ELEKTROTEHNIŠKI VESTNIK 83(3): 115–121, 2016
ORIGINAL SCIENTIFIC PAPER

# An Improved State-Counting-Based Construction of Complete Test Suites for FSM Implementations

**Monika Kapus-Kolar**

*Jožef Stefan Institute, Department of Communication Systems, Jamova 39, SI-1111 Ljubljana, Slovenia*
*E-mail: monika.kapus-kolar@ijs.si*

**Abstract.** The paper addresses black-box conformance testing of objects specified as an arbitrary observable finite-state machine. It proposes a new state-counting-based method for the construction of complete test suites. The method brings new options for the definition of conformance, for assumptions on the object under test and for test preambles, and a strategy against harmfully redundant members of the employed preamble sets. Compared to similar methods, it is more widely applicable and customizable and in many cases able to produce a much shorter test suite.

**Keywords:** finite-state machine, conformance testing, conformance relation, complete test suite, state counting

**Izboljšano na štetju stanj temelječe snovanje popolnih nizov preizkusov za izvedbe končnih avtomatov**

Članek obravnava preizkušanje skladnosti po principu črne škatle za objekte, specificirane kot poljuben končni avtomat. Predlaga novo na štetju stanj temelječo metodo za snovanje popolnih nizov preizkusov. Metoda prinaša nove možnosti za definicijo skladnosti, za predpostavke o preizkušanem objektu in za uvodne dele preizkusov ter strategijo proti škodljivo odvečnim članom uporabljenih množic uvodnih delov preizkusov. V primerjavi s podobnimi je širše uporabna, bolj prilagodljiva in v mnogih primerih sposobna zasnovati dosti krajši niz preizkusov.

## 1 INTRODUCTION

Observable finite-state machines (OFSMs) are abstract machines that to each input specified in their current state reply with an output and the corresponding change of the state. In model-based black-box testing, they are widely employed both as models of what the object under test is supposed to be and as models of what it might be in the reality. In this paper, we propose a new complete test suite (CTS) construction method that is able to handle any specification OFSM and for the OFSM under test assumes only that it has at most a given number of reachable states and never refuses a relevant input without previously producing an erroneous output. Currently, the only such method is the state-counting-based method of [1] (with a slight improvement proposed in [2]), for which theoretical foundations are laid in [3]. In the following, the method of [1] improved as suggested in [2] is called 'the old method'.

*Received 22 January 2016*
*Accepted 23 March 2016*

In the old method, the only two conformance relations foreseen are quasi-equivalence and quasi-reduction, and they are covered with two separate procedures. The new method covers a much wider class of the conformance relation, with a single procedure. Besides, it is able to exploit additional assumptions on the extent to which the specified input/output sequences (IOSs) diverge (i.e., lead the OFSM to different states) in the implementation. As a third generalization, it is no longer assumed that the members of the initially constructed sets of test preambles must be IOSs taking the specification OFSM to a definitely reachable state. Unlike the old method, the new one also has a strategy against harmfully redundant elements of the preamble sets.

The new method was developed in [4], as an efficient specialization of our there proposed and formally proved new generic CTS construction method. It is presented in Section 3, after Section 2 presents our notation and definitions. In Section 3, we formally specify also the old method, by specifying its virtually only differences from the new one. Section 4 provides examples showing how useful the novelties of our method can be. Section 5 comprises a discussion and conclusions.

## 2 NOTATION AND DEFINITIONS

Definitions 1-5 introduce the basics of our notation.

**Definition 1.**

- $\mathcal{X}$ denotes the universe of inputs.
- An IO is a sequence of an input and an output.
- $Q$, $s$, $x$, $X$, $y$, $U$, $z$ and $Z$, possibly decorated, denote, respectively, an OFSM, a state, an input, an input set, an output, an IO set, an IOS and an IOS set.

**Definition 2.** For each OFSM $Q$:

- $st(Q)$ denotes the set of all its reachable states.
- $init(Q)$ denotes the initial state.
- For each state $s \in st(Q)$, $ios(s)$ denotes the set of all IOSs executable from $s$.
- $ios(Q)$ denotes $ios(init(Q))$.
- For each IOS $z \in ios(Q)$, $ios(Q, z)$ denotes the IOS set $\{z' | zz' \in ios(Q)\}$.
- For each IOS set $Z \subseteq ios(Q)$, $ioss(Q, Z)$ denotes the IOS set set $\{ios(Q, z) | z \in Z\}$.
- For each IOS $z \in ios(Q)$, $in(Q, z)$ denotes the input set $\{x | \exists y : (zxy \in ios(Q))\}$.
- For each IOS $z \in ios(Q)$ and input $x$, $out(Q, z, x)$ denotes the output set $\{y | zxy \in ios(Q)\}$.
- For each IOS set $Z \subseteq ios(Q)$, $end(Q, Z)$ denotes the set of all states in which $Q$ can be immediately after executing from $init(Q)$ a member of $Z$.

