic, finite-state* transition systems. A finite-state labelled transition system has a finite number of states, i.e., Q is finite. A transition system is deterministic if for any state $q \in Q$ and action $a \in L$ there is at most one successor state, i.e., $T : Q \times L \rightarrow Q$ is a (partial) function. A transition system is minimal if there are no equivalent states, i.e., no two states with exactly the same traces, which means: $\nexists q, q' \in Q : traces(q) = traces(q')$. We (ab)use the word *automaton* for these minimal, deterministic, finite-state transition systems.

Although it may seem a severe limitation to restrict to automata, an important formal test theory, viz. **ioco**-testing [12], can be expressed in terms of so-called *suspension automata*, which are deterministic. So the test selection approach which is presented in this paper can be integrated with **ioco**-testing.

In testing, the traces of the automata are used. A complete (maximal) test suite for an automaton specification s is expressed as $traces(s)$. However, even if s is finite-state, its set of traces will usually be infinite and contain traces of unbounded length. Such a test suite can never be executed within any reasonable limits of time and resources. Consequently, the problem of *test selection* consists of selecting a finite sub-

set $T \subseteq traces(s)$, such that we end up with a reasonably sized set of bounded-length test cases.

The challenge of test selection now is to choose T such that the resulting test suite still has a large error-detecting capability. Moreover, we wish to quantify this capability in order to compare and select test suites. The next sections will present and formalize an approach to selection and quantification.

3. THE TRACE DISTANCE AND THE TEST HEURISTICS

In our test selection method we use heuristics which are applied on traces and distances between traces. This section describes the formal definitions of these notions.

In a formal way a trace heuristic is a function between two sets of traces such that the range is a proper subset of the domain (so the heuristic reduces the size of the initial set).

Definition 1 *Let T be a set of traces.*
i) A trace heuristic h is a function $h : T \longrightarrow T$, where $Ran(h) \subset T$.
ii) A function $d : T \times T \longrightarrow \mathbb{R}_{\geq 0}$ is a trace distance iff: 1) $d(x, x) = 0$; 2) $d(x, y) = d(y, x)$; 3) $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in T$.

The pair (T, d) is a metric space. It is customary to express coverages by numbers in the range $[0, 1]$ and therefore we restrict ourselves, without loss of generality, to trace distance functions such that $0 \leq d(x, y) \leq 1$ for all x, y. In order to use a trace distance for test selection the concepts of ε-cover and total boundedness are useful.

Definition 2 *A set T' is an ε-cover of T ($T' \subseteq T, \varepsilon \geq 0$) if for every $t \in T$ there exists $t' \in T'$ such that $d(t, t') \leq \varepsilon$. A metric space (T, d) is totally bounded if for every $\varepsilon > 0$ it is possible to find a finite set $T_\varepsilon \subseteq T$ such that T_ε is an ε-cover of T with respect to distance d.*

Now a link between a heuristic and a trace distance is established: if the subset obtained by the application of that heuristic is an ε-cover of the original set, then the trace distance implements the heuristic.

Definition 3 *Let T be a set of traces and h be a trace heuristic such that $h : T \longrightarrow T$. Let d be a trace distance defined on T. Then d implements the heuristic h iff: $\exists \varepsilon_h \geq 0 :$ Ran(h) is an ε_h-cover of T with respect to the distance d.*

The following definition shows how to obtain the coverage.

Definition 4 *Let T be a set of traces and $T' \subseteq T$ an ε-cover of T with respect to a trace distance d. Let $\varepsilon_m = inf\{\varepsilon \geq 0 \mid T' \text{ is an } \varepsilon\text{-cover of } T\}$ be the inferior minimum of the ε values. Then the coverage of T' with respect to T is $cov(T', T) = 1 - \varepsilon_m$.*

4. THE MARKED TRACE REPRESENTATION

When we want to make the two heuristics more precise, defining them as functions according to Definition 1 point i), we observe that an appropriate trace representation is needed. When we apply the *Cycling* heuristic on a trace, we observe that the trace does not have enough information regarding how it was generated, what states it has been going through and how often it went through them. As a result, we will represent the trace in such a way that the information regarding its generation from the automaton will be included. This leads us to a concept called *marked traces*. In the marked trace representation, we associate the cycles with how many times a trace is traversing a state. The name of the state, which is seen as a mark, will serve as the identifier of the cycle. Also we will include the number of cycles through a state.

Definition 5 *Let L be a labelset and Q a set of states. Then a marked trace is inductively defined by: 1) $a \in L$, ϵ and $[\ ]^{0,q}$ ($q \in Q$) are marked traces; 2) if u ($u \in L$ or $u = [\_]^{\neg -}$) and v are marked traces then uv is a marked trace; 3) if u and $[\sigma]^{n,q}$ ($n \in \mathbb{N}, q \in Q$) are marked traces then $[\sigma\langle u\rangle]^{n+1,q}$ is a marked trace (σ is a sequence of type $\langle\sigma_1\rangle...\langle\sigma_n\rangle$ where σ_i are marked traces, $i = 1, ...n$).*

Example $a_0[\langle bd\rangle]^{1,II}e$, $a_0[\langle bd\rangle\langle bd\rangle]^{2,II}e$ with $II \in Q$ are marked traces.

