

Safety-complete Test Suites

Wen-ling Huang · Sadik Özoguz · Jan Peleska

Received: date / Revised version: date

Abstract In this paper, a novel safety-related variant of complete test suites for finite state machines is introduced. Under certain hypotheses which are similar to the ones used in the well-known W-Method and its improved versions, the new method guarantees to uncover every safety violation, while erroneous behaviour without safety-relevance may remain undetected. While the method can be based on any of the known complete strategies for FSM testing, its most effective variant is based on the H-method, and this variant is presented in detail, denoted as the Safety-complete H-Method. It is guaranteed that application of the Safety-complete H-Method always results in less or equally many test cases than when applying the original H-Method. In well-defined situations that can be pre-determined from the reference model, the Safety-complete H-Method leads to a substantial reduction of test cases in comparison to the size of the analogous H test suites. We advocate this new test suite for situations, where exhaustive testing of the complete system is too expensive. In these cases, strong guarantees with respect to fault coverage should only be given for the errors representing safety violations, while it may be considered as acceptable if less critical errors remain undetected.

Keywords Model-based testing, Complete testing theories, Safety

1 Introduction

1.1 Motivation

Complete test suites guarantee to uncover all conformance violations of the system under test (SUT) checked against a given reference model, provided that certain hypotheses – typically captured in a fault model – are fulfilled. This ideal test strength has attracted many researchers over the last 50 years, so that a large

Send offprint requests to: Jan Peleska

Department of Mathematics and Computer Science
University of Bremen, Germany
{huang,sadik,jp}@cs.uni-bremen.de

variety of contributions exists. On the other hand, the often infeasible size of the test suites involved has frequently prevented their practical application. As a result, there is a considerable interest in testing strategies allowing to focus the effort on certain critical properties, while requiring lesser fault coverage for non-critical ones. These approaches can be regarded as a combination of conformance testing and property-based testing: we are no longer interested in full conformance, but only require complete fault coverage for safety-properties.

1.2 Main Contributions

In this article, a novel contribution to property-oriented testing for the domain of deterministic, completely specified, finite state machines is presented. Our approach is based on the H-Method (Dorofeeva et al (2005)) which – to our best knowledge – currently is the most effective complete test method for FSMs, because it usually requires fewer test cases than the W-Method (Vasilevskii (1973); Chow (1978)), Wp-Method (Luo et al (1994)), or HSI-Method (Luo et al (1995)), when applied to the same reference model.

We extend the H-Method in such a way, that complete coverage for output and transition faults (including addition of new states) is guaranteed, if these lead to erroneous outputs representing safety-violations. To this end, an abstraction concept for outputs is introduced, so that it can be formally captured whether an erroneous replacement of another output for the expected one presents a safety violation or just a non-critical deviation. In contrast to other publications in this field, we formally prove that our strategy is complete with respect to this safety-related fault coverage. We show by means of examples, that applying this *Safety-complete H-Method* can lead to significantly reduced test suites in comparison to the H-Method, though this is not guaranteed, but depends on the nature of the reference model and its safety-related abstraction.

1.3 Main Contributions and Relation to Previous Work

The material presented here extends the publication Huang and Peleska (2017a) in the following ways.

1. While Huang and Peleska (2017a) was based on the Wp-Method, the present article uses the more effective H-Method, which requires less test cases while still ensuring completeness.
2. A new algorithm for calculating test suites according to the Safety-complete H-Method is presented.
3. The evaluation is now based on the Safety-complete H-Method, and it has been extended by using random-generations of FSMs.

1.4 Overview

In Section 2, basic terms and concepts are introduced, so that this article remains sufficiently self-contained. In Section 3, the Safety-complete H-Method and its related definitions are introduced, and its completeness properties are proven. An

algorithm for calculating safety-complete test suites is specified. In Section 4, three case studies are presented that provide some insight into the situations where the new method leads to a significant test case reduction. These case studies are complemented by statistical evaluations based on state machines generated at random. In Section 5, references to related work are given. Section 6 presents the conclusion.

2 Notation and Technical Background

2.1 Deterministic Finite State Machines

A *deterministic finite state machine (DFSM)* is a tuple $M = (Q, \underline{q}, \Sigma_I, \Sigma_O, h)$ denoting the finite state space Q , initial state $\underline{q} \in Q$, finite input and output alphabets Σ_I and Σ_O , and the transition relation $h \subseteq Q \times \Sigma_I \times \Sigma_O \times Q$. For deterministic machines, pre-state q and input x uniquely determine the associated output y and the post-state q' , such that $h(q, x, y, q')$ holds. We assume that all DFSMs are *completely specified*. This means that for every q and every x , there exists y and q' such that $h(q, x, y, q')$.

The *after operator* $q\text{-after-}\bar{x}$ maps a pre-state q and a finite sequence \bar{x} of inputs to the uniquely determined post-state q' resulting from repetitive application of h . The *language* of a DFSM is the set of finite input/output traces $\bar{x}/\bar{y} \in (\Sigma_I \times \Sigma_O)^*$ resulting from applying all $\bar{x} \in \Sigma_I^*$ to the initial state \underline{q} and associating the output trace \bar{y} which is uniquely determined by \underline{q} , \bar{x} , and h . Two DFSMs are *I/O-equivalent* ($M \sim M'$) if they produce the same language. The language of a state q (denoted by $L(q)$) is the set of all \bar{x}/\bar{y} generated by applying all $\bar{x} \in \Sigma_I^*$ to q . Two states q, q' are *distinguishable* if and only if their languages differ, that is, $L(q) \neq L(q')$. This implies the existence of an input trace \bar{x} and output traces $\bar{y} \neq \bar{y}'$, such that $\bar{x}/\bar{y} \in L(q)$ and $\bar{x}/\bar{y}' \in L(q')$. In this case, \bar{x} is said to be a *distinguishing (input) trace* of q and q' . The set of all distinguishing traces for states q, q' is denoted by $\text{Dist}(q, q')$. The *prime machine* $\text{prime}(M)$ of a DFSM M is the minimal DFSM producing the same language as M .

Let $A \subseteq \Sigma_I^*$. We denote the set of all prefixes of input traces from A , including these traces themselves, by $\text{Pref}(A)$. Obviously, $A \subseteq \text{Pref}(A)$ holds. The *boundary* of A is the subset

$$\partial A = \{\pi \in A \mid \forall \pi' \in A : \pi \in \text{Pref}(\pi') \Rightarrow \pi' = \pi\} \subseteq A.$$

Intuitively, ∂A contains the longest input traces from A , where no continuations inside A exist. The boundary of A satisfies the properties $A \subseteq \text{Pref}(\partial A)$ and $\forall \pi \in \partial A : \text{Pref}(\pi) \cap \partial A = \{\pi\}$.

Given $M = (Q, \underline{q}, \Sigma_I, \Sigma_O, h)$, a *state cover* $V \subseteq \Sigma_I^*$ is a set of input traces fulfilling

$$\forall q \in Q : \exists \bar{x} \in V : \underline{q}\text{-after-}\bar{x} = q.$$

Since $\underline{q} = \underline{q}\text{-after-}\varepsilon$, where ε denotes the empty trace, it is practical to assume for the remainder of this article that $\varepsilon \in V$. Note that in principle, it is possible to specify state covers not containing ε , if there exist non-empty paths re-visiting the initial state. If M has reachable states only (in particular, if M is a prime machine), state covers with cardinality $|Q|$ can always be found.

For input traces $\bar{x} = x_1 \dots x_k$, the suffix starting at index $i \in \{1, \dots, k\}$ is denoted by $\bar{x}^i = x_i \dots x_k$. The trace segment starting at i and ending at j , where $i \leq j$ and $i, j \in \{1, \dots, k\}$, is denoted by $\bar{x}^{[i,j]} = x_i \dots x_j$.

2.2 Test Suites

A *test suite* is a subset $TS \subseteq \Sigma_I^*$, each $\bar{x} \in TS$ is a *test case*. This simplified notation is possible, since only deterministic machines are considered, so that the input trace \bar{x} uniquely determines the output trace to be expected according to the reference DFSM. An implementation *passes* a test case \bar{x} if the application of this input sequence produces an output sequence \bar{y} , such that \bar{x}/\bar{y} is in the language of the reference DFSM.

2.3 Fault Models and Complete Test Suites

In the context of this article, a *fault model* is a triple $\mathcal{F} = (M, \leq, \mathcal{D})$. In this triple, M is a DFSM called the *reference model*. \mathcal{D} is a set of DFSMs over the same input/output alphabet as M called the *fault domain*, and \leq is a relation between DFSMs called the *conformance relation*. The fault domain \mathcal{D} may contain both DFSMs conforming to M and others non-conforming to M .

A test suite TS is called *complete with respect to fault model \mathcal{F}* , if and only if for all $M' \in \mathcal{D}$, the following properties are fulfilled.

1. If $M' \leq M$, then M' passes all tests in TS (soundness).
2. If $M' \not\leq M$, then M' fails at least one test in TS (exhaustiveness).

By $\mathcal{D}(\Sigma_I, \Sigma_O, m)$, we denote the fault domain of all deterministic, completely specified DFSMs over alphabet (Σ_I, Σ_O) , whose prime machines have at most m states.

2.4 The H-Method

The *H-Method* is a method for generating complete test suites, originally presented in Dorofeeva et al (2005). It can be applied to completely or incompletely specified, deterministic or nondeterministic finite state machines. To our best knowledge, the H-Method is the most effective complete testing method currently known, in the sense that it can guarantee completeness with respect to a given fault model with the lowest number of test cases required. Therefore, our safety-oriented test method presented in the next section will be based on the H-Method. In the context of our article, we only need the completely specified, deterministic variant. This is described in the sequel, to make this article sufficiently self-contained.

The following theorem has been proven in (Dorofeeva et al, 2005, Theorem 1). It is re-phrased here in the slightly specialised form for complete DFSMs.