**Definition 3.** For each IOS $z = x_1y_1x_2y_2 \dots x_ky_k$:

- $ln(z)$ denotes its length $k$, with $\varepsilon$ denoting an IOS of the length 0.
- $pf(z)$ denotes the set of all IOSs that are its prefix.
- For each IOS $z'$, $z' < z$ iff (i.e. if and only if) $z' \in (pf(z) \setminus \{z\})$.

**Definition 4.** For each IOS set $Z$:

- $max(Z)$ denotes the set $\{z | (z \in Z) \wedge \neg\exists z' \in Z : (z < z')\}$ of all its maximal members.
- $pf(Z)$ denotes the IOS set $\{z | \exists z' \in Z : (z \in pf(z'))\}$.
- For each IOS $z \in pf(Z)$, $io(Z, z)$ denotes the IO set $\{xy | zxy \in pf(Z)\}$.

**Definition 5.** For each IO set U:

- $in(U)$ denotes the input set $\{x | \exists y : (xy \in U)\}$.
- For each input $x$, $out(U, x)$ denotes the output set $\{y | xy \in U\}$.
- For each input $x$, $io(U, x)$ denotes the IO set $\{xy | xy \in U\}$.

Definitions 6-9 gradually introduce our candidates for the conformance relation.

**Definition 6.** A well-formed binary relation for IO sets (WBRIO) is such a set $R$ of IO set pairs that for each IO set pair $(U, U')$, all the following is true:

(1) $(U, U') \in R$ iff $(io(U, x), io(U', x)) \in R$ for each input $x \in in(U')$.
(2) $(U, U') \in R$ only if $out(U, x) \subseteq out(U', x)$ for each input $x \in in(U')$.
(3) If $out(U, x) = out(U', x)$ for each input $x \in in(U')$, then $(U, U') \in R$.
(4) $(U, U'') \in R$ for each IO set $U''$ with $((U, U') \in R) \wedge ((U', U'') \in R)$.

**Definition 7.** For each input set $X$, $br(X)$ denotes the WBRIO consisting of all IO set pairs $(U, U')$ that for each input $x \in in(U')$ satisfy all the following:

(1) $\emptyset \subset out(U, x) \subseteq out(U', x)$
(2) If $x \in X$ then $out(U, x) = out(U', x)$.

**Definition 8.** For each IOS set pair $(Z, Z')$ and WBRIO $R$:

- For each IOS $z \in pf(Z')$, $Z \sqsupseteq_{R,z} Z'$ iff for each IOS $z'$ with $(z' < z) \wedge (z' \in pf(Z))$, $(io(Z, z'), io(Z', z')) \in R$.
- For each IOS set $Z'' \subseteq pf(Z')$, $Z \sqsupseteq_{R,Z''} Z'$ iff $Z \sqsupseteq_{R,z} Z'$ for each IOS $z \in Z''$.
- $Z \sqsupseteq_R Z'$ iff $Z \sqsupseteq_{R,Z'} Z'$.

**Definition 9.** For each OFSM pair $(Q, Q')$:

- For each WBRIO $R$ and IOS $z \in ios(Q')$, $Q \sqsupseteq_{R,z} Q'$ iff $ios(Q) \sqsupseteq_{R,z} ios(Q')$.
- For each WBRIO $R$ and IOS set $Z \subseteq ios(Q')$, $Q \sqsupseteq_{R,Z} Q'$ iff $ios(Q) \sqsupseteq_{R,Z} ios(Q')$.
- For each WBRIO $R$, $Q \sqsupseteq_R Q'$ iff $Q \sqsupseteq_{R,ios(Q')} Q'$.
- $Q$ is quasi-equivalent to $Q'$ iff $Q \sqsupseteq_{br(\mathcal{X})} Q'$.
- $Q$ is a quasi-reduction of $Q'$ iff $Q \sqsupseteq_{br(\emptyset)} Q'$.

Definitions 10-15 introduce concepts related to the CTS construction.

**Definition 10.**

- $M$ denotes the specification OFSM.
- $N$ denotes the OFSM under test.
- $\sqsupseteq_A$ denotes the conformance relation, with $A$ presumably a given WBRIO.
- $\mathcal{I}$ denotes the set of all candidates for $N$, presumably the set of all OFSMs $Q$ that satisfy all the following:
  (1) For each IOS $z \subseteq (ios(M) \cap ios(Q))$, $in(M, z) \subseteq in(Q, z)$.
  (2) $|st(Q)|$ is not more than a given upper bound $m$, presumably a non-zero natural.
  (3) For each IOS set $Z \subseteq (ios(M) \cap ios(Q))$, $|end(Q, Z)|$ is not more than a given upper bound $ub(Z)$, presumably a natural with $1 \leq ub(Z) \leq m$ whose default value is $m$.

**Definition 11.** For each OFSM $Q$:

- For each IOS $z = x_1y_1x_2y_2 \dots x_ky_k$, $imp(Q, z)$ denotes the set of all IOSs in $ios(Q)$ that are of the form $x_1y_1x_2y_2 \dots x_{i-1}y_{i-1}x_iy_i'$ with $0 \leq i \leq k$, i.e., the set representing the implementation of $z$ in $Q$.
- For each IOS set $Z$, $imp(Q, Z)$ denotes the IOS set $\{z | \exists z' \in Z : (z \in imp(Q, z'))\}$.