We will denote the set of all the marked traces over a labelset L and a set of marks Q as L^*_Q. The transformation between the marked representation of a trace and a normal representation of a trace can be made easily by eliminating all the parentheses which occur in the marked representation. For example the marked trace $a_0[\langle bd\rangle\langle bd\rangle]^{2,II}e$ is transformed into the trace a_0bdbde.

In the following example, we will illustrate a way in which a trace of a particular automaton can be transformed into a marked trace. In general, this transformation is not unique.

Example Consider the specification automaton from Figure 1. The labelset is $L = \{b, c, d, e, f\} \cup \{a_i \mid i \in \mathbb{N}\}$ and the initial state is the state *I*. Via a transition a_i from the initial state, one arrives at *II*. This state contains a cycle which goes via *III* using the transitions b and d; the state *III* contains another cycle, via the transition c. From *II* one arrives at *IV* using e or f.

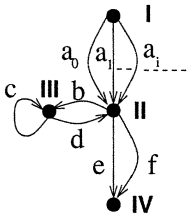

Figure 1. A minimal automaton of a specification

Let us consider the trace $a_0bcdbcde$. Adding boxes to reflect nesting structure, the corresponding path is $Ia_0\boxed{II}b\boxed{\boxed{III}}c\boxed{\boxed{III}}d\boxed{II}b\boxed{\boxed{III}}c\boxed{\boxed{III}}d\boxed{II}eIV$. The state II (surrounded with a box in the path) is repeated three times. Between two occurrences of state II in the path, the state III (surrounded with two boxes) appears twice. If we match every new occurrence of state II in the path with its first occurrence (we will call this way of matching the states *first state matching*), the path will be divided in 4 paths: $\underbrace{Ia_0II}_{1}\underbrace{[\langle IIbIIIcIIIdII\rangle}_{2}\underbrace{\langle IIbIIIcIIIdII\rangle]^{2,II}}_{3}\underbrace{IIeIV}_{4}$. If we do the same for III in the paths 2) $IIbIII\ cIIIdII$ and 3) $IIbIII cIIIdII$ and eliminate all the states, we obtain the marked trace $a_0[\langle b[\langle c\rangle]^{1,III}d\rangle\langle b[\langle c\rangle]^{1,III}d\rangle]^{2,II}e$. This marked trace corresponds to the initial trace $a_0bcdbcde$. However, there are also other ways of transforming it into a marked trace. For example, the states of the same trace can be grouped in another way as $Ia_0IIb\boxed{III}c\boxed{III}dIIb\boxed{III}c\boxed{III}dIIeIV$ and this trace has another correspondent marked trace which is $a_0b[\langle c\rangle\langle db\rangle\langle c\rangle]^{3,III}de$.

For this example we see that there is not a unique way of transforming a trace in a marked trace. We leave it as an option to the implementer (the user of our theory) to choose the way by which he transforms a trace into a marked trace. We will give a particular way to implement the transformation of a trace in a marked trace and a way to obtain the set of marked traces. If the implementer chooses another transformation, he can still use the theory presented in this paper if the correspondence between marked traces and traces is unique, and if its set of marked traces respects the property that the widths and the nesting depth are uniformly bounded (we will come to this later).

We obtain a set of marked traces by applying the following function (ALG) on each trace (path) of a finite-state minimal deterministic automaton s. The function builds a marked trace from a trace using a first state match technique like the one we used for the trace $a_0bcdbcde$

at the beginning of the previous example. In ALG we use the following operator, function and procedure: 1) the operator $|_{trace}$ transforms a path to a trace by eliminating all the states which appear in the path and keeping all the labels; 2) the function $NotRepetitivesState(p, Q)$, p a path, Q a set of states, returns true if every state of p which is contained in Q occurs only once in p and 3) the procedure $Divide(p, Q, q, p_1, ..., p_n)$ finds $q \in Q$ and splits p in $p_1, ..., p_n$ ($n \in \mathbb{N}$, $i = 2, ..., n-1$, $p_1 q$, $qp_i q$, qp_n paths) such that: i) $q \in Q$ is the first repetitive state in p, ii) $p = p_1 q p_2 q ... q p_n$ and iii) the set of states of p_j does not contain q ($j = 1, ..., n$).

function ALG (p : *Path*, Q : *SetStates*) : *MarkedTrace;*
var q : *State;*
$p_1, ..., p_n$: $(\epsilon + Label)(State\ Label)^*(\epsilon + State)$;
begin
if ($NotRepetitiveState(p, Q)$) **then**
(1) **return** $p\,|_{trace}$;
else
(2) $Divide(p, Q, q, p_1, ..., p_n)$;
(3) $Q = Q \setminus \{q\}$;
(4) **return** $ALG(p_1 q, Q)[\langle ALG(qp_2 q, Q)\rangle ... \langle ALG(qp_{n-1} q, Q)\rangle]^{n-2,q}$ $ALG(qp_n, Q)$;
end

Example Let us consider the path $p = Ia_0 IIbIIIdIIeIV$ of the automaton from Figure 1 and the automaton set of states $Q = \{I, II, III, IV\}$. The call of $ALG(Ia_0 IIbIIIdIIeIV, \{I, II, III, IV\})$ implies