Theorem 1 (H-Method) *Let M be a completely specified, deterministic prime machine with n states. Let $m \geq n \geq 2$. Let V be a state cover of M , and let TS be a finite*

test suite of finite test cases. Define auxiliary sets

$$\begin{aligned}
A &= V \times V \\
B &= V \times \left(V \cdot \bigcup_{i=1}^{m-n+1} \Sigma_I^i \right) \\
C &= \{(\nu.\gamma', \nu.\gamma) \mid \nu \in V \wedge \gamma \in \left(\bigcup_{i=1}^{m-n+1} \Sigma_I^i \right) \wedge \gamma' \in \text{Pref}(\gamma) - \{\varepsilon\}\} \\
D &= \{(\alpha, \beta) \in A \cup B \cup C \mid \underline{q\text{-after-}}\alpha \neq \underline{q\text{-after-}}\beta\}
\end{aligned}$$

Let

$$\Delta : D \rightarrow \Sigma_I^*$$

be a map satisfying

$$\forall (\alpha, \beta) \in D : \Delta(\alpha, \beta) \in \text{Dist}(\underline{q\text{-after-}}\alpha, \underline{q\text{-after-}}\beta)$$

Define a test case set H by

$$H = \{\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta) \mid (\alpha, \beta) \in D\}$$

Then any test suite TS containing ∂H is complete with respect to fault model $\mathcal{F} = (M, \sim, \mathcal{D}(\Sigma_I, \Sigma_O, m))$, that is,

$$\forall M' \in \mathcal{D}(\Sigma_I, \Sigma_O, m) : M' \stackrel{\partial H}{\sim} M \Leftrightarrow M' \sim M$$

□

For the degenerate case where the reference model M has just one state ($n = 1$), the set D specified in Theorem 1 is empty, because there are no two states in M that can be distinguished. Therefore, also H is empty, and another test suite definition is needed for this situation.

Theorem 2 *If M has only one state ($n = 1$), any test suite containing Σ_I^m is complete with respect to fault model $\mathcal{F} = (M, \sim, \mathcal{D}(\Sigma_I, \Sigma_O, m))$.* □

We do not present the original proofs of Theorem 1 and 2 established in Dorofeeva et al (2005), because it will turn out (see Corollary 1 below) that for the case where only completely specified DFSMs are considered, the H-method can be regarded as a special case of the more general *safety-complete H-method* introduced and proven in Section 3 below.

3 A Safety-complete H-Method

3.1 Safety-related Output Abstractions

Let $M = (Q, q, \Sigma_I, \Sigma_O, h)$ be a deterministic completely specified FSM. Then any reflexive and transitive relation $\leq_s \subseteq \Sigma_O \times \Sigma_O$ is called a *safety-related output abstraction*. The intuition behind this definition is that $y \leq_s y'$ indicates that an erroneous output of y' instead of an expected output y does not induce a safety

violation. Reflexivity just indicates that the occurrence of the output expected according to the reference model M can never be a safety violation. Transitivity implies that output z must also be a safe replacement of w , if $w \leq_s y \wedge y \leq_s z$ holds. Relation \leq_s induces an equivalence relation \sim_s on $\Sigma_O \times \Sigma_O$ by defining

$$y_1 \sim_s y_2 \equiv y_1 \leq_s y_2 \wedge y_2 \leq_s y_1$$

Example 1 Consider a train onboard controller which compares actual train speed against the allowed speed and progressively outputs

$$\Sigma_O = \{\text{ok}, \text{warning}, \text{ServiceBrakeTrigger}, \text{EmergencyBrakeTrigger}\},$$

depending on how much the train is overspeeding. The outputs **ok** and **warning** are shown on the display unit of the train engine driver, whereas the outputs **ServiceBrakeTrigger** and **EmergencyBrakeTrigger** directly act on the train's braking system. The service brake slows the train down with lower braking force than the emergency brake, so that the latter is used only as the "last resort", when warnings and service brake interventions do not suffice. These considerations induce a safety-related output abstraction \leq_s as the reflexive and transitive closure of

$$\text{ok} \leq_s \text{warning} \leq_s \text{ServiceBrakeTrigger} \leq_s \text{EmergencyBrakeTrigger}$$

The intuition behind this definition is that a warning or even a braking intervention performed by the controller is an acceptable substitute for an expected **ok**-output from the safety perspective: the substitute output may be a nuisance (a spurious warning when the speed is within range) or even a severe reduction of reliability (triggering the emergency brake without need), but it does not introduce a safety threat. The same holds for situations where the service brake should be triggered but instead, the emergency brakes are activated.

When an intervention by service brakes or emergency brakes is expected, however, an output **ok** or **warning** would certainly be regarded as a safety hazard.

Next, suppose that the outputs to the train engine driver are extended by status messages

$$\Sigma'_O = \{\mathbf{s}_1, \dots, \mathbf{s}_n\}.$$

Since these informative messages have no safety-relevance at all, we wish to extend the relation \leq_s in a way expressing that each status message can be replaced by any other output of $\Sigma_O \cup \Sigma'_O$ without causing any safety hazard. This is achieved by extending \leq_s according to the rules

$$\begin{aligned} \mathbf{s} &\sim_s \mathbf{s}' \text{ for all } \mathbf{s}, \mathbf{s}' \in \Sigma'_O \\ \mathbf{s} &\leq_s \mathbf{e} \text{ for all } \mathbf{s} \in \Sigma'_O, \mathbf{e} \in \Sigma_O \end{aligned}$$

Finally consider a design extension, where the onboard controller operates in a de-centralised distributed train control environment, so that it switches its own points

$$\Sigma''_O = \{\mathbf{p}_i^+, \mathbf{p}_i^- \mid i = 1, \dots, m\}$$

along the route (such a system has been investigated, for example, in Haxthausen and Peleska (2000)). Notation \mathbf{p}_i^+ stands for switching point number i into the straight position, \mathbf{p}_i^- for switching the point into the branching position. From the safety-perspective, switching a point into the desired position cannot be replaced

by any other event without introducing a safety hazard. Therefore we extend \leq_s this time as follows.

$$\begin{aligned} p &\leq_s p \text{ for all } p \in \Sigma_O'' \\ \mathbf{s} &\leq_s \mathbf{p} \text{ for all } \mathbf{s} \in \Sigma_O', \mathbf{p} \in \Sigma_O'' \end{aligned}$$

The events of Σ_O and Σ_O'' , however, are incomparable: switching a point can never be a safe replacement for braking a train, and vice versa. \square

Given a safety-related output abstraction \leq_s on Σ_O , this is extended in the natural way to a reflexive and transitive relation (again denoted by \leq_s) on output traces $\iota, \pi \in \Sigma_O^*$ by setting

$$\iota \leq_s \pi \equiv (\#\iota = \#\pi \wedge \forall i \in \{1, \dots, \#\iota\} : \iota(i) \leq_s \pi(i))$$

for $\iota, \pi \in \Sigma_O^*$, where $\#\iota, \#\pi$ is the length of ι, π , respectively.

Now let q, q' be two states of the same state machine or of different state machines over the same input/output alphabet (Σ_I, Σ_O) . In the latter case, it is assumed without loss of generality that their states come from disjoint sets Q, Q' . Then it is possible to specify a joint output function $\omega : (Q \cup Q') \times \Sigma_I \rightarrow \Sigma_O$ which is extended in the natural way to operate on sequences of inputs, i.e. $\omega : (Q \cup Q') \times \Sigma_I^* \rightarrow \Sigma_O^*$. Let $\bar{x} \in \Sigma_I^*$ be an input trace. We define

$$q' \stackrel{\bar{x}}{\leq}_s q \equiv (\omega(q', \bar{x}) \leq_s \omega(q, \bar{x})).$$

Intuitively speaking, $q' \stackrel{\bar{x}}{\leq}_s q$ states that applying input trace \bar{x} to state q produces an output sequence $\omega(q, \bar{x})$ which is an admissible substitute to the output sequence $\omega(q', \bar{x})$ expected when applying the same input sequence to q' .

Relation $\stackrel{\bar{x}}{\leq}_s$ induces an equivalence relation on states by defining

$$q' \stackrel{\bar{x}}{\sim}_s q \equiv (q' \stackrel{\bar{x}}{\leq}_s q \wedge q \stackrel{\bar{x}}{\leq}_s q')$$

These relations can be extended to sets of input traces in the natural way by defining

$$\begin{aligned} q' \stackrel{W}{\leq}_s q &\equiv (\forall \bar{x} \in W : q' \stackrel{\bar{x}}{\leq}_s q) \\ q' \stackrel{W}{\sim}_s q &\equiv (\forall \bar{x} \in W : q' \stackrel{\bar{x}}{\sim}_s q) \end{aligned}$$

for arbitrary $W \subseteq \Sigma_I^*$. Finally, the specific case where $W = \Sigma_I^*$ is written in the simplified notation

$$\begin{aligned} q' \leq_s q &\equiv (q' \stackrel{\Sigma_I^*}{\leq}_s q) \\ q' \sim_s q &\equiv (q' \stackrel{\Sigma_I^*}{\sim}_s q). \end{aligned}$$

If $q' \sim_s q$ holds, any input trace applied to q' will lead to an output trace which – regarded from the safety perspective – is an admissible replacement of the outputs expected when applying the same inputs to q and vice versa. If the initial states \underline{q} and \underline{q}' of two state machines M, M' are s-equivalent ($\underline{q}' \sim_s \underline{q}$), we denote this by $M' \sim_s M$.

The following lemma states a simple, but important fact which links \sim_s to \sim . To state the lemma, we introduce the *identity relation*

$$\mathbf{id}(X) = \{(x, x) \mid x \in X\}$$

which relates every element of some set to itself, but leaves distinct elements incomparable.

Lemma 1 *If $\leq_s = \mathbf{id}(\Sigma_O)$, then $\sim_s = \sim$.*

Proof Since \leq_s is the diagonal relation by assumption, two output traces $\iota, \pi \in \Sigma_O^*$ are only comparable if they coincide, that is,

$$\iota \leq_s \pi \Leftrightarrow \pi \leq_s \iota \Leftrightarrow \iota = \pi.$$

As a consequence, $q' \leq_s^{\bar{x}} q$ if and only if $q' \stackrel{\bar{x}}{\sim} q$, and therefore, $\sim_s = \sim$ follows. \square

3.2 The Safety-complete H-Method

Throughout this section, let $M = (Q, \underline{q}, \Sigma_I, \Sigma_O, h)$, $M' = (Q', \underline{q}', \Sigma_I, \Sigma_O, h')$ be completely specified, deterministic, and minimised FSMs over the same input/output alphabet $\Sigma = \Sigma_I \times \Sigma_O$ with $|Q| = n$, $|Q'| \leq m$ and $m \geq n$. Let $\leq_s \subseteq \Sigma_O \times \Sigma_O$ be a safety-related output abstraction with associated equivalence relation \sim_s .