**Definition 12.**

- A test is a member of $ios(M)$.
- A test suite is a set of tests.
- A given OFSM $Q$ passes a given test suite $Z$ iff $Q \sqsupseteq_{A,Z} M$, i.e., iff $Q \sqsupseteq_{A, max(imp(M,Z)) \setminus \{\varepsilon\}} M$.
- Accordingly, we define the length $len(Z)$ of a given test suite $Z$ as $\Sigma_{z \in (max(imp(M,Z)) \setminus \{\varepsilon\})}(1 + ln(z))$, counting also the resets that are assumed to precede each test in $max(imp(M, Z)) \setminus \{\varepsilon\}$.

- A given test suite $Z$ is complete iff $Q \sqsupseteq_A M$ for each OFSM $Q \in \mathcal{I}$ with $Q \sqsupseteq_{A,Z} M$.

**Definition 13.** A given state $s \in st(M)$ is definitely reachable if there exists an IOS set $Z \subseteq ios(M)$ satisfying all the following:

(1) $end(M, Z) = \{s\}$
(2) $Z \cap Z' \neq \emptyset$ for each IOS set $Z' \subseteq imp(M, Z)$ with $(Z' \sqsupseteq_A imp(M, Z)) \wedge (pf(Z') = Z')$.

**Definition 14.** For each IOS set $Z$ and IOS pair $\{z, z'\} \subseteq pf(Z)$, a $(Z, z, z')$-distinguisher is a pair $(z'', x)$ satisfying all the following:

(1) $z'' \in (ios(Z, z) \cap ios(Z, z'))$
(2) $x \in (in(Z, zz'') \cap in(Z, z'z''))$
(3) $out(Z, zz'', x) \neq out(Z, z'z'', x)$

**Definition 15.** For each IOS pair $\{z, z'\} \subseteq ios(M)$, a $\{z, z'\}$-separator is a set $D$ satisfying all the following:

(1) Every member of $D$ is an $(ios(M), z, z')$-distinguisher.
(2) For the IOS set
$Z = \{z''z'''xy | (z'' \in \{z, z'\}) \wedge ((z''', x) \in D) \wedge (y \in out(M, z''z''', x))\}$
and each IOS set $Z' \subseteq imp(M, Z)$ with
$(Z' \sqsupseteq_A imp(M, Z)) \wedge (pf(Z') = Z') \wedge (\{z, z'\} \subseteq Z')$,
at least one of the following is true:
   - $D$ comprises a $(Z', z, z')$-distinguisher.
   - $D$ comprises a $(z''xyz''', x')$ with $(z'', x)$ a $(Z', z, z')$-distinguisher.

## 3 THE NEW METHOD

The new method is specified in Fig. 1. The constructed CTS is the $T'$ conceived in Step 6. In Steps 1-5, one constructs an IOS set $T \subseteq ios(M)$ and a set $D$ of IOS pairs $\{z, z'\} \subseteq ios(M)$ with a $\{z, z'\}$-separator. For each IOS $z \in T$, $T'$ is supposed to check $N \sqsupseteq_{A,z} M$. For each pair $\{z, z'\} \in D$, $T'$ is supposed to check that in the case of $\{z, z'\} \subseteq ios(N)$, $z$ and $z'$ diverge in $N$. $(T, D)$, hence, defines a set of very simple test goals, the goals into which Steps 1-5 decompose the goal that $T'$ should check $N \sqsupseteq_A M$. In Step 6, each of the simple goals is satisfied in a most simple way.

The contributors of the elements of $T$ and $D$ are the IOS sets $K$ collected in the set $\mathcal{K}$ constructed in Steps 1-3. The sets in $\mathcal{K}$ are subsets of $ios(M)$ and we call their elements preambles, because the tests in $T'$ are constructed by their extension. For the optimality of $\mathcal{K}$, there are conflicting criteria:

- It is desirable that it is small, but one must secure that in the case of $N \sqsupseteq_A M$, there is a $K \in \mathcal{K}$ with $K \subseteq ios(N)$. If there are multiple ways for implementing $ios(M)$ correctly, one might prefer a $\mathcal{K}$ with more than one element. In any case, for each IOS set pair $\{K, K'\} \subseteq \mathcal{K}$ with $K \subset K'$, $K'$ is a redundant member of $\mathcal{K}$.
- For each $K \in \mathcal{K}$, it is desirable that it is small, but it must comprise at least $\varepsilon$ and it can be helpful if it has at least one large subset $K'$ with a $\{z, z'\}$-separator for each $\{z, z'\} \subseteq K'$ with $z \neq z'$. In any case, for each subset $\{z, z'\}$ of a $K \in \mathcal{K}$ with $(z \neq z') \wedge (z' \neq \varepsilon) \wedge (ios(M, z) \sqsupseteq_A ios(M, z'))$, $z'$ is a redundant member of $K$.
- The optimality of $\mathcal{K}$ also depends on the exact nature of the available separators, but this is not considered in the method.