Apply (2) $Divide(Ia_0 IIbIIIdIIeIV, \{I, II, III, IV\}, II, Ia_0, bIIId, eIV)$
where $q = II$ and $p_1 = Ia_0$, $p_2 = bIIId$, $p_3 = eIV$

Apply (3) $Q = \{I, II, III, IV\} \setminus \{II\} = \{I, III, IV\}$

Apply (4) $ALG(Ia_0 II, \{I, III, IV\})[\langle ALG(IIbIIIdII, \{I, III, IV\})\rangle]^{1,II}$
$ALG(IIeIV, \{I, III, IV\}) \overset{\text{Apply (1)}}{=} Ia_0 II\,|_{trace}\,[\langle IIbIIId$
$II\,|_{trace}\rangle]^{1,II} IIeIV\,|_{trace} = a_0[\langle bd\rangle]^{1,II} e$

The set of marked traces is $traces^m(s) = \{ALG(p, Q) \mid p \in path(s)\}$. In the remainder of this paper we will work with marked traces generated with ALG in place of traces.

As one can see, our way of building the set of marked traces is rather complex. One can imagine trivial solutions as for example: every marked trace is the trace itself. But the marked traces built with ALG have nice properties which are required for the application of our test selection theory. For example the width of such marked traces is uniformly bounded (Lemma 7), a property which is used for proving the total boundedness property (a direct consequence of Theorem 13). The marked traces generated with the trivial solution do not have this property.

In conclusion, once we have the set of marked traces, we want to know if it has some specific properties. So for a marked trace of an automaton we want to know if the width of it is uniformly bounded (Lemma 7), and if the nesting depth of it is also bounded (Lemma 9). Here uniformly bounded means that the same upperbound applies at all nesting levels.

Definition 6 *Let s be an automaton. Let L be the labelset and Q the set of states of s. Then the function width : $traces^m(s) \longrightarrow \mathbb{N}$ is : 1) if $a \in L, q \in Q$ then $width(a) = 1$, $width(\epsilon) = 0$, $width([\ ]^{0,q}) = 1$; 2) if u ($u \in L$ or $u = [\_]^{\neg -}$) and v are marked traces then $width(uv) = width(u) + width(v)$; 3) if u and $[\sigma]^{n,q}$ ($q \in Q, n \in \mathbb{N}$) are marked traces then $width([\sigma\langle u\rangle]^{n+1,q}) = 1$.*

Lemma 7 *The width of a marked trace generated with ALG from an automaton and the widths of all its component marked traces are less than or equal to $2m - 1$, where m is the number of the states of the automaton. (Without proof)*

Definition 8 *Let s be an automaton. Let L be the labelset and Q the set of states of s. Then the function nesting : $traces^m(s) \longrightarrow \mathbb{N}$ is: 1) if $a \in L, q \in Q$ then $nesting(a) = 0$, $nesting(\epsilon) = 0$, $nesting([\ ]^{0,q}) = 1$; 2) if u ($u \in L$ or $u = [\_]^{\neg -}$) and v are marked traces then $nesting(uv) = max(nesting(u), nesting(v))$; 3) if u and $[\sigma]^{n,q}$ ($q \in Q, n \in \mathbb{N}$) are marked traces then $nesting([\sigma\langle u\rangle]^{n+1,q}) = 1 + max(nesting(\sigma), nesting(u))$.*

Lemma 9 *The nesting depth of a marked trace generated with ALG from an automaton is less than or equal to the number of the states of the automaton. (Without proof)*

5. THE HEURISTICS DEFINED FOR MARKED TRACES

As we presented in the introduction, the intuition behind the heuristics *Reduction* and *Cycling* is that they take into account two aspects: the finiteness of 1) the number of outgoing transitions of certain states and of 2) the number of times each cycle can be traversed by every single trace.

When *Reduction* is applied, the labelset L is split in two parts: the selected labels which form a finite set $L' \subseteq L$ and the set of unselected labels which is $L \backslash L'$. This application can be seen as the application of a mapping function *trans*: $L \longrightarrow L'$ which maps every unselected label to a selected label from L' and every selected label to itself. One practical way to make the selection and to obtain L' and *trans* is by defining a distance d_L between labels, such that the metric space (L, d_L) is totally

bounded. Let us fix a positive real number $\varepsilon_L \geq 0$. Now L' will be a labelset which is an ε_L-cover of L. The labels which are remote from each other (their distance is greater than ε_L) are selected and the labels from $L \setminus L'$ remain unselected. The function *trans*: $L \longrightarrow L'$ can be defined in this case such that $trans(a) = b$ with $a \in L$ and $b \in L'$ has the minimum label distance to a.

For the *Cycling* heuristic we relate the cycles of the automaton to the marked representation of the trace; limiting the numbers of times of traversing the cycles means limiting the powers of the marked symbols in the marked traces. Now, let us define these heuristics in a formal way.