Definition 1 (Safety-complete Test Suite) With the terms introduced above and in Section 2.3, define fault model $\mathcal{F}_s = (M, \sim_s, \mathcal{D}(\Sigma_I, \Sigma_O, m))$. Let $\text{TS} \subseteq \Sigma_I^*$ be a test suite.

1. TS is called *sound* w.r.t. fault model \mathcal{F}_s , if and only if every member $M' \in \mathcal{D}(\Sigma_I, \Sigma_O, m)$ which is I/O-equivalent to M ($M' \sim M$) passes the test suite.
2. TS is called *safety-exhaustive* w.r.t. fault model \mathcal{F}_s , if and only if every member $M' \in \mathcal{D}(\Sigma_I, \Sigma_O, m)$ which is not safety-equivalent to M ($M' \not\sim_s M$) fails at least one test case in TS.
3. TS is called *safety-complete* w.r.t. fault model \mathcal{F} , if it is both sound and safety-exhaustive.

\square

Note that in the definition above, we do not require every safety-equivalent implementation to pass the safety-complete suite, because we are happy if the safety-driven suite also uncovers some non-critical failures violating I/O-equivalence. Furthermore, observe that the definition of safety-completeness coincides with ordinary completeness, if \leq_s is the identity relation on Σ_O : this follows directly from Lemma 1.

Theorem 3 (Safety-complete H-method) *Using the notation above, let $V \subseteq \Sigma_I^*$ be a minimal state cover of M containing ε . Define auxiliary sets*

$$A = V \times V$$

$$B = V \times \left(V \cdot \bigcup_{i=1}^{m-n+1} \Sigma_I^i \right)$$

$$C = \{(\nu.\gamma', \nu.\gamma) \mid \nu \in V \wedge \gamma \in \left(\bigcup_{i=1}^{m-n+1} \Sigma_I^i \right) \wedge \gamma' \in \text{Pref}(\gamma) - \{\varepsilon\}\}$$

$$D_s = \{(\alpha, \beta) \in A \mid \underline{q}\text{-after-}\alpha \neq \underline{q}\text{-after-}\beta\} \cup \{(\alpha, \beta) \in B \cup C \mid \underline{q}\text{-after-}\alpha \not\sim_s \underline{q}\text{-after-}\beta\}$$

Let

$$\Delta : D_s \rightarrow \Sigma_I^*$$

be a map satisfying

$$\forall (\alpha, \beta) \in D_s : \Delta(\alpha, \beta) \in \text{Dist}(\underline{q}\text{-after-}\alpha, \underline{q}\text{-after-}\beta).$$

Define a test case set H_s by

$$H_s = V \cdot \Sigma_I^{m-n+1} \cup \{\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta) \mid (\alpha, \beta) \in D_s\}.$$

Then any test suite TS containing ∂H_s is safety-complete w.r.t. \mathcal{F}_s , that is,

$$M' \overset{\partial H_s}{\sim} M \Rightarrow M' \sim_s M.$$

Proof Assume that M' passes the test suite, but is not safety-equivalent to M , that is, $\underline{q}' \overset{\partial H_s}{\sim} \underline{q}$ but $\underline{q}' \not\sim_s \underline{q}$. Since $\varepsilon \in V$ by assumption and $V \subseteq \Sigma_I^*$, we can

conclude that $\Sigma_I^* = V \cdot \Sigma_I^*$. Therefore, any sequence $\bar{z} \in \Sigma_I^*$ satisfying $\underline{q}' \not\sim_s \underline{q}$ can be re-written as $\bar{z} = \bar{v}.\bar{x}$ with $\bar{v} \in V$ and $\bar{x} \in \Sigma_I^*$ (note that \bar{v} may be the empty trace ε). As a consequence, we can select a *shortest* sequence \bar{x} satisfying $\underline{q}' \not\sim_s \underline{q}$.

Since $\underline{q}' \overset{\partial H_s}{\sim} \underline{q}$, we also have $\underline{q}' \overset{\partial H_s}{\sim} \underline{q}$. Therefore, the traces of $V.\bar{x}$ are not completely contained in ∂H_s . Now, by definition of H_s , the set of prefixes $\text{Pref}(\partial H_s)$ contains all input traces of $V \cdot \left(\bigcup_{i=1}^{m-n+1} \Sigma_I^i \right)$. As a consequence, the length of \bar{x} must be *greater* than $(m - n + 1)$.

Let $\nu \in V$ such that $\underline{q}' \not\sim_s \underline{q}$. Then $V \cup \{\nu.\bar{x}^{[1,i]} \mid i = 1, \dots, m - n + 1\}$ contains $n + (m - n + 1) = m + 1$ input sequences, because the two operands of the set union are disjoint for the following reasons: suppose that $\nu.\bar{x}^{[1,i]} \in V$ for some $i \geq 1$. Then the suffix \bar{x}^{i+1} would be a shorter input sequence with $\underline{q}' \overset{V.\bar{x}^{i+1}}{\sim} \underline{q}$, a contradiction to the assumption about \bar{x} being a shortest sequence of this kind.

Since M' has at most m states, there are two input sequences $\alpha \neq \beta$ in $V \cup \{\nu.\bar{x}^{[1,i]} \mid i = 1, \dots, m - n + 1\}$ with $\underline{q}'\text{-after-}\alpha = \underline{q}'\text{-after-}\beta$. There are three cases to be distinguished about α, β :

1. $\alpha, \beta \in V$.
2. $\alpha \in V, \beta = \nu.\bar{x}^{[1,j]}$ for some $j = 1, \dots, m - n + 1$.
3. $\alpha = \nu.\bar{x}^{[1,i]}, \beta = \nu.\bar{x}^{[1,j]}$, for some $i < j$ with $i, j \in \{1, \dots, m - n + 1\}$

Before analysing the 3 cases, we first observe that for the α, β occurring in cases 1, 2, or 3, the following fact holds.

$$\underline{q'}\text{-after-}\alpha = \underline{q'}\text{-after-}\beta \Rightarrow \underline{q}\text{-after-}\alpha \sim_s \underline{q}\text{-after-}\beta. \quad (1)$$

This is shown by contradiction; suppose therefore, that $\underline{q'}\text{-after-}\alpha = \underline{q'}\text{-after-}\beta$ and $\underline{q}\text{-after-}\alpha \not\sim_s \underline{q}\text{-after-}\beta$. Then H_s contains two tests $\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta)$ distinguishing $\underline{q}\text{-after-}\alpha$ from $\underline{q}\text{-after-}\beta$ w.r.t. \sim_s , and therefore also w.r.t. \sim . Since $\underline{q'}\text{-after-}\alpha = \underline{q'}\text{-after-}\beta$ by assumption, M' would produce the same outputs when applying $\Delta(\alpha, \beta)$ to these identical states, whereas M produces different outputs when applying $\Delta(\alpha, \beta)$ to $\underline{q}\text{-after-}\alpha$ and $\underline{q}\text{-after-}\beta$. This is a contradiction, because M' is supposed to pass all tests in ∂H_s .

With this information at hand, it can be concluded that Case 1 above is not possible, because the input traces in V lead to n distinguishable states in M , so they must also lead to n distinguishable states in M' according to implication (1). As a consequence, only Case 2 or Case 3 can apply. Hence $\beta = \nu.\bar{x}^{[1,j]}$ for some $j = 1, \dots, m - n + 1$, and $\nu.\bar{x} = \beta.\bar{x}^{j+1}$. Since $|\bar{x}| > m - n + 1$ the suffix \bar{x}^{j+1} is non-empty. This leads to the derivation

$$\begin{aligned} \underline{q'} \not\sim_s^{\nu.\bar{x}} \underline{q} &\Leftrightarrow \underline{q'} \not\sim_s^{\beta.\bar{x}^{j+1}} \underline{q} && [\nu.\bar{x} = \beta.\bar{x}^{j+1}] \\ &\Leftrightarrow \underline{q'}\text{-after-}\beta \not\sim_s^{\bar{x}^{j+1}} \underline{q}\text{-after-}\beta \quad [\beta \in \text{Pref}(H_s), \text{ therefore } \underline{q'} \stackrel{\beta}{\sim} \underline{q}] \\ &\Leftrightarrow \underline{q'}\text{-after-}\alpha \not\sim_s^{\bar{x}^{j+1}} \underline{q}\text{-after-}\beta \quad [\underline{q'}\text{-after-}\alpha = \underline{q'}\text{-after-}\beta] \\ &\Leftrightarrow \underline{q'}\text{-after-}\alpha \not\sim_s^{\bar{x}^{j+1}} \underline{q}\text{-after-}\alpha \quad [(1) \text{ implies } \underline{q}\text{-after-}\alpha \sim_s \underline{q}\text{-after-}\beta] \\ &\Leftrightarrow \underline{q'} \not\sim_s^{\alpha.\bar{x}^{j+1}} \underline{q} && [\alpha \in \text{Pref}(H_s), \text{ therefore } \underline{q'} \stackrel{\alpha}{\sim} \underline{q}] \end{aligned}$$

Since $\alpha.\bar{x}^{j+1} \in V.\bar{x}^{j+1}$ or $\alpha.\bar{x}^{j+1} \in V.\bar{x}^{[1,i]}. \bar{x}^{j+1}$, both \bar{x}^{j+1} and $\bar{x}^{[1,i]}. \bar{x}^{j+1}$ are shorter than \bar{x} , a contradiction to the assumption that \bar{x} is a shortest input sequence with $\underline{q'} \not\sim_s^{\nu.\bar{x}} \underline{q}$. Therefore, such an \bar{x} cannot exist, and $\underline{q'}$ must be \sim_s -equivalent to \underline{q} . This concludes the proof. \square

Observe that in the proof of the theorem above, no reference to the completeness of the original H-method stated in Theorem 1 was needed. From the perspective of complete testing theories, it is interesting to note that Theorem 3 can be regarded as a generalisation of the H-Method, whose completeness has been stated above in Theorem 1. This fact is expressed by the following corollary.