In Step 1, the method constructs the initial version of $\mathcal{K}$, with the Fig. 2 procedure. Its strategy in the step is to make $\mathcal{K}$ small, but still sufficient, and its individual members $K$ small, but still possessing a promising $end(M, K)$. Unlike in the old method, those IOSs in $ios(M)$ that do not lead $M$ to a definitely reachable state are also considered for inclusion into the elements of $\mathcal{K}$.

In Step 3 of the method, one forms a plan how

1. With the Fig. 2 procedure, obtain a $\mathcal{K}$.
2. $\mathcal{K}' \leftarrow \emptyset$
3. While one exists, take an IOS set $K \in (\mathcal{K} \setminus \mathcal{K}')$ for which there is no IOS set $K' \in (\mathcal{K} \setminus \mathcal{K}')$ with $ioss(M, K) \subset ioss(M, K')$ and do the following:
   i. With the Fig. 3 procedure, obtain the relevant part $K'$ of $K$ and a plan $(V, W)$ for it.
   ii. $V^{K'} \leftarrow V$; $W^{K'} \leftarrow W$
   iii. While one exists, delete from $\mathcal{K}$ an IOS set $K'' \in (\mathcal{K} \setminus \mathcal{K}')$ with $ioss(M, K') \subseteq ioss(M, K'') \subseteq ioss(M, K)$ and do the following:
      a. $K''' \leftarrow \{z | (z \in K'') \wedge \exists z' \in K' : (ios(M, z) = ios(M, z'))\}$
      b. $k(K''') \leftarrow K'$; $\mathcal{K} \leftarrow (\mathcal{K} \cup \{K'''\})$
      c. $\mathcal{K} \leftarrow \{Z | (Z \in \mathcal{K}) \wedge \neg\exists Z' \in \mathcal{K} : (Z' \subset Z)\}$
      d. $\mathcal{K}' \leftarrow (\mathcal{K}' \cup \{K'''\})$
4. $T \leftarrow \{\alpha\beta | \exists K \in \mathcal{K}, (\alpha', \beta) \in V^{k(K)} : (ios(M, \alpha) = ios(M, \alpha'))\}$
5. $D \leftarrow \{\{\alpha_1 z, \alpha_2 z'\} | \exists K \in \mathcal{K}, (\alpha_1', z, \alpha_2', z') \in W^{k(K)} : (\{\alpha_1, \alpha_2\} \subseteq K) \wedge (ios(M, \alpha_1) = ios(M, \alpha_1')) \wedge (ios(M, \alpha_2) = ios(M, \alpha_2'))\}$
6. Let $T' \leftarrow T$ and then for each IOS pair $\{z, z'\} \in D$:
   1. Choose a $\{z, z'\}$-separator.
   2. For each member $(z'', x)$ of the separator:
      i. Choose an output $y \in out(M, zz'', x)$ and an output $y' \in out(M, z'z'', x)$.
      ii. $T' \leftarrow (T' \cup \{zz''xy, z'z''xy'\})$

Figure 1. The new method

1. $S \leftarrow st(M)$
2. While one exists, take a state pair $\{s, s'\} \subseteq S$ with $(s \neq init(M)) \wedge (s' \neq s) \wedge (ios(s') \sqsupseteq_A ios(s))$ and delete $s$ from $S$.
3. For each state $s \in S$, choose a test set $Z_s$ satisfying all the following:
   (1) $s \in end(M, Z_s)$
   (2) $Z \cap Z_s \neq \emptyset$ for each IOS $Z \subseteq imp(M, Z_s)$ with $(Z \sqsupseteq_A imp(M, Z_s)) \wedge (pf(Z) = Z)$.
   (3) If $s = init(M)$ then $Z_s = \{\varepsilon\}$.
   The default is to minimize, in the given order, $|end(M, Z_s)|$, $|Z_s|$ and $len(Z_s)$.
4. $P \leftarrow \{z | \exists s \in S : (z \in Z_s)\}$
5. $\mathcal{K} \leftarrow \{(P \cap Z) | (Z \subseteq imp(M, P)) \wedge (pf(Z) = Z) \wedge (Z \sqsupseteq_A imp(M, P)) \wedge$
6. $\mathcal{K} \leftarrow \{Z | (Z \in \mathcal{K}) \wedge \neg\exists Z' \in \mathcal{K} : (Z' \subset Z)\}$
7. For each IOS set $K \in \mathcal{K}$:
   i. $\mathcal{K} \leftarrow (\mathcal{K} \setminus \{K\})$
   ii. While one exists, take an IOS pair $\{z, z'\} \subseteq K$ with $(z \neq \varepsilon) \wedge (z' \neq z) \wedge (ios(M, z') \sqsupseteq_A ios(M, z))$ and delete $z$ from $K$.
   iii. $\mathcal{K} \leftarrow (\mathcal{K} \cup \{K\})$
8. $\mathcal{K} \leftarrow \{Z | (Z \in \mathcal{K}) \wedge \neg\exists Z' \in \mathcal{K} : (Z' \subset Z)\}$

Figure 2. Construction of the initial version of $\mathcal{K}$

to accommodate individual $K \in \mathcal{K}$, i.e., what their contributions to $T$ and $D$ should be. Actually, for each IOS set set $Y \in \{ioss(M, K) | K \in \mathcal{K}\}$, one forms a plan for just one $K \in \mathcal{K}$ with $ioss(M, K) = Y$, whereas for the remaining $K' \in \mathcal{K}$ with $ioss(M, K') = Y$, one just sets $k(K')$ to $K$, thereby deciding that they will be accommodated analogously. $k(K)$ is also set to $K$.