Definition 10 *Let s be an automaton. Let L be the labelset and Q the set of states of s. Let $L' \subseteq L$ be a finite subset of L and let trans: $L \longrightarrow L'$ be the mapping function. Let l_c be the cycle limit. Then*
i) The heuristic Reduction: $traces^m(s) \longrightarrow traces^m(s)$ is: 1) if $a \in L, q \in Q$ then $Reduction(a) = trans(a)$, $Reduction(\epsilon) = \epsilon$, $Reduction([\ ]^{0,q}) = [\ ]^{0,q}$; 2) if u ($u \in L$ or $u = [\_]^{\neg -}$) and v are marked traces then $Reduction(uv) = Reduction(u)\ Reduction(v)$; 3) if u and $[\sigma]^{n,q}$ ($q \in Q, n \in \mathbb{N}$) are marked traces then $Reduction([\sigma\langle u\rangle]^{n+1,q}) = [Reduction(\sigma)\ Reduction(u)]^{n+1,q}$.
ii) The heuristic Cycling: $traces^m(s) \longrightarrow traces^m(s)$ is: 1) if $a \in L, q \in Q$ then $Cycling(a) = a$, $Cycling(\epsilon) = \epsilon$, $Cycling([\ ]^{0,q}) = [\ ]^{0,q}$; 2) if u ($u \in L$ or $u = [\_]^{\neg -}$) and v are marked traces then $Cycling(uv) = Cycling(u)\ Cycling(v)$; 3) if u and $[\sigma]^{n,q}$ ($q \in Q, n \in \mathbb{N}$) are marked traces then: a) $Cycling([\sigma\langle u\rangle]^{n+1,q}) = [Cycling(\sigma)\ Cycling(u)]^{n+1,q}$, for $l_c > n$; b) $Cycling([\sigma\langle u\rangle]^{n+1,q}) = [Cycling(\sigma')]^{l_c,q}$, for $l_c \leq n$, where $\sigma = \langle\sigma_1\rangle...\langle\sigma_n\rangle$ and $\sigma' = \langle\sigma_1\rangle...\langle\sigma_{l_c}\rangle$ is obtained by cutting σ after l_c symbols.

Example Let us consider the automaton from Figure 1. For this automaton the set of labels is $L = \{c, b, d, e, f\} \cup \{a_i \mid i = 0, 1, ...\}$. Let $L' = \{a_0, c, b, d, e, f\}$ be a finite subset of L, $l_c = 2$ and *trans*: $L \longrightarrow L'$ $trans(x) = \begin{cases} a_0 & x = a_i, i \in \mathbb{N} \\ x & \text{otherwise} \end{cases}$. Then

i) $Reduction(a_3e) = Reduction(a_3)Reduction(e) = trans(a_3)trans(e) = a_0e$.

ii) $Cycling(a_0[\langle bd\rangle\langle bd\rangle\langle bd\rangle]^{3,II}e) = Cycling(a_0)\,Cycling([\langle bd\rangle\langle bd\rangle\langle bd\rangle]^{3,II})\ Cycling(e) = a_0[\langle bd\rangle\langle bd\rangle]^{2,II}e$.

6. THE TRACE DISTANCE FOR MARKED TRACES

In this section we make the trace distance more precise, defining it as a distance function according to Definition 1 point ii). As explained in

the introduction, this gives us an alternative formalization of the ideas behind the heuristics (they will be compared in Section 7). We will combine these ideas with another well-known idea, viz. the edit distance. Section 6.1 introduces the edit distance. After this preparation, the definition of the trace distance function can be given (Section 6.2).

6.1 The edit distance between strings

Because in our trace distance we use the concept of edit distance we shall present this first. Informally the edit distance is defined as the minimum number of insertions, deletions and substitutions required to transform one string into another. Wagner and Fisher ([11]) adopted different costs for the various atomic edit actions. According to Wagner-Fisher transforming a into a b costs $w(a,b)$. Extending this notation, $w(a,\epsilon)$ is the cost of deleting a and $w(\epsilon,b)$ is the cost of inserting b. The cost of editing is the sum of the costs of the atomic edit actions, and $d(x,y)$ is the minimum cost over all possible edit sequences that transform x into y.

Definition 11 *Let $w(a,b)$ be the weighting for the cost of transforming symbol a in symbol b, $w(a,\epsilon)$ be the cost of deleting a and $w(\epsilon,b)$ be the cost of inserting b. Of course $w(a,a) = 0$. Then the edit distance between the strings x and y is denoted as $ED(x,y)$ and it is computed as: 1) $ED(au,bv) = min(w(a,b)+ED(u,v), w(a,\epsilon)+ED(u,bv), w(\epsilon,b)+ED(au,v))$; 2) $ED(au,\epsilon) = w(a,\epsilon) + ED(u,\epsilon)$; 3) $ED(\epsilon,bv) = w(\epsilon,b)+ED(\epsilon,v)$; 4) $ED(\epsilon,\epsilon) = 0$ (where a,b are symbols and u,v are strings).*

6.2 Defining the trace distance

Our test selection technique uses two heuristics. For expressing these heuristics in the trace distance, it is important to remember that in the formalization of the *Reduction* heuristic a label distance was used. The incorporation of this heuristic in the trace distance is achieved in a simple way by using the label distance in the formula of the trace distance. Now a solution should be found for the *Cycling* heuristic.

For the *Cycling* heuristic we simply weight every level k of a cycling symbol (a marked trace of type $[\_]^{n,q}, n \in \mathbb{N}, q \in Q$) with a weight from a series of positive numbers p_k. This series should have the property that $\sum_{k=1}^{\infty} p_k = 1$. The logic behind this weighting is that summing the weights after a given limit (which is the cycle limit) will contribute with a small number reflecting our assumption that the first cycles are more important than the later cycles.