Corollary 1 *For completely specified DFSMs, the safety-complete H-method implies the completeness of the H-method.*

Proof Let $\leq_s = \mathbf{id}\Sigma_O$. Then Lemma 1 implies that $\sim_s = \sim$. For this case, the set D_s specified in Theorem 3 coincides with D as specified in Theorem 1. As a consequence, also H_s in Theorem 3 equals H as specified in Theorem 1. This establishes that Theorem 1 is the special case of Theorem 3, where $\leq_s = \mathbf{id}\Sigma_O$. \square

It is noteworthy that

$$V.\Sigma_I^{m-n+1} \subseteq \{\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta) \mid (\alpha, \beta) \in D_s\}$$

holds whenever M has at least two \sim_s -distinguishable states. This follows directly from the definition of the auxiliary sets B and D_s in Theorem 3. Therefore,

$$H_s = \{\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta) \mid (\alpha, \beta) \in D_s\}$$

if two \sim_s -distinguishable states can be found in M . Only for the degenerate case where all states of M are equivalent w.r.t. \sim_s , the set $\{\alpha.\Delta(\alpha, \beta), \beta.\Delta(\alpha, \beta) \mid (\alpha, \beta) \in D_s\}$ is empty, so the test suite consists of the traces in $V.\Sigma_I^{m-n+1}$.

Observe further, that Theorem 3 leaves considerable degrees of freedom regarding the implementation of the Safety-complete H-Method. This is due to the fact that the mapping $\Delta : D_s \rightarrow \Sigma_I^*$ is not uniquely determined. Different choices of Δ result in different sizes of the test suite ∂H_s . Finding the optimal choice resulting in a minimal test suite size has very high complexity. Even if we restrict the possible choices of image values $\Delta(\alpha, \beta) \in \Sigma_I^*$ just to the shortest distinguishing traces, there would be up to $|\Sigma_I|^{|D_s|}$ different choices for selecting the best possible solution Δ . Therefore we suggest a greedy algorithm below for implementing the method: for each $(\alpha, \beta) \in D_s$, a distinguishing trace ω is selected, such that $(\alpha.\omega, \beta.\omega)$ extends the *current state* of H_s in the least possible way.

3.3 Implementation

For implementing an algorithm calculating the safety-complete test suite according to Theorem 3, we proceed as follows.

FSM Abstraction. Given a completely specified, deterministic, minimal FSM $M = (Q, \underline{q}, \Sigma_I, \Sigma_O, h)$, every safety-related output abstraction $\leq_s \subseteq \Sigma_O \times \Sigma_O$ induces an abstraction α_s of the alphabet by mapping each output $y \in \Sigma_O$ to the set of outputs $y' \in \Sigma_O$ that are greater or equal to y according to \leq_s .

$$\alpha_s : \Sigma_O \rightarrow \mathbb{P}(\Sigma_O); \quad y \mapsto \{y' \in \Sigma_O \mid y \leq_s y'\}$$

The image $\Sigma_O^s = \alpha_s(\Sigma_O)$ is again finite, therefore it can be used as a new output alphabet of a state machine M_s which is an abstraction of M with respect to \leq_s in the following sense.

$$\begin{aligned} M_s &= \mathbf{prime}(Q, \underline{q}, \Sigma_I, \Sigma_O^s, h_s) \\ h_s &= \{(q, x, \alpha_s(y), q') \mid (q, x, y, q') \in h\} \end{aligned}$$

Though M is assumed to be already minimised, the abstracted machine $(Q, \underline{q}, \Sigma_I, \Sigma_O^s, h_s)$ will not be minimal in general, because the output abstraction

may result in fewer states of Q being distinguishable. Therefore M_s is specified as the prime machine of $(Q, q, \Sigma_I, \Sigma_O^s, h_s)$.

By construction, two different states (q, q') in M_s produce outputs for certain input traces that differ in Σ_O^s . As a consequence, $q \not\sim_s q'$ holds.

Algorithm. With the DFSM abstraction at hand, the algorithm for calculating a safety-complete test suite is specified as follows.

1. **Input 1.** Reference model $M = (Q, q, \Sigma_I, \Sigma_O, h)$ with $|Q| = n$.
2. **Input 2.** Integer m satisfying $m \geq n$.
3. **Input 3.** Deterministic, completely specified, minimised FSM $M_s = (Q_s, q_s, \Sigma_I, \Sigma_O^s, h_s)$ resulting from the abstraction of M with respect to \leq_s .
4. **Output.** Test suite TS which is safety-complete with respect to fault model $\mathcal{F} = (M, \sim_s, \mathcal{D}(\Sigma_I, \Sigma_O, m))$.
5. Calculate state cover V from M .
6. Calculate $E = \bigcup_{i=1}^{m-n+1} \Sigma_I^i$.
7. Define sets A, B, C of pairs as specified in Theorem 3.
8. Calculate a distinguishability function

$$\text{Dist} : Q \times Q \rightarrow \mathbb{P}(\Sigma_I^*)$$

mapping pairs of non-equal states to the set of shortest input traces distinguishing the two in M .

9. Calculate a distinguishability function

$$\text{Dist}_s : Q \times Q \rightarrow \mathbb{B}$$

returning **true** for pairs of states (q_1, q_2) that are distinguishable in M_s , which means that $q_1 \not\sim_s q_2$.

10. Initialise the test case set H_s by setting $H_s = V.E$.
11. For all $(\alpha, \beta) \in A$ do
 - (a) Set $q_1 = q\text{-after-}\alpha$ and $q_2 = q\text{-after-}\beta$, with both states calculated in M .
 - (b) If $q_1 = q_2$ in M , continue with the next pair (α, β) at Step 11.
 - (c) Select $\omega \in \text{Dist}(q_1, q_2)$ such that the costs for adding $\alpha.\omega$ and $\beta.\omega$ are minimal. The costs are defined as follows. (a) If $\alpha.\omega \in \text{Pref}(H_s)$ and $\beta.\omega \in \text{Pref}(H_s)$, the costs are zero. (b) If $\alpha.\omega \in \text{Pref}(H_s)$ and $\beta.\omega$ is an extension of an existing input trace in ∂H_s , the costs are 1; the same holds for the case $\beta.\omega \in \text{Pref}(H_s)$ and $\alpha.\omega$ an extension of a trace in ∂H_s . (c) If $\alpha.\omega \in \text{Pref}(H_s)$ and $\beta.\omega$ neither is contained in $\text{Pref}(H_s)$, nor extends a trace from ∂H_s , or vice versa, the costs are 3. (d) If $\alpha.\omega$ extends ∂H_s and $\beta.\omega$ neither is contained in $\text{Pref}(H_s)$, nor extends a trace from ∂H_s , or vice versa, the costs are 4. (e) If both $\alpha.\omega$ and $\beta.\omega$ lead to new test cases, the costs are 5.
 - (d) Set $H_s = H_s \cup \{\alpha.\omega, \beta.\omega\}$.
12. For all $(\alpha, \beta) \in B \cup C$ do
 - (a) Set $q_1 = q\text{-after-}\alpha$ and $q_2 = q\text{-after-}\beta$, with both states calculated in M .
 - (b) If $\neg \text{Dist}_s(q_1, q_2)$, continue with the next pair (α, β) at Step 12.
 - (c) Otherwise select $\omega \in \text{Dist}(q_1, q_2)$ such that the costs for adding $\alpha.\omega$ and $\beta.\omega$ are minimal as specified above.
 - (d) Set $H_s = H_s \cup \{\alpha.\omega, \beta.\omega\}$.

13. Return $TS = \partial H_s$.

Termination of this algorithm is obvious, since the sets A, B, C are finite. The cost function is motivated by the goal to add as few test cases as possible to H_s . An extension of an existing trace from ∂H_s does not add a new test case, but just extends an existing one, so that only the number of test steps is increased. A trace, however, that is neither contained in H_s nor extends a trace in ∂H_s , corresponds to a new test case; so the highest costs occur when $\alpha.\omega, \beta.\omega$ give rise to two new test cases.

In embedded systems testing, the time needed to restart an SUT for a new test cases is usually much longer than the time needed to execute one test step; therefore adding a test step causes lower costs than adding a test case.

Note that this algorithm also allows to calculate test cases for the original H-Method, by setting $M_s = M$, which holds for $\leq_s = \text{id}(\Sigma_O)$.

FSM Open Source Library. We have published the open source C++ library `fsmlib-cpp`¹ which contains all algorithms needed for implementing the Safety-complete H-Method.² This library also provides essential methods for minimising DFSMs and for making nondeterministic FSMs observable. Moreover, a generator main program is provided which uses these methods to calculate test suites following the W-Method, Wp-Method, H-Method, and their safety-complete equivalents. An overview over this library is given in the lecture notes (Peleska and Huang, 2017, Appendix B).

4 Case Studies and Strategy Evaluation

4.1 Control of Fasten Seat Belt and Return-to-Seat Signs in the Aircraft Cabin

Application. The following example is a (slightly simplified) real-world example concerning safety-related and uncritical indications in an aircraft cabin. A *cabin controller* in a modern aircraft switches the *fasten seat belt (FSB) signs* located above the passenger seats in the cabin and the *return to seat (RTS) signs* located in the lavatories according to the rules modelled in the DFSM shown in Table 1.

As inputs, the cabin controller reads the actual position of the fasten seat belts switch in the cockpit, which has the position **f0** (OFF), **f1** (ON), and **f2** (AUTO). Further inputs come from the cabin pressure control system which indicates “cabin pressure low” by event **d1** and “cabin pressure ok” by **d0**. This controller also indicates “excessive altitude” by **e1** or “altitude in admissible range” by **e0**. Another sub-component of the cabin controller determines whether the so-called AUTO condition is true (event **a1**) or false (**a0**).