Whenever the Fig. 1 procedure decides to construct a plan for a $K \in \mathcal{K}$, it calls the Fig. 3 procedure. The procedure constructs the plan and possibly detects some redundant members of $K$, so that the plan is actually for $K'$, the computed relevant part of $K$. After $K'$ and its plan are computed, the Fig. 1 procedure reduces $K$ to $K'$, analogously reduces the remaining $K'' \in \mathcal{K}$ with $ioss(M, K') \subseteq ioss(M, K'') \subseteq ioss(M, K)$ and deletes every member of $\mathcal{K}$ that consequently becomes redundant.

In the Fig. 3 procedure, one builds the relevant part $K'$ of the given $K$ and a plan $(V, W)$ for it gradually. Initially, $K'$ comprises just $\varepsilon$, whereas $V$ and $W$ are empty sets. For each IOS $\alpha \in K$ already in $K'$, giving priority to the maximal ones among the yet unconsidered current members of $K'$, one then constructs the following:

1. A set $B$ comprising each IOS $\beta \in ios(M, \alpha)$ that is just enough long that $ios(M, \alpha\beta) = \emptyset$ or one can construct for it a specifically formed triplet $(L, C, W')$ (the conditions which the triplet is supposed to satisfy originate in a state-counting-based conformance testing principle suggested in [3] and improved and generalized in [4]). For each $\beta \in B$, $(\alpha, \beta)$ is added to $V$, where it specifies that $\alpha\beta$ is to be included into $T$.
2. For each $\beta \in B$ with $ios(M, \alpha\beta) \neq \emptyset$, the required $(L, C, W')$. Note the following:
   i. $L$ is a subset of $K$ and each of its members is added to $K'$.
   ii. $C$ is a subset of $pf(\alpha\beta)$ with $|C| > ub(L \cup C) - |L|$. Unlike the old method, we foresee also the possibility of $ub(L \cup C) < m$. In such a case, $\beta$ can be shorter and $C$ smaller.
   iii. $W'$ is a set of 4-tuples $(\alpha', z, \alpha'', z'')$ with $(\{\alpha', \alpha''\} \subseteq (L \cup \{\alpha\})) \wedge (\{\alpha' z, \alpha'' z'\} \subseteq \{L \cup C\}$ and an $\{\alpha' z, \alpha'' z'\}$-separator. Each such 4-tuple is added to $W$, where it specifies that $\{\alpha' z, \alpha'' z'\}$ is to be included into $D$.

Note that the plan $(V, W)$ constructed for the final $K'$,

1. $V \leftarrow \emptyset$; $W \leftarrow \emptyset$; $K' \leftarrow \{\varepsilon\}$; $K'' \leftarrow \emptyset$
2. While one exists, add to $K''$ an IOS $\alpha \in max(K' \setminus K'')$ and do the following:
   i. $B \leftarrow \{\varepsilon\}$;
   ii. While $B \neq \emptyset$, delete from $B$ an IOS $\beta$ and then if $ios(M, \alpha\beta) = \emptyset$ then let $V \leftarrow (V \cup \{(\alpha, \beta)\})$ else if there is a triplet $(L, C, W')$ satisfying all the following:
      (1) $L \subseteq K$
      (2) $\emptyset \subset C \subseteq pf(\alpha\beta) \setminus pf(\alpha)$
      (3) $|C| > ub(L \cup C) - |L|$
      (4) $W'$ is a set of 4-tuples $(\alpha', z, \alpha'', z')$ satisfying all the following:
         (a) $\{\alpha', \alpha''\} \subseteq (L \cup \{\alpha\})$
         (b) $\{\alpha' z, \alpha'' z'\} \subseteq (L \cup C)$
         (c) There is an $\{\alpha' z, \alpha'' z'\}$-separator.
      (5) $\{\alpha', \varepsilon, \alpha'', \varepsilon\} \in W'$ for each IOS pair $\{\alpha', \alpha''\} \subseteq L$ with $\alpha' \neq \alpha''$.
      (6) $\{\alpha, z, \alpha, z'\} \in W'$ for each IOS pair $\{\alpha z, \alpha z'\} \subseteq C$ with $(z < z') \wedge (ios(M, \alpha z) \not\sqsupseteq_A ios(M, \alpha z'))$.
      (7) $\{\alpha', \varepsilon, \alpha, z\} \in W'$ for each IOS pair $\{\alpha', \alpha z\}$ with $(\alpha' \in L) \wedge (\alpha z \in C) \wedge (ios(M, \alpha') \not\sqsupseteq_A ios(M, \alpha z))$.
      then do all the following:
      a. Choose such a triplet $(L, C, W')$. The default is to minimize, in the given order, $|L \setminus K'|$ and $|W' \setminus W|$.
      b. $K' \leftarrow (K' \cup L)$
      c. $V \leftarrow (V \cup \{(\alpha, \beta)\})$; $W \leftarrow (W \cup W')$
      else $B \leftarrow (B \cup \{\beta xy | xy \in io(M, \alpha\beta)\})$