We will define the trace distance for all the possible combinations of the points (1), (2), (3) of Definition 5 (which are generating marked traces). We summarize these combinations below.

Between the marked traces generated with point (1) (such as $[\ ]^{0,q}, q \in Q$ and $a \in L$) we will define a distance function called *AtD* (atomic distance) because these are the atomic elements which form the marked trace; of course the *AtD* between two labels will be given by d_L, the distance between these labels; between a label and a marked trace such as $[\ ]^{0,q}$ it will be maximum (one) and between two marked traces such as $[\ ]^{0,q}$,$[\ ]^{0,q'}$ $(q, q' \in Q, q \neq q')$ it will also be one.

Between the marked traces generated with point (2) (such as *af* or *ae*) we will use a distance function called *EdD* (edit distance); we took this option because these traces are generated in a similar style as the strings are formed and it is quite natural to use it because it compares in a good way the terms which form the marked traces (for example in the traces $a_0 e$ and $a_0[\langle bd \rangle]^{1,II} e$ the edit distance will recognize that the labels a_0 and e from the first trace are present in the second trace).

Between the marked traces generated with point (3) (such as building $[\langle bd \rangle]^{1,II}$ once we know that $[\ ]^{0,II}$ is a marked trace) we employ the principle that cycles of different marks are very remote and hence have the maximum distance, i.e, 1; when dealing with cycles of the same mark we employ weighting factors p_k with the effect that the later iterations are considered less important than e.g. the first iteration; this can be done by using a function *EdDW* which is an edit distance for which the formula of Definition 11 is modified in such a way to take into account the weights.

The rest of the possible combinations such as (1) with (2), (2) with (3) etc. are defined in a similar style by using one of the techniques mentioned above (*EdD* or *AtD*).

We observe also that this trace distance is to be used in the computation of coverage which should be in the range $[0, 1]$. For simplifying the computation of coverage, we want the trace distance values to be in the range $[0, 1]$. This can be done by dividing all the above mentioned values (generated with an *EdD* or *AtD*) by the maximum width of the marked traces from $traces^m(s)$ (the maximum width is finite, see Section 4). For completing the picture it is necessary to add that the trace distance between a null trace (ϵ) and any other marked trace is maximum (1).

Now we have all the ingredients to define a trace distance on marked traces. We will call it d. In the definition, the distances already mentioned (*EdD* and *AtD*) will be used; also it is implicitly assumed that the definition is symmetric in the sense that $d(x, y) = d(y, x)$, x and y being marked traces and that $d(x, x) = 0$.

As explained above, the function AtD deals with the cases ϵ, $a \in L$ and $[\ ]^{0,-}$. We generalize it to marked traces of the form $[\_]^{-,-}$ as well.

Definition 12 *Let s be an automaton. Let L be the labelset of s, d_L the label distance defined on it and Q the set of states of s. The metric space (L, d_L) is totally bounded and d_L has all its values in the range $[0,1]$. Let l_m be the maximum of the width of the marked traces from traces$^m(s)$. Let p_k $(k = 1, 2, ...)$ be a series of positive numbers such that $\sum_{k=1}^{\infty} p_k = 1$. Then*

1. $d(a,b) = \frac{AtD(a,b)}{l_m}$; $d(a,\epsilon) = d(\epsilon,[\ ]^{0,q}) = 1$; $d(a,[\ ]^{0,q}) = \frac{AtD(a,[\ ]^{0,q})}{l_m}$; $d([\ ]^{0,q},[\ ]^{0,q'}) = \frac{AtD([\ ]^{0,q},[\ ]^{0,q'})}{l_m}$ *with $a, b \in L, q, q' \in Q$;*	2. $d(a,uv) = \frac{EdD(a,uv)}{l_m}$; $d(\epsilon,uv) = 1$; $d([\ ]^{0,q},uv) = \frac{EdD([\ ]^{0,q},uv)}{l_m}$ *with u ($u \in L$ or $u = [\_]^{-,-}$),* *v marked traces and $a \in L, q \in Q$;*
3. $d(a,[u\langle v\rangle]^{n+1,q}) = \frac{AtD(a,[u\langle v\rangle]^{n+1,q})}{l_m}$; $d(\epsilon,[u\langle v\rangle]^{n+1,q}) = 1$; $d([\ ]^{0,q},[u\langle v\rangle]^{n+1,q}) = \frac{AtD([\ ]^{0,q},[u\langle v\rangle]^{n+1,q'})}{l_m}$ *with v and $[u]^{n,q'}$ marked traces ($n \in \mathbb{N}$, $q' \in Q$) and $a \in L, q' \in Q$;*	4. $d(uv,rt) = \frac{EdD(uv,rt)}{l_m}$ *with u ($u \in L$ or $u = [\_]^{-,-}$),* *v, r ($r \in L$ or $r = [\_]^{-,-}$),* *t marked traces;*
5. $d(uv,[r\langle t\rangle]^{n+1,q}) = \frac{EdD(uv,[r\langle t\rangle]^{n+1,q})}{l_m}$ *with u ($u \in L$ or $u = [\_]^{-,-}$) ,$v, t, [r]^{n,q}$ marked traces ($n \in \mathbb{N}, q \in Q$);*	6. $d([u\langle v\rangle]^{n+1,q},[r\langle t\rangle]^{n'+1,q'})$ $= \frac{AtD([u\langle v\rangle]^{n+1,q},[r\langle t\rangle]^{n'+1,q'})}{l_m}$ *with $v, t, [u]^{n,q}, [r]^{n',q'}$ marked traces ($n, n' \in \mathbb{N}, q, q' \in Q$).*