The cabin controller switches the fasten seat belt signs and return to seat signs on and off, depending on the actual input change and its current internal state. As long as the cabin pressure and the cruising altitude are ok (after initialisation of the cabin controller or if last events from the cabin pressure controller were **d0**, **e0**), the status of the FSB and RTS signs is determined by the cockpit switch and the AUTO condition: if the switch is in the ON position, both FSB and RTS signs

¹ <https://github.com/agbs-uni-bremen/fsmlib-cpp.git>

² The Safety-complete H-Method is implemented in file `fsm-generator.cpp`.

are switched on (output 11 in Table 1). Turning the switch into the OFF position switches the signs off. If the switch is in the AUTO position, both FSB and RTS signs are switched on if the AUTO condition becomes true with event **a1**, and they are switched off again after event **a0**. The AUTO condition may depend on the status of landing gears, slats, flaps, and oil pressure, these details are abstracted to **a1**, **a0** in our example.

As soon as there occurs a loss of pressure in the cabin (event **d1**) or an excessive altitude is reached, the FSB signs must be switched on and remain in this state, regardless of the actual state of the cockpit switch and the AUTO condition. The RTS signs, however, need to be switched off, because passengers should not be encouraged to leave the lavatories in a low pressure or excessive altitude situation.

After the cabin pressure and the altitude are back in the admissible range, the FSB and RTS signs shall automatically resume their state as determined by the “normal” inputs from cockpit switch and AUTO condition.

Safety considerations. Analysing the outputs

$$(\text{FSB}, \text{RTS}) \in \Sigma_O = \{00, 10, 11, 01\}$$

from the safety-perspective, leads to the identification of one safety-critical output $(\text{FSB}, \text{RTS}) = (1, 0)$, which should be set whenever cabin decompression or excessive altitude occurs. If the other outputs $\{00, 11, 01\}$ are changed due to an application error, this is certainly undesirable, but does not represent a safety hazard. Note that the output combination 01 should never occur at all.

These considerations lead to an abstraction function

$$\begin{aligned} \alpha_s : \Sigma_O &\rightarrow \mathbb{P}(\Sigma_O) \\ 00 &\mapsto \{00, 10, 11, 01\} \\ 11 &\mapsto \{00, 10, 11, 01\} \\ 01 &\mapsto \{00, 10, 11, 01\} \\ 10 &\mapsto \{10\} \end{aligned}$$

as introduced in Section 3.3, and the abstracted FSM described there is obtained by replacing outputs 00, 01, 11 by $YY = \{00, 10, 11, 01\}$, while leaving every occurrence of output 10 unchanged.

Comparison H-Method versus Safety-complete H-Method. The reference DFSM with 24 states as specified in Table 1 is already minimal. Applying our implementation of the H-Method results in a test suite with 511 test cases, for the case $m = n = 24$. The Safety-complete H-Method results in 337 test cases; this corresponds to an improvement of 34%. The Wp-Method requires 841 test cases.

The main reason for this significant test case reduction is that the prime machine of the new DFSM resulting from the safety abstraction only has 4 states which corresponds to a state reduction of 83%, when compared to the original DFSM from Table 1. This observation will be confirmed in the more general evaluation below.

Table 1 State-transition table of DFSM specifying the control of FSB signs and RTS signs in an aircraft cabin.

	f0	f1	f2	d1	d0	e1	e0	a1	a0
s0	s0/00	s1/11	s2/00	s3/10	s0/00	s6/10	s0/00	s12/00	s0/00
s1	s0/00	s1/11	s2/00	s4/10	s1/11	s7/10	s1/11	s13/11	s1/11
s2	s0/00	s1/11	s2/00	s5/10	s2/00	s8/10	s2/00	s14/11	s2/00
s3	s3/10	s4/10	s5/10	s3/10	s0/00	s9/10	s3/10	s15/10	s3/10
s4	s3/10	s4/10	s5/10	s4/10	s1/11	s11/10	s4/10	s16/10	s4/10
s5	s3/10	s4/10	s5/10	s5/10	s2/00	s11/10	s5/10	s17/10	s5/10
s6	s6/10	s7/10	s8/10	s9/10	s6/10	s6/10	s0/00	s18/10	s6/10
s7	s6/10	s7/10	s8/10	s10/10	s7/10	s7/10	s1/11	s19/10	s7/10
s8	s6/10	s7/10	s8/10	s11/10	s8/10	s8/10	s2/00	s20/10	s8/10
s9	s9/10	s10/10	s11/10	s9/10	s6/10	s9/10	s3/10	s21/10	s9/10
s10	s9/10	s10/10	s11/10	s10/10	s7/10	s10/10	s4/10	s22/10	s10/10
s11	s9/10	s10/10	s11/10	s11/10	s8/10	s11/10	s5/10	s23/10	s11/10
s12	s12/00	s13/11	s14/11	s15/10	s12/00	s18/10	s12/00	s12/00	s0/00
s13	s12/00	s13/11	s14/11	s16/10	s13/11	s19/10	s13/11	s13/11	s1/11
s14	s12/00	s13/11	s14/11	s17/10	s14/11	s20/10	s14/11	s14/11	s2/00
s15	s15/10	s16/10	s17/10	s15/10	s12/00	s21/10	s15/10	s15/10	s3/10
s16	s15/10	s16/10	s17/10	s16/10	s13/11	s22/10	s16/10	s16/10	s4/10
s17	s15/10	s16/10	s17/10	s17/10	s14/11	s23/10	s17/10	s17/10	s5/10
s18	s18/10	s19/10	s20/10	s21/10	s18/10	s18/10	s12/00	s18/10	s6/10
s19	s18/10	s19/10	s20/10	s22/10	s19/10	s19/10	s13/11	s19/10	s7/10
s20	s18/10	s19/10	s20/10	s23/10	s20/10	s20/10	s14/11	s20/10	s8/10
s21	s21/10	s22/10	s23/10	s21/10	s18/10	s21/10	s15/10	s21/10	s9/10
s22	s21/10	s22/10	s23/10	s22/10	s19/10	s22/10	s16/10	s22/10	s10/10
s23	s21/10	s22/10	s23/10	s23/10	s20/10	s23/10	s17/10	s23/10	s11/10

First column defines the states (initial state s_0)

First row defines the inputs

Fields s/y denote ‘Post-state/Output’

Inputs:

f0, f1, f2 : FSB switch in position OFF, ON, AUTO

d1, d0 : Cabin decompression true, false

e1, e0 : Excessive altitude true, false

a1, a0 : Auto condition true, false

Outputs:

00 denotes (FSB,RTS)=(0,0)

11 denotes (FSB,RTS)=(1,1)

10 denotes (FSB,RTS)=(1,0)

4.2 Synthetic Example

Application. The following example does not come from a practical application, but has been constructed to illustrate that the reduction of test cases in comparison to the original H-Method can be quite significant. The reference state machine is shown in Table 2.

Safety considerations. We assume that outputs 1 and 2 can be considered as non-critical, so that they can be abstracted to a single output Y . Output 0 is considered as critical.

Table 2 State-transition table for the DFSM presented in the synthetic example in Section 4.2.

	a	b	c	d	e
s0	s1/1	s3/2	s2/0	s4/1	s5/1
s1	s1/1	s3/1	s2/0	s4/2	s5/1
s2	s1/1	s3/1	s2/0	s4/1	s5/1
s3	s1/2	s3/2	s1/0	s4/1	s5/1
s4	s1/2	s3/2	s2/0	s4/1	s5/1
s5	s1/0	s3/1	s0/0	s4/1	s6/1
s6	s1/0	s3/1	s2/0	s4/1	s5/1

First column defines the states (initial state s_0)

First row defines the inputs

Fields s/y denote ‘Post-state/Output’

Comparison H-Method versus Safety-complete H-Method. The reference machine in Table 2 with its 7 states is already minimal, but the minimised abstracted DFSM only has 2 states, this corresponds to a state reduction of 71%. The Wp-Method with assumption $m = n$ required 87 test cases, the H-Methods requires 61 test cases, while the Safety-complete H-Method only requires 31, this corresponds to a test case reduction of approx. 49%.

While this example is of no practical value, it illustrates that test case reductions of approximately 50% are possible when using the Safety-complete H-Method instead of the original H-Method.

4.3 Garage Door Controller

Application. This example has originally been proposed in Jorgensen (2017). We use it here to confirm the fact discussed above, that the Safety-complete H-Method does not lead to test case reductions, if the prime machine of the abstracted DFSM has just as many states as the prime machine of the original reference model.

The garage door controller uses inputs from a remote control, two sensors indicating whether the door has reached the up position or the down position, respectively, and a light sensor indicating whether the door area is crossed while the door is closing or opening. The controller commands the motor to go down, up, stop, or to reverse the down direction to the up direction. Its detailed behaviour is specified in Table 3.

Safety considerations. The only output considered as safety-critical is the command to reverse the down-direction to the up direction. All other outputs can be abstracted to some value Y .

Comparison H-Method versus Safety-complete H-Method. Both the reference model in Table 3 and its abstraction are not minimal. It turns out, however, that the minimised abstracted model still has as many states as the minimised reference model (4 states). As a consequence, both the H-Method and the Safety-complete H-Method requires 13 test cases for $m = n$. The Wp-Method generates 17 test cases.

Table 3 DFSM modelling the garage door controller.

	e1	e2	e3	e4
DU	DC/a1	DU/a3	DU/a3	DU/a3
DD	DO/a2	DD/a3	DD/a3	DD/a3
DSD	DC/a1	DSD/a3	DSD/a3	DSD/a3
DSU	DO/a2	DSU/a3	DSU/a3	DSU/a3
DC	DSD/a3	DD/a3	DC/a1	DO/a4
DO	DSU/a3	DO/a2	DU/a3	DO/a2

Inputs:

e1 : Remote control has been pressed

e2 : Sensor indicates “door reaches down position”

e3 : Sensor indicates “door reaches up position”

e4 : Sensor indicates “light beam crossed”

Outputs:

a1 : Command “start down movement” to motor

a2 : Command “start up movement” to motor

a3 : Command “stop movement” to motor

a4 : Command “reverse down movement to up” to motor

States:

DU : Door is in up position

DD : Door is in down position

DSD : Door is stopped while going down

DSU : Door is stopped while going up

DC : Door is closing

DO : Door is opening

4.4 Statistical Experiments

In addition to the “hand-crafted” DFSMs discussed above, a comprehensive collection of experiments has been performed with DFSMs generated at random. For each DFSM, a safety-abstraction has been generated. Then the Wp-Method, the H-Method, the Safety-complete Wp-Method, and the Safety-complete H-Method have been applied to generate test suites.