Figure 3. Computation of the relevant part $K'$ of a $K$ and conception of a plan $(V, W)$ for its contributions to $(T, D)$

subsequently called $(V^{K'}, W^{K'})$, explicitly refers to individual members of $K'$. This is to make it possible that the remaining $K'' \in \mathcal{K}$ with $ioss(M, K') \subseteq ioss(M, K'') \subseteq ioss(M, K)$ are handled analogously to $K$. Like the relevant part of $K$, their relevant parts contribute to $T$ and $D$ only if they are not subsequently deleted from $\mathcal{K}$. In the old method, there is no optimization of $\mathcal{K}$ beyond its initial version.

The ways in which the old method differs from the new one are virtually just the following (plus the consequently possible procedural simplifications):

1. For the conformance relation $\sqsupseteq_A$, one assumes $A \in \{br(\emptyset), br(\mathcal{X})\}$.
2. For each IOS set $Z \subseteq ios(M)$, one assumes $ub(Z) = m$.
3. In Step 1 of the Fig. 2 procedure, $S$ is initialized not to $st(M)$, but to the set of all definitely reachable states of $M$.
4. In Step 1 of the Fig. 3 procedure, $K'$ is initialized not to $\{\varepsilon\}$, but to $K$.

## 4 EXAMPLES SHOWING HOW USEFUL THE NOVELTIES CAN BE

Each of the example OFSMs $Q$ will be presented in the usual way, as a rooted directed graph in which each state of $Q$ is represented as a vertex, the initial state of $Q$ is represented as the root vertex and each transition of $Q$, executing an IO $xy$ and leading from a state $s$ to a state $s'$, is represent as an edge labelled with $x/y$ and leading from the $s$ vertex to the $s'$ vertex.

**Example 1.** The common property of quasi-equivalence and quasi-reduction is that for each of the specified inputs, the legal responses are only the specified alternative outputs. The difference is that in the first case, all the alternatives need to be implemented in the OFSM under test, whereas in the second case, it suffices to implement just one. We observe that to instead prescribe 'all or at least three', for example, would also be a practically interesting option. This conformance relation also belongs to the class that the new method can handle.

**Example 2.** Suppose that $M$ is the OFSM in Fig. 4(a), $A = br(\emptyset)$, $m = 2$ and the IOSs starting with $x_1y_1$ are considered more important than those starting with $x_1y_2$, for example because $y_1$ is an alarm, whereas $y_2$ is not. The old method cannot be instructed to check the implementation of the less important IOSs to a desired extent less thoroughly, whereas the new method can. For this, one would for each such IOS $z$ set $ub(pf(z))$ to an appropriate value of less than $m$, let's say to 1. The best $T'$ that one can obtain with the new method is then $\{x_1y_1x_2y_3x_2y_3, x_1y_2x_3y_4\}$, whereas with $ub(Z) = m$ for each IOS set $Z \subseteq ios(M)$, the best possible $T'$ is $\{x_1y_1x_2y_3x_2y_3, x_1y_2x_3y_4x_3y_4\}$.

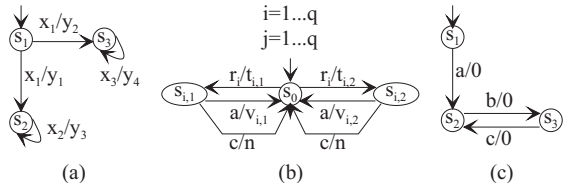


Figure 4. Three example OFSMs

**Example 3.** Suppose that $M$ is the OFSM depicted in Fig. 4(b), with $q$ a given non-zero natural. The OFSM specifies, for example, an agent selling vouchers for transactions of types 1 to $q$ and acting as follows:

- For each $1 \leq i \leq q$, the agent can in the initial state accept a request $r_i$ for a voucher for a type $i$ transaction.
- For each voucher request $r_i$, the agent first returns its terms of sale, by an unknown algorithm choosing between $t_{i,1}$ and $t_{i,2}$. By consequently changing its state to $s_{i,1}$ or $s_{i,2}$, respectively, it records the offer and waits for an input $a$ indicating that the client accepts the terms or an input $c$ indicating that the client cancels the request.
- Upon receiving $a$ in a state $s_{i,j}$, the agent issues a voucher $v_{i,j}$ for a type $i$ transaction under terms $t_{i,j}$ and returns to the initial state.
- Upon receiving $c$ in a state $s_{i,j}$, the agent issues $n$, to indicate its readiness for the next request, and returns to the initial state.