where

$$AtD(x,y) = \begin{cases} d_L(x,y) & x,y \in L \\ EdDW(x,y) & x = [\_]^{n,q}, y = [\_]^{n',q}, q \in Q \\ & n, n' \in \mathbb{N}, n \neq 0, n' \neq 0 \\ 1 & otherwise \end{cases}$$

$EdD(uv,rt) = min(AtD(u,r) + EdD(v,t),\ AtD(u,\epsilon) + EdD(v,rt),$
$AtD(\epsilon,r) + EdD(uv,t));$
$EdD(uv,\epsilon) = AtD(u,\epsilon) + EdD(v,\epsilon);\ EdD(\epsilon,rt) = AtD(\epsilon,r) + EdD\ (\epsilon,t);$
$EdD(\epsilon,\epsilon) = 0$
with u ($u \in L$ or $u = [\_]^{-,-}$), v, r ($r \in L$ or $r = [\_]^{-,-}$), t marked traces;

$EdDW([\langle u_1\rangle ... \langle u_n\rangle]^{n,q}, [\langle v_1\rangle ... \langle v_p\rangle]^{p,q}) = EDW^1([\langle u_1\rangle ... \langle u_n\rangle]^{n,q}, [\langle v_1\rangle ... \langle v_p\rangle]^{p,q})$
$EDW^k([\langle u\rangle\sigma]^{h,q}, [\langle v\rangle\sigma']^{g,q}) = min(p_k \times d(u,v) + EDW^{k+1}([\sigma]^{h-1,q}, [\sigma']^{g-1,q}),$
$p_k \times d(u,\epsilon) + EDW^{k+1}([\sigma]^{h-1,q}, [\langle v\rangle\sigma']^{g,q}),$
$p_k \times d(\epsilon,v) + EDW^{k+1}([\langle u\rangle\sigma]^{h,q}, [\sigma']^{g-1,q}));$
$EDW^k([\langle u\rangle\sigma]^{h,q}, [\ ]^{0,q}) = p_k \times d(u,\epsilon) + EDW^{k+1}([\sigma]^{h-1,q}, [\ ]^{0,q});$
$EDW^k([\ ]^{0,q}, [\langle v\rangle\sigma']^{g,q}) = p_k \times d(\epsilon,v) + EDW^{k+1}([\ ]^{0,q}, [\sigma']^{g-1,q});$
$EDW^k([\ ]^{0,q}, [\ ]^{0,q}) = 0$
with u_i and v_j marked traces, $k, h, g \in \mathbb{N}$, $q \in Q$ and $u, v, [\sigma]^{h-1,q}, [\sigma']^{g-1,q}$ marked traces.

We add some explanation. It is easy to check that the definition of *EdD* and *EdDW* are copied from Definition 11 except for the fact that suitable weighting factors p_k have been incorporated. The parameter k in EDW^k indicates the position at which the next edit action takes place. Please note that the recursive definition of EDW^k is well defined because at least one of the right-hand sides of the equation is one symbol shorter than the corresponding left-hand side (therefore, the fact that k is increasing causes no problem). The base case is $EDW^k([\,]^{0,q}, [\,]^{0,q}) = 0$.

Example Let us consider the automaton from Figure 1. For this automaton the maximum width of the marked trace is 3. Let $p_k = \frac{1}{2^k}, k = 1, 2, \ldots$. Let d_L be the following label distance

$$d_L(x,y) = \begin{cases} 0 & x = y \\ \mid \frac{1}{4^{i+1}} - \frac{1}{4^{j+1}} \mid & x = a_i, y = a_j, i, j \in \mathbb{N} \\ 1 & \text{otherwise} \end{cases}$$

1) *The edit distance effect:*

Let us consider the traces $t_1 = a_0 e$ and $t_2 = a_0[\langle bd\rangle]^{1,II} e$.

$d(t_1, t_2) = \frac{1}{3} \times EdD(a_0 e, a_0[\langle bd\rangle]^{1,II} e) = \frac{1}{3} \times AtD(\epsilon, [\langle bd\rangle]^{1,II}) = \frac{1}{3}$.

The trace distance takes into account that the symbols a_0 and e from the first trace are present in the second trace.

2) *The cycling effect:*

Let us take the traces $t_1 = a_0[\langle bd\rangle]^{1,II} e$, $t_2 = a_0[\langle bd\rangle\langle bd\rangle]^{2,II} e$ and $t_3 = a_0[\langle bd\rangle\langle bd\rangle\langle bd\rangle]^{3,II} e$.