For each generation, the following parameters were recorded; these are also shown in the tables below.

$|REF_{MIN}|$ Number of states in the prime machine of the reference DFSM.

$|ABS_{MIN}|$ Number of states in the prime machine of the DFSM abstraction.

$|\Sigma_I|$ Size of the input alphabet.

$m - n$ Hypothesis about the difference between maximal number m of states in the prime machine modelling the true implementation behaviour and number n of states in the prime machine of the reference model.

$|TS_{WP}|$ Test suite size (i.e., number of test cases) of the suite generated using the Wp-Method.

$|TS_H|$ Test suite size of the suite generated using the H-Method.

$|TS_{SWP}|$ Test suite size of the suite generated using the Safety-complete Wp-Method.

$|TS_{SH}|$ Test suite size of the suite generated using the Safety-complete H-Method.

Table 4 Number of test cases generated by different methods and test case reduction compared to H-method in dependency of the size of the abstraction model's prime machine.

$$|\Sigma_I| = 4$$

$$m - n = 0$$

$ ABS_{MIN} $	$ REF_{MIN} $	$ TS_{WP} $	$ TS_{SWP} $	$ TS_H $	$ TS_{SH} $	Red_H%	Red_{H_{total}}%
2.00	99.97	913.60	593.56	796.84	414.27	48.01	51.82
3.00	102.44	1020.45	780.97	811.69	535.65	34.01	37.82
4.00	102.60	1030.44	868.20	837.07	640.30	23.51	27.42
5.00	103.25	1055.04	939.02	862.26	718.09	16.72	20.26
6.00	101.42	1073.24	963.92	840.63	728.93	13.29	16.87
7.00	102.64	1068.38	1020.32	880.71	781.87	11.22	14.34
8.00	103.01	1074.51	1047.98	880.79	809.69	8.07	11.11
9.00	100.76	1033.94	1055.74	866.15	809.37	6.56	9.10
10.00	102.85	1033.20	1089.53	904.97	852.88	5.76	8.10
20.00	103.73	1107.67	1367.51	960.16	943.61	1.72	3.24
30.00	103.92	1079.82	1529.54	975.72	968.98	0.69	1.66

Red_H% Reduction of H-test suite versus Safety-H test suite size in percent, calculated using the formula

$$\mathbf{Red}_H\% = 100 - 100 \cdot \frac{|TS_H|}{|TS_{SH}|}.$$

$|TS_H|_{total}, |TS_{SH}|_{total}$ Total number of test steps occurring in the test suite generated using the H-Method and total number of test steps occurring in the test suite generated using the Safety-complete H-Method, respectively. These parameters are not explicitly shown in the tables presented here, but used to calculate the reduction percentage **Red_{H_{total}}%** described next.

Red_{H_{total}}% Reduction of the number of test steps in percent, calculated using the formula

$$\mathbf{Red}_{H_{total}}\% = 100 - 100 \cdot \frac{|TS_H|_{total}}{|TS_{SH}|_{total}}$$

In Table 4, the reduction of test suite size is recorded in dependency of the size $|ABS_{MIN}|$ of the safety abstraction's prime machine. An input alphabet of size 4 was used, together with the hypothesis $m = n$. The generated reference prime machines have sizes varying between 95 and 105 states. Each line in this table is based on 100 experiments, therefore the values are given as the average over 100 generated reference machines. The reduction results are displayed for variations of $|ABS_{MIN}|$ from 2 to 30.

In Fig. 1, a plot displaying the number of test cases generated by each method in dependency of $|ABS_{MIN}|$ is shown. The plot is based on the same data as presented in Table 4. It can be seen that the Safety-complete Wp Method produces even more test cases than the H-Method, if $|ABS_{MIN}| \geq 9$. This presents clear evidence that the H-Method and its Safety-complete variant outperform the Wp-Method and the Safety-complete Wp-Method.

The values of **Red_H%** show that significant test suite reductions can be expected if $|ABS_{MIN}|$ has a value below $|REF_{MIN}|/10$. The examples from Section 4.1 and Section 4.2 perform better with state reductions $|ABS_{MIN}| = |REF_{MIN}|/6$ for the Fasten Seatbelt Sign application and $|ABS_{MIN}| = |REF_{MIN}|/3.5$ for the synthetic example. We see this as an indication that this type of random generation is not fully

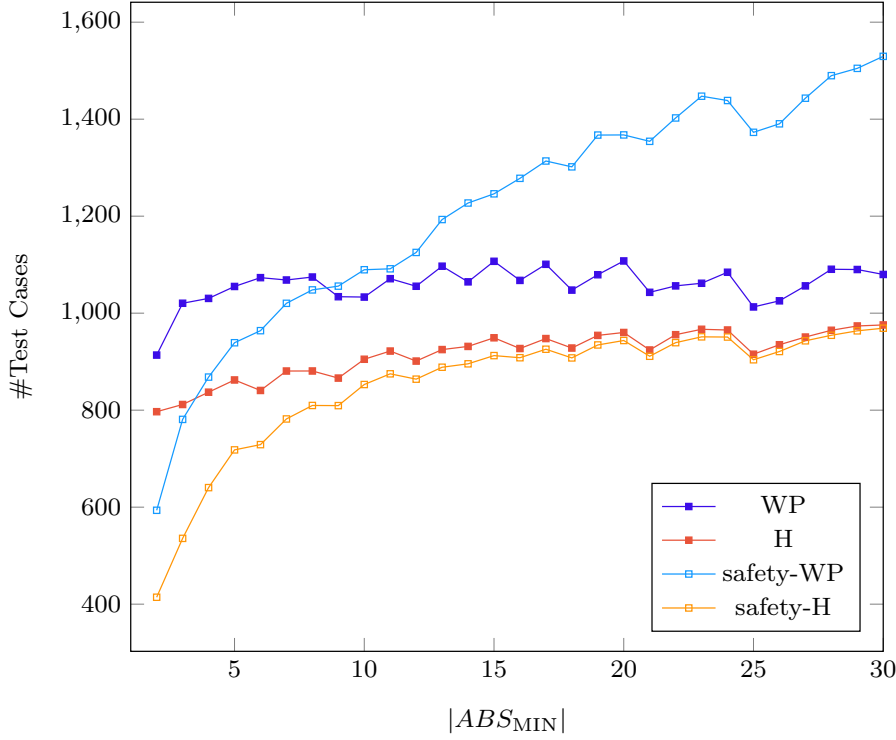


Fig. 1 Number of Test cases in dependency of the size of the safety abstraction’s prime machine, $m = n$.

representative for meaningful DFSM constructions. This hypothesis is confirmed by the experiments documented in Table 5, where the test suite reductions are recorded in dependency of $|REF_{MIN}|$, while $|ABS_{MIN}|$ keeps the constant value 2. Here, the reduction of 34% achieved for the Fasten Seatbelt application described above is achieved for an approximate ratio of $|ABS_{MIN}| = |REF_{MIN}|/4$, whereas the 50% reduction achieved in the synthetic example (Section 4.2) is nearly achieved for a ratio $|ABS_{MIN}| = |REF_{MIN}|/40$. The plot based on the data presented in Table 5 is shown in Fig. 3.

The plots in Fig. 2 and Fig. 4 show that the test step reductions follow the same trend as the test case reductions. The former figure shows the reductions in dependency of the safety abstraction’s size, the latter in dependency on the size of the reference model.

The increasing difference $m - n$ has no significant impact on the test case reduction that can be obtained by using the Safety-complete H-Method instead of the normal one: Fig. 5 shows the reduction curve again in dependency from the size $|ABS_{MIN}|$, but now for the value $m - n = 3$. Comparing this with Fig. 2 shows the same trends for both cases $m = n$ and $m - n = 3$.

Table 5 Number of test cases generated by different methods and test case reduction compared to H-method in dependency of the size of the reference model's prime machine.

$$|ABS_{min}| = 2$$

$$|\Sigma_I| = 4$$

$$m - n = 0$$

$ REF_{MIN} $	$ TS_{WP} $	$ TS_{SWP} $	$ TS_H $	$ TS_{SH} $	Red_H%	Red_HTotal %
2.00	7.00	7.00	7.00	7.00	0.00	0.00
4.00	19.48	13.29	17.82	13.69	23.18	25.47
6.00	32.33	20.94	29.86	20.94	29.87	32.80
8.00	46.15	29.57	43.03	28.19	34.49	37.78
10.00	60.71	38.64	55.91	36.24	35.18	38.37
12.00	74.18	49.05	70.15	44.66	36.34	39.96
14.00	87.82	59.77	82.90	50.47	39.12	42.74
16.00	105.87	68.91	100.04	57.21	42.81	45.99
18.00	115.70	75.00	113.81	69.61	38.84	41.31
19.66	134.05	90.20	126.57	73.27	42.11	45.80
21.57	147.04	100.06	142.35	80.61	43.37	46.88
31.59	219.40	153.41	216.99	124.15	42.79	46.11
41.09	319.50	213.25	295.62	160.31	45.77	49.32
51.03	423.16	276.64	378.06	210.15	44.41	48.35
60.61	499.79	339.02	443.75	243.50	45.13	49.17
70.41	587.89	396.58	538.07	293.83	45.39	49.17
79.79	701.84	457.32	605.98	312.12	48.49	52.25
89.94	813.69	518.85	690.72	364.56	47.22	51.21
97.49	889.05	564.21	756.92	408.52	46.03	49.80

5 Related Work

Complete Testing Theories. The investigation of complete theories for testing against finite state machine models has a long tradition. This started with the W-Method published by Vasilevskii (1973) and Chow (1978), who were the first to show that implementations whose true behaviour is represented by completely specified DFSMs with at most m states can be tested against reference models with $n \leq m$ states, such that any violation of I/O-equivalence will be uncovered by finitely many finite test cases derived from the reference model and the estimate m . Since then, these initial results have been extended to incompletely specified and nondeterministic finite state machines, and to reduction (language inclusion) as an alternative conformance relation to I/O-equivalence. Moreover, many of the new theories guaranteed complete fault coverage with considerably less test cases than needed according to the W-Method.