Suppose that $A = br(\emptyset)$ and $m \geq 2q+1$. If one executes the new method optimally, one makes the following choices:

- $\mathcal{K} \leftarrow \{K | (\{\varepsilon\} \subseteq K \subseteq ios(M)) \wedge (|K| = q+1) \wedge \forall 1 \leq i \leq q : \exists r_i t_{i,j} \in K\}$
- $T \leftarrow \{z | (z \in ios(M)) \wedge (ln(z) = 2(m-q)+3)\}$
- $D \leftarrow \{\{z, z'\} | (\{z, z'\} \subseteq pf(T)) \wedge (|end(M, \{z, z'\})| = 2) \wedge (s_0 \notin end(M, \{z, z'\}))\}$
- For each IOS pair $\{z, z'\} \in D$, the selected $\{z, z'\}$-separator is $\{(\varepsilon, a)\}$.
- $T' \leftarrow \{zav_{i,j} | (zav_{i,j} \in ios(M)) \wedge (ln(z) = 2(m-q)+3)\}$

The CTS $T'$ is of the length $(2(m-q)+5)q(4q)^{m-q+1}$. For a comparison, if Step 1 of the Fig. 2 procedure is executed as in the old method, the best choices that one can make are the following:

- $\mathcal{K} \leftarrow \{\{\varepsilon\}\}$
- $T \leftarrow \{z | (z \in ios(M)) \wedge (ln(z) = 2m)\}$
- $D \leftarrow \emptyset$
- $T' \leftarrow T$

Now $len(T')$ is $(2m+1)(4q)^m$. Note that for any fixed $m - 2q$,

$$\lim_{q \to \infty} \frac{(2(m-q)+5)q(4q)^{m-q+1}}{(2m+1)(4q)^m} = 2^{1-2q}q^{2-q},$$

meaning that for any given percentage $p$, there are infinitely many pairs $(q, m)$ for which the savings in the CTS length thanks to the additional freedom for preambles can be at least $p$.

**Example 4.** Suppose that $M$ is the OFSM in Fig. 4(c), $A = br(\emptyset)$ and $m = 2$. If one executes the new method optimally, one makes the following choices:

- $\mathcal{K} \leftarrow \{\{\varepsilon, a0\}\}$
- $T' \leftarrow \{a0b0c0b0c0\}$

For a comparison, if Step 1 of the Fig. 3 procedure is executed as in the old method, the best choices that one can make are the following:

- $\mathcal{K} \leftarrow \{\{\varepsilon, a0, a0b0\}\}$
- $T' \leftarrow \{a0b0c0b0c0b0\}$

Obviously, $a0b0$ should better not be present in the only member of $\mathcal{K}$. Unlike the old method, the new one is able to detect that.

## 5 Discussion and conclusions

The current CTS construction methods similar to ours, hereby called 'the current methods', are those of papers [1]–[3], [5]–[24]. Here it is necessary to note that papers [9], [11], [19], [24] describe an adaptive testing process, in which CTS construction is interspersed with applications of the already constructed part. For each of the papers, what we call 'the method of the paper' is the method that it virtually proposes for test suite completion, where we assume that the method is given an empty test suite.

Unlike our method, the current ones are unable to handle conformance relations other than quasi-equivalence and quasi-reduction or exploit arbitrary upper bounds given on the extent to which individual sets of the specified IOSs diverge in the implementation. What they do have in common with our method is that they consist, at least implicitly, of two phases: decomposition of the initial test goal into a set of simple test goals and satisfaction of each of the simple goals. The initial goal is to test the conformance, while each of the simple goals is either to test the implementation of a given IOS or to test the divergence of a given pair of IOSs.

The way in which the current methods decompose the intial goal is virtually the same as in our method, except that the latter does not restrict preambles to IOSs leading the specification OFSM to a definitely reachable state and has a strategy against harmfully redundant members of the employed preamble sets. With the constraint relaxation and the strategy, our method potentially decomposes the initial goal into a simple goal set that can be satisfied by a shorter test suite.

While our method has a more advanced goal-decomposition procedure, its goal-satisfaction procedure is the simplest possible. Many current methods have a much better one, as they support also indirect goal satisfaction [14]–[16], [18], [21], [23], [24]. In [25], we proposed and proved an advanced goal-satisfaction method that generalizes the goal-satisfaction procedures of all current methods and can be easily combined with our new goal-decomposition procedure. The method is generic, but we are developing for it also an efficient specialization. Another item for further study is how the goal-decomposition phase of the CTS construction could take into account the exact nature of the available separators.

## References


[1] A. Petrenko, N. Yevtushenko, "Conformance tests as checking experiments for partial nondeterministic FSM," in: Proc. FATES 2006 (Lecture Notes in Computer Science, vol. 3997), Springer, Berlin, 2006, pp. 118-133.

[2] A. Petrenko, N. Yevtushenko, "Adaptive testing of nondeterministic systems with FSM," in: Proc. HASE 2014, IEEE CS, 2014, pp. 224-228.

[3] A. Petrenko, N. Yevtushenko, "Testing from partial deterministic FSM specifications," *IEEE T. Comput.*, vol. 54, no. 9, pp. 1154-1165, 2005.