$d(t_1, t_2) = \frac{1}{3} \times EdD(a_0[\langle bd\rangle]^{1,II} e, a_0[\langle bd\rangle\langle bd\rangle]^{2,II} e) = \frac{1}{3} \times EdDW([\langle bd\rangle]^{1,II}, [\langle bd\rangle\langle bd\rangle]^{2,II}) = \frac{p_2}{3} = \frac{1}{12}$;

$d(t_1, t_3) = \frac{1}{3} \times EdD(a_0[\langle bd\rangle\langle bd\rangle]^{2,II} e, a_0[\langle bd\rangle\langle bd\rangle\langle bd\rangle]^{3,II} e) = \frac{1}{3} \times EdDW([\langle bd\rangle\langle bd\rangle]^{2,II}, [\langle bd\rangle\langle bd\rangle\langle bd\rangle]^{3,II}) = \frac{p_3}{3} = \frac{1}{24}$.

When two marked traces are cycling more times through the same cycle, the values of the trace distance start to decrease.

3) *The reduction effect:*

Let us take the traces $t_1 = a_0 e$, $t_2 = a_1 e$ and $t_3 = a_2 e$.

$d(t_1, t_2) = \frac{1}{3} \times EdD(a_0 e, a_2 e) = \frac{d_L(a_0, a_2)}{3} = \frac{0.23}{3}$;

$d(t_2, t_3) = \frac{1}{3} \times EdD(a_1 e, a_2 e) = \frac{d_L(a_1, a_2)}{3} = \frac{0.06}{3}$.

When the label distance between the labels (which compose the marked traces) is decreasing, the trace distance is also decreasing.

7. TRANSFORMING THE HEURISTICS INTO A COVERAGE

The trace distance formula depends on the label distance d_L which implements the *Reduction* heuristic and the weights p_k which implement the *Cycling* heuristic. On the other hand, by choosing for each automaton s a finite set $L_\varepsilon \subseteq L$ which is an ε_L-cover of L with respect to d_L

and a cycling limit l_c, a finite set of marked traces $T \subseteq traces^m(s)$ can be obtained. This is done by the application of the *Cycling* and the *Reduction* heuristics on $traces^m(s)$ by taking T =*Ran(Cycling* $\circ$ *Reduction)*. Now for this T and using d we want to approximate its ε-cover of $traces^m(s)$ so that we can compute the coverage $cov(T, traces^m(s))$ – see Definition 4. Intuitively ε should depend on ε_L and l_c. Its formula is given by Theorem 13.

Theorem 13 *Let s be an automaton. Let L be the labelset of s and d_L the label distance defined on it. The metric space (L, d_L) is totally bounded and d_L has all its values in the range $[0, 1]$. Let l_c be the cycle limit. Let p_k $(k = 1, 2, ...)$ be a series of positive numbers such that $\sum_{k=1}^{\infty} p_k = 1$. Let l_m be the maximum of the width and z the maximum of the nesting depth of the marked traces from $traces^m(s)$. Let $L_\varepsilon \subseteq L$ be an ε_L-cover of L. Then the finite set T =Ran(Reduction $\circ$ Cycling) of traces obtained by the application of the two heuristics on $traces^m(s)$ is an ε-cover of $traces^m(s)$ with $\varepsilon = \varepsilon^z$ and $\varepsilon^0 = \varepsilon_L$; for $i = 1, ..., z$: $\varepsilon_c^i = \sum_{k=1}^{l_c} p_k \times (max_{j=0,...,i-1}(\varepsilon^j)) + \sum_{k=l_c+1}^{\infty} p_k$; $\varepsilon^i = max_{\text{cycles}=0,...,l_m}(\frac{\text{cycles} \times \varepsilon_c^i + (l_m - \text{cycles}) \times \varepsilon_L}{l_m})$. (Without proof)*

Using the results of Theorem 13 it is easy to prove that the metric space $(traces^m(s), d)$ is a totally bounded metric space and that the trace distance d implements the *Cycling* and the *Reduction* heuristics, in the sense of Definition 3.

For the computation of the coverage we approximate the minimum ε_m from Definition 4 with the ε^z computed in Theorem 13. We will illustrate the computation of the coverage in the following example.

Example Consider the automaton from Figure 1. Let us fix the final state to be *IV*. Let us consider the reduced set $L_\varepsilon = \{a_0, b, c, d, e, f\}$ which is an ε_L-cover of the labelset L with $\varepsilon_L = 0.25$ (ε_L is computed with respect to the d_L defined in the example from Section 6.2). For this automaton the maximum width is $l_m = 3$ and the maximum nesting depth is $z = 2$. Let us fix the series $p_k = \frac{1}{2^k} (k \in \mathbb{N})$ and in the beginning $l_c = 1$. Then the set of traces T which is obtained by the application of the heuristics *Reduction* and *Cycling* is an ε-cover of $traces^m(s)$, with ε computed with the formula from Theorem 13 as:

1) $l_c = 1$, $L_\varepsilon = \{a_0, b, c, d, e, f\}$

$\varepsilon^0 = \varepsilon_L = 0.25$;

$\varepsilon_c^1 = \sum_{k=1}^{1} p_k \times \varepsilon^0 + \sum_{k=l_c+1}^{\infty} p_k = \frac{0.25}{2} + \sum_{k=l_c+1}^{\infty} \frac{1}{2^k} = 0.63$;

$\varepsilon^1 = max_{\text{cycles}=0,...,3}(\frac{\text{cycles} \times \varepsilon_c^1 + (l_m - \text{cycles}) \times \varepsilon_L}{l_m}) = 0.63$; $\varepsilon = \varepsilon^2 = 0.81$;