For the work presented here, the following results are of particular importance. The W-Method is outperformed by the Wp-Method published by Fujiwara et al (1991) and extended by Luo et al (1995) to nondeterministic incomplete FSMs. The key idea to the considerable test case reduction achieved in comparison with the original W-Method consisted in the fact that it is not always necessary to distinguish states reached during a test by means of *all* distinguishing traces contained in a characterisation set. Instead, so-called state identification sets – these are subsets of a characterisation set, distinguishing just one state from all the others – can be applied. In Petrenko et al (1993), the HSI-Method has been presented which yields further test case reductions in comparison to the Wp-Method.

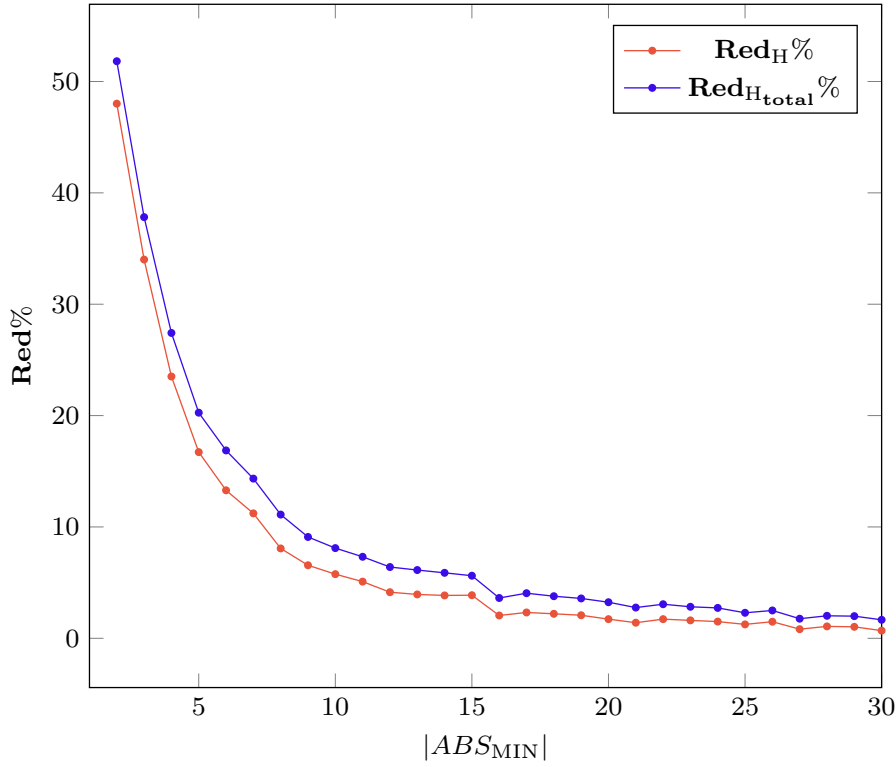


Fig. 2 Test case reduction and test step reduction compared to H-method in dependency of the size of the safety abstraction's prime machine, $m = n$.

The key insight that arbitrary distinguishing traces can be used instead of (subsets of) the characterisation set, in order to check the correctness of target states reached in a test case execution has been perfected in the H-Method published in Dorofeeva et al (2005), leading to a further reduction of test cases needed to guarantee full fault coverage. This motivated our decision to base the safety-complete testing method introduced here on the H-Method. Previous experiments (Huang and Peleska (2017a)) based on the W-Method and the Wp-Method confirm that (1) the H-Method outperforms the Wp-Method and W-Method, as stated in Dorofeeva et al (2005), and (2) that the safety-complete variants of these methods are also outperformed by the safety-complete H-Method described here. It is not surprising that the reductions of test suite sizes achieved with the H-Method carry over to the safety-complete variant: we have shown in Corollary 1 that the H-Method can be considered as a special case of the more general safety-complete H-Method, at least if completely specified DFSMs are considered.

@todo SPY method Simo et al (2012)

Property-oriented Testing. The approach to focus test suites on safety-critical behavioural aspects is a special variant of *property-oriented testing*; see Machado et al (2007) for an overview of this research field. As stated there, the main motivation for property-oriented testing is the possibility to reduce test suite size in comparison

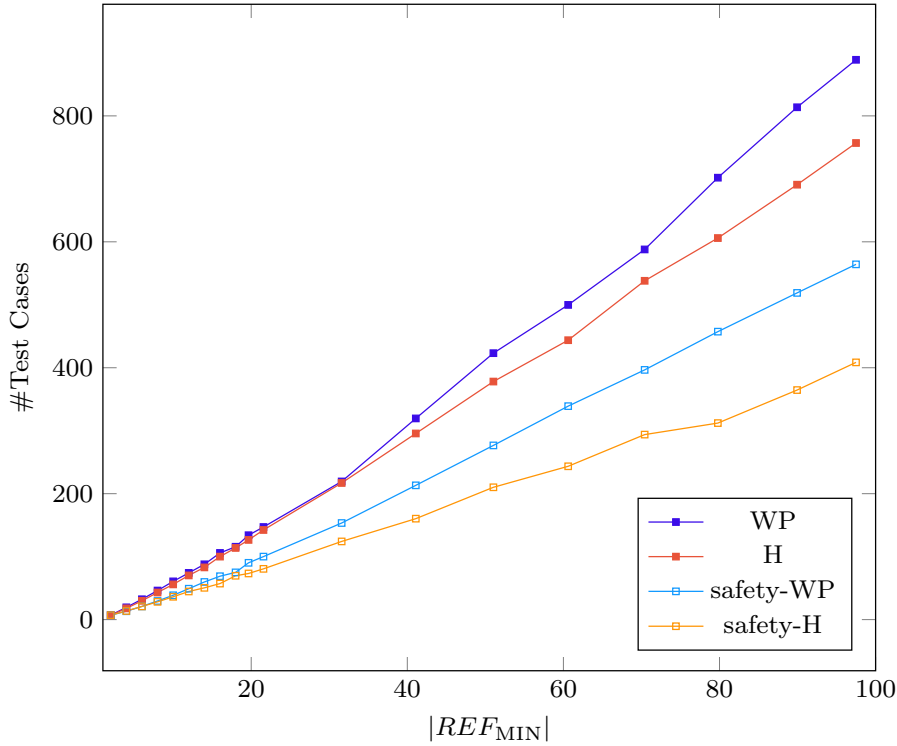


Fig. 3 Number of Test cases in dependency of the size of the reference model’s prime machine, $m = n$.

to a complete model-based conformance test suite: while the latter requires test cases to verify *all* possible behaviours in relation to a reference model, the former only requires to investigate the subset of possible behaviours which is relevant to verify a certain set or properties (i.e. requirements).

A typical approach to property-oriented testing is to state desired system requirements as formulas in a temporal logic (see e.g. Fernandez et al (2003) and Li and Qi (2004)) and construct test cases based on that formula. Since we are only interested in testable properties, the set of properties is usually restricted to safety properties. For linear safety properties (usually expressed as LTL formulas), the test oracle problem is solved by noting that the test executions violating the property (the so-called *bad prefixes*) are inside the language of a finite automaton, as described in (Baier and Katoen, 2008, Section 4.2). This holds for the important class of regular safety properties. In the more general case, property violations can be characterised by accepting states of Büchi automata (Huang et al, 2014, Chapter 4) or Rabin automata (Safr (1988)). For test generation, the subsets of model executions are of interest, where the premises associated with the property under consideration are fulfilled, so that the expected SUT reactions can be observed. To our best knowledge, no general theory which guarantees complete fault coverage for the general class of safety properties has been elaborated yet: the publications Fernandez et al (2003) and Li and Qi (2004), for example, only suggest heuristics

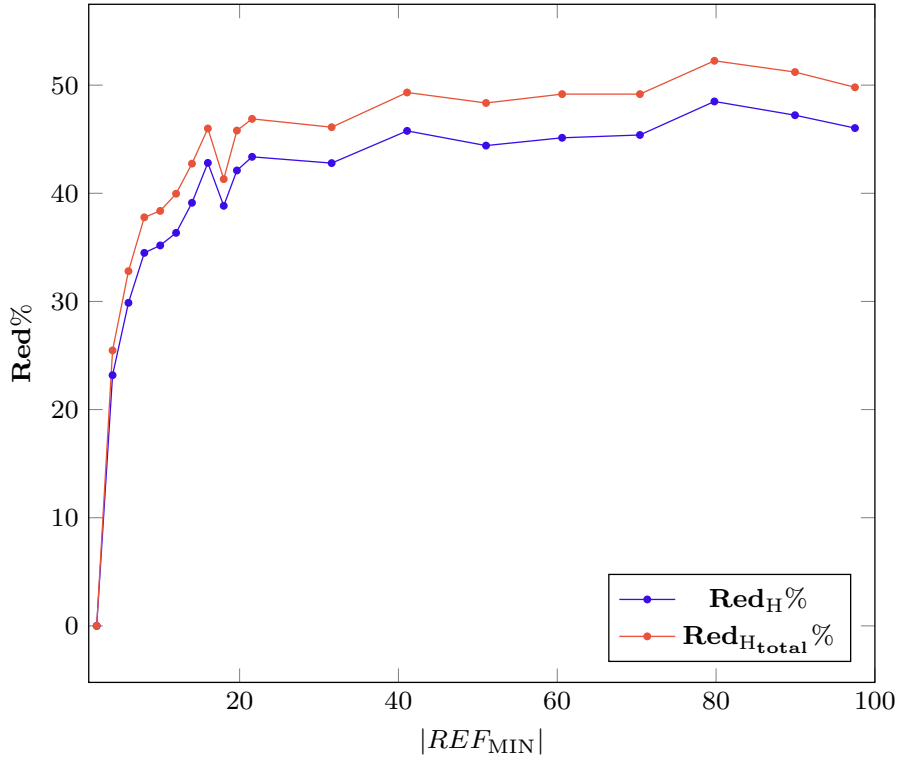


Fig. 4 Test case reduction and test step reduction compared to H-method in dependency of the size of the reference model's prime machine, $m = n$.

for generating suitable test suites, without any guarantees concerning test strength or fault coverage.