[4] M. Kapus-Kolar, *Three Generalizations and a Better Implementation of the State Counting Method for the Construction of Complete Test Suites for FSM Implementations*, Jožef Stefan Institute technical report #12039, 2016.

[5] T.S. Chow, "Testing software design modeled by finite-state machines," *IEEE T. Software Eng.*, vol. 4, no. 5, pp. 178-187, 1978.

[6] S. Fujiwara, G. von Bochmann, F. Khendek, M. Amalou, "Test selection based on finite state models," *IEEE T. Software Eng.*, vol. 17, no. 6, pp. 591-603, 1991.

[7] G. Luo, G. von Bochmann, A. Petrenko, "Test selection based on communicating nondeterministic finite-state machines using a generalized Wp-method," *IEEE T. Software Eng.*, vol. 20, no. 2, pp. 149-161, 1994.

[8] A. Petrenko, N. Yevtushenko, G. von Bochmann, "Testing deterministic implementations from their nondeterministic specifications," in: Proc. IWTCS'96 (IFIP Advances in Information and Communication Technology, vol. 21), Chapman&Hall, London, 1996, pp. 125-140.

[9] R.M. Hierons, "Adaptive testing of deterministic implementation against a nondeterministic finite state machine," *Comput. J.*, vol. 41, no. 5, pp. 349-355, 1998.

[10] I. Koufareva, A. Petrenko, N. Yevtushenko, "Test generation driven by user-defined fault models," in: Proc. IWTCS'99 (IFIP Advances in Information and Communication Technology, vol. 21), Springer, 1999, pp. 213-233.

[11] R.M. Hierons, "Testing from a nondeterministic finite state machine using adaptive state counting," *IEEE T. Comput.*, vol. 53, no. 10, pp. 1330-1342, 2004.

[12] R. Dorofeeva, K. El-Fakih, N. Yevtushenko, "An improved conformance testing method," in: Proc. FORTE 2005 (Lecture Notes in Computer Science, vol. 3731), Springer, Berlin, 2005, pp. 204-218.

[13] A.L. Bonifácio, A. Vieira Moura, A. da Silva Simão, "A generalized model-based test generation method," in: Proc. SEFM 2008, IEEE CS, 2008, pp. 139-148.

[14] A. Simão, A. Petrenko, "Generating checking sequences for partial reduced finite state machines," in: Proc. TestCom/FATES 2008 (Lecture Notes in Computer Science, vol. 5047), Springer, Berlin, 2008, pp. 153-168.

[15] A. Simão, A. Petrenko, "Checking sequence generation using state distinguishing subsequences," in: Proc. ICSTW 2009, IEEE CS, 2009, pp. 48-56.

[16] M.E. Dinçtürk, *A Two Phase Approach for Checking Sequence Generation*, M.Sc. Thesis, SabancıUniversity, August 2009.

[17] L.L. Chaves Pedrosa, A. Vieira Moura, "Generalized partial test case generation method," in: Proc. SSIRI-C 2010, IEEE CS, 2010, pp. 70-77.
[18] A. Simão, A. Petrenko, "Fault coverage-driven incremental test generation," *Comput. J.*, vol. 53, no. 9, pp. 1508-1522, 2010.
[19] A. Petrenko, N. Yevtushenko, "Adaptive testing of deterministic implementations specified by nondeterministic FSMs," in: Proc. ICTSS 2011 (Lecture Notes in Computer Science, vol. 7019), Springer, Berlin, 2011, pp. 162-178.
[20] A.L. Bonifácio, A. Vieira Moura, A. Simão, "Model partitions and compact test case suites," *Int. J. Found. Comput. S.*, vol. 23, no. 1, pp. 147-172, 2012.
[21] A. Simão, A. Petrenko, N. Yevtushenko, "On reducing test length for FSMs with extra states," *Softw. Test. Verif. Rel.*, vol. 22, no. 6, pp. 435-454, 2012.
[22] L.L. Chaves Pedrosa, A. Vieira Moura, "Incremental testing of finite state machines," *Softw. Test. Verif. Rel.*, vol. 23, no. 8, pp. 585-612, 2012.
[23] A. Petrenko, A. Simão, N. Yevtushenko, "Generating checking sequences for nondeterministic finite state machines," in: Proc. ICST 2012, IEEE CS, 2012, pp. 310-319.
[24] A. Petrenko, A. Simão, "Generalizing the DS-methods for testing non-deterministic FSMs," *Comput. J.*, vol. 58, no. 7, pp. 1656-1672, 2015.
[25] M. Kapus-Kolar, *Incremental Construction of Complete Test Suites for Finite State Machine Implementations - Principles Behind the Current Methods and Beyond*, Jožef Stefan Institute technical report #11855, 2015.

**Monika Kapus-Kolar** received her B.Sc. degree in electrical engineering from the University of Maribor, Slovenia, and her M.Sc. and Ph.D. degrees in computer science from the University of Ljubljana, Slovenia. Since 1981 she has been with the Jožef Stefan Institute, Ljubljana, where she is currently a researcher at the Department of Communication Systems. Her current research interests include formal specification techniques and methods for the development of real-time, concurrent and reactive systems.