The coverage is computed via Definition 4 and it is $cov(T, traces^m(s)) = 1 - \varepsilon = 0.19$;

2) $l_c = 2$, $L_\varepsilon = \{a_0, b, c, d, e, f\}$
When we enlarge the set T to T' for $l_c = 2$ we find that $cov(T', traces^m(s)) = 0.51$;

3) $l_c = 1$, $L_{\varepsilon'} = \{a_0, a_1, b, c, d, e, f\}$
When we enlarge the set T to T'' for $L_{\varepsilon''} = \{a_0, a_1, b, c, d, e, f\}$ we find that $\varepsilon''_L = 0.06$ and that $cov(T'', traces^m(s)) = 0.29$.

In this case, one can see that the coverage increases more by adopting a higher value for the cycling limit than by using a larger label subset. But this is not always true, e.g. increasing l_c from 101 to 102 might give lesser increase than taking a larger labelset. Moreover we defined specific values for p_k and d_L.

It can be seen that in this example the monotonicity property required in [4] viz. $T \subseteq T' \Rightarrow cov(T) \leq cov(T')$ is respected. From an intuitive point of view this property is reasonable: if one wants a better coverage, one needs to generate more tests. This property can also be proven to hold in general.

8. CONCLUSIONS

A heuristic is a general guideline for reducing test suites, which must be made more precise to be practically applicable. In this paper we have studied two heuristics for reducing the number of traces in a test suite. Especially for the cycling heuristic we had to introduce additional notation. The reason is that the cycling structure of a trace through a finite-state automaton must be made explicit. We introduced *marked traces* for this purpose, which enabled us to extend the work on cycle reduction by Vuong [1, 2].

In order to introduce a notion of *coverage* for the test suites reduced by means of the above mentioned heuristics, we defined a *trace distance* on marked traces. The results of our studies can be used to effectively calculate the coverage of a test suite reduced with our techniques.

The proposed test selection technique can be compared to the existing theories in this area. In particular, these are the hypothesis theory developed by [6] and the trace distance theory of [1, 2]. The hypothesis theory embodies the trace distance theory (see [6]), but the nice thing about trace distance theory is that it gives a measure for the degree to which a reduced set of traces approximates the original one. So we chose an approach which combines these two theories. In our view, first the heuristics (test hypotheses in the theory of [6]) are to be defined. After that, based on these heuristics a trace distance is built. This gives the possibility to make a test selection with a given ε approximation. The

change of the heuristics leads to the change of the trace distance used in test selection.

We have started work on implementing our techniques in the TorX tool environment ([3]). An assumption for implementing our work is that a label distance exists. Because the TorX tools support the input of finite automata defined in LOTOS [7], we could use a label distance on LOTOS labels. This is not trivial because LOTOS labels may be parameterized by arbitrary data types.

References

[1] J. Alilovic-Curgus and S.T. Vuong. A metric based theory of test selection and coverage. In A. Danthine, G. Leduc, and P. Wolper, editors, *Protocol Specification, Testing, and Verification*, pages 289–304. North-Holland XIII, 1993.

[2] J. Alilovic-Curgus and S.T. Vuong. Sensitivity analysis of the metric based test selection. In M. Kim, S. Kang, and K. Hong, editors, *Int. Workshop on Testing of Communicating Systems*, volume X, pages 200–219. Chapman & Hall, 1997.

[3] A. Belinfante, J. Feenstra, R.G. Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, *Intenational Workshop on Testing of Comunication Systems*, pages 179–196. Kluwer Academic, 1999.

[4] E. Brinksma, J. Tretmans, and L. Verhaard. A framework for test selection. In B. Jonsson, J. Parrow, and B. Pehrson, editors, *Protocol, Specification, Testing, and Verification*, volume XI, pages 233–248. North-Holland, 1991.

[5] O. Charles and R. Groz. Formalisation d'hypothèses pour l'evaluation de la couverture de test. In *Actes du Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'96)*, Editions Hermes, 1996.

[6] O. Charles and R. Groz. Basing test coverage on a formalization of test hypotheses. In M. Kim, S. Kang, and K. Hong, editors, *Int. Workshop on Testing of Communicating Systems*, volume X, pages 109–124. Chapman & Hall, 1997.

[7] ISO. *Information Processing Systems, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.* International Standard IS–8807, 1989.

[8] T. Jéron and P. Morel. Test generation derived from model-checking. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification CAV'99*, volume 1633 of *Lecture Notes in Computer Science*, pages 108–121. Springer-Verlag, 1999.

[9] G.J. Myers. The art of software testing. John Wiley & Sons Inc, 1979.

[10] M. Schmitt, B. Koch, J. Grabowski, and D. Hogrefe. Autolink – a tool for the automatic and semi-automatic test generation. In A. Wolisz, I. Schieferdecker, and A. Rennoch, editors, *Formale Beschreibungstechniken für verteilte Systeme*, volume 315. GMD-Studien, St. Augustin, GI/ITG-Fachgespräch, GMD, 1997.

[11] G.A. Stephen. String search. Technical Report TR-92-gas-01, School of Electronic Engineering Science, University College of North Wales, 1992.

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