The approach presented in this article differs fundamentally from the general approach described above: by specialising on safety properties that can be expressed by means of output abstractions, we can apply the conformance testing approach on a simpler model, instead of using a completely different property-oriented method. As a consequence, the concepts of fault coverage and completeness are very close to those used in general conformance testing, and the completeness proofs can be elaborated in a similar way. As a tradeoff, our safety properties are less general: there are linear safety requirements that cannot be expressed by safety-related output abstractions and their resulting DFSMs. From safety-critical systems testing in the railway and avionic domain, however, we know that the class of safety properties covered in this article fits the practical requirements quite often.

6 Conclusion

We have presented a testing strategy which guarantees to uncover every safety violation when testing an implementation against a deterministic finite state

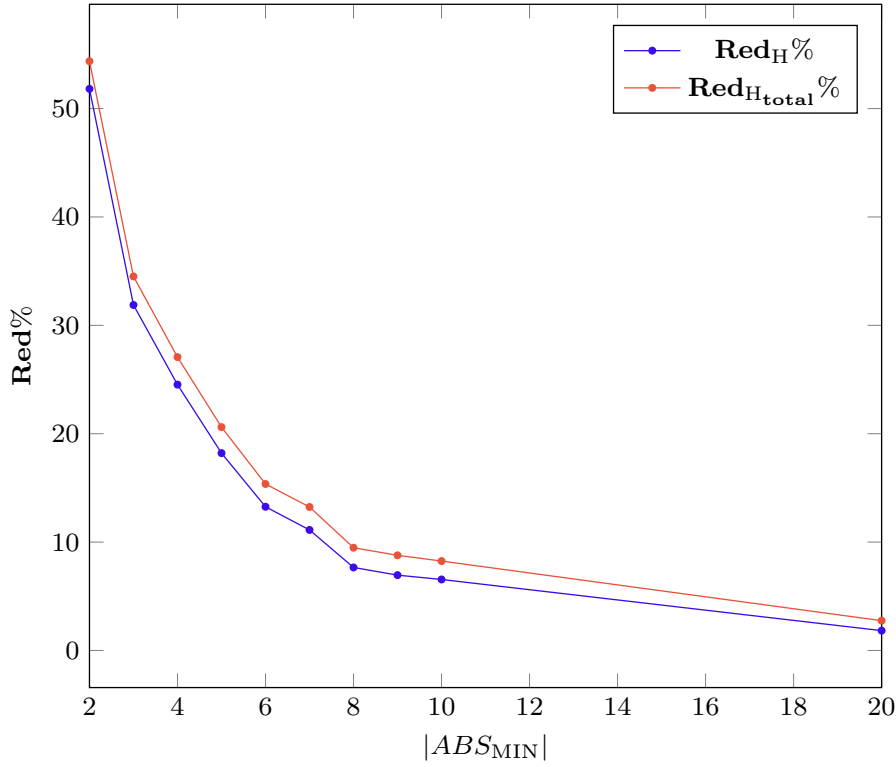


Fig. 5 Test case reduction and test step reduction in dependency of the size of the safety abstraction's prime machine, $m - n = 3$.

machine reference model. These guarantees hold under the assumption that the true behaviour of the implementation, when expressed by a minimised state machine, does not exceed a certain maximum number of states m , in comparison to the number n of states in the minimised reference model. Safety criticality has been modelled by means of a safety-related output abstraction which allows to express that certain outputs can be exchanged by certain others without introducing a safety threat. The new strategy has been derived from the H-Method which is known to guarantee complete fault coverage, while producing smaller test suites, when compared to the well-known older W-Method or Wp-Method. A proof has been presented which shows that – while no longer guaranteeing to uncover *every* violation of input/output equivalence – the new strategy is safety-complete: it will uncover every failure which ends in an erroneous output representing a safety violation.

The experiments show that this Safety-complete H-Method may require significantly fewer test cases than the H-Method (reductions up to 50% have been observed). It has been indicated by another example, however, that this reduction is not guaranteed: the most important factor influencing the decrease of test cases is the reduction of states achieved when transforming the original reference machine to its safety-related abstraction.

The concept described here can be extended to more complex systems whose behaviour is represented by Extended Finite State Machines or, equivalently, by a certain class of Kripke structures over infinite input domains, but with finite domains for internal states and outputs. It has been shown in Huang and Peleska (2017b) that a specific input equivalence class construction technique can be applied, so that any complete testing theory valid for FSMs can be translated to a likewise complete equivalence class partition testing strategy for these systems with Kripke semantics.

References

- Baier C, Katoen J (2008) Principles of model checking. MIT Press
- Chow TS (1978) Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering* SE-4(3):178–186
- Dorofeeva R, El-Fakih K, Yevtushenko N (2005) An improved conformance testing method. In: Wang F (ed) *Formal Techniques for Networked and Distributed Systems - FORTE 2005, 25th IFIP WG 6.1 International Conference, Taipei, Taiwan, October 2-5, 2005, Proceedings, Springer, Lecture Notes in Computer Science*, vol 3731, pp 204–218, DOI 10.1007/11562436_16, URL https://doi.org/10.1007/11562436_16
- Fernandez J, Mounier L, Pachon C (2003) Property oriented test case generation. In: Petrenko A, Ulrich A (eds) *Formal Approaches to Software Testing, Third International Workshop on Formal Approaches to Testing of Software, FATES 2003, Montreal, Quebec, Canada, October 6th, 2003, Springer, Lecture Notes in Computer Science*, vol 2931, pp 147–163, DOI 10.1007/978-3-540-24617-6_11, URL https://doi.org/10.1007/978-3-540-24617-6_11
- Fujiwara S, Bochmann Gv, Khendek F, Amalou M, Ghedamsi A (1991) Test selection based on finite state models. *IEEE Transactions on Software Engineering* 17(6):591–603, DOI 10.1109/32.87284
- Haxthausen AE, Peleska J (2000) Formal Development and Verification of a Distributed Railway Control System. *IEEE Transaction on Software Engineering* 26(8):687–701
- Huang W, Peleska J (2017a) Safety-complete test suites. In: Yevtushenko N, Cavalli AR, Yenigün H (eds) *Testing Software and Systems - 29th IFIP WG 6.1 International Conference, ICTSS 2017, St. Petersburg, Russia, October 9-11, 2017, Proceedings, Springer, Lecture Notes in Computer Science*, vol 10533, pp 145–161, DOI 10.1007/978-3-319-67549-7_9, URL https://doi.org/10.1007/978-3-319-67549-7_9
- Huang W, Peleska J (2017b) Complete model-based equivalence class testing for nondeterministic systems. *Formal Aspects of Computing* 29(2):335–364, DOI 10.1007/s00165-016-0402-2, URL <http://dx.doi.org/10.1007/s00165-016-0402-2>
- Huang W, Peleska J, Schulze U (2014) Contract support for an evolving SoS. Tech. Rep. D34.3, COMPASS Comprehensive Modelling for Advanced Systems of Systems, available under <http://www.compass-research.eu/deliverables.html>
- Jorgensen PC (2017) *The Craft of Model-Based Testing*. CRC Press, Boca Raton
- Li S, Qi Z (2004) Property-oriented testing: An approach to focusing testing efforts on behaviours of interest. In: Beydeda S, Gruhn V, Mayer J, Reussner RH, Schweiggert F (eds) *Testing of Component-Based Systems and Soft-*

- ware Quality, Proceedings of SOQUA 2004 (First International Workshop on Software Quality) and TECOS 2004 (Workshop Testing Component-Based Systems), GI, LNI, vol 58, pp 191–206, URL <http://subs.emis.de/LNI/Proceedings/Proceedings58/article3512.html>
- Luo G, Bochmann G, Petrenko A (1994) Test selection based on communicating nondeterministic finite-state machines using a generalized wp-method. *IEEE Trans Software Eng* 20(2):149–162, DOI 10.1109/32.265636, URL <http://doi.ieeecomputersociety.org/10.1109/32.265636>
- Luo G, Petrenko A, v Bochmann G (1995) Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines, Springer US, Boston, MA, pp 95–110. DOI 10.1007/978-0-387-34883-4_6, URL https://doi.org/10.1007/978-0-387-34883-4_6
- Machado PDL, Silva DA, Mota AC (2007) Towards Property Oriented Testing. *Electronic Notes in Theoretical Computer Science* 184(Supplement C):3–19, DOI 10.1016/j.entcs.2007.06.001, URL <http://www.sciencedirect.com/science/article/pii/S157106610700432X>
- Peleska J, Huang WL (2017) Test Automation - Foundations and Applications of Model-based Testing. University of Bremen, lecture notes, available under <http://www.cs.uni-bremen.de/agbs/jp/papers/test-automation-huang-peleska.pdf>
- Petrenko A, Yevtushenko N, Lebedev A, Das A (1993) Nondeterministic state machines in protocol conformance testing. In: Rafiq O (ed) *Protocol Test Systems, VI, Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems*, Pau, France, 28-30 September, 1993, North-Holland, IFIP Transactions, vol C-19, pp 363–378
- Safra S (1988) On the complexity of omega-automata. In: *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Washington, DC, USA, SFCS '88, pp 319–327, DOI 10.1109/SFCS.1988.21948, URL <https://doi.org/10.1109/SFCS.1988.21948>
- Simo A, Petrenko A, Yevtushenko N (2012) On reducing test length for FSMs with extra states. *Software Testing, Verification and Reliability* 22(6):435–454, DOI 10.1002/stvr.452, URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.452>
- Vasilevskii MP (1973) Failure diagnosis of automata. *Kibernetika (Transl)* 4:98–108