On "Exploring alternatives for transition verification"

Monika Kapus-Kolar

Jožef Stefan Institute, Department of Communication Systems, Jamova 39, SI-1111 Ljubljana, Slovenia

Abstract

Duan and Chen (doi:10.1016/j.jss.2009.05.019) proposed two methods for constructing an input/output sequence (IOS) whose execution on an implementation N of a given deterministic finite state machine (DFSM) M tests that N can be interpreted as a DFSM properly implementing every individual state and transition of M. This paper shows that the methods and three earlier similar methods potentially introduce cyclic dependencies between the essential segments of the produced IOS, meaning that the IOS might fail to be a complete test under the default interpretation. It then proposes modifications provably preventing such cycles. All the methods assume that M is completely specified and strongly connected and possesses a distinguishing set and that N has at most as many states as M.

Keywords: Conformance testing, Deterministic finite state machine, Checking sequence, Distinguishing set.

1. Introduction

Duan and Chen [3] proposed two methods for constructing an IOS whose execution on an implementation N of a given DFSM M tests that N can be interpreted as a DFSM properly implementing every individual state and transition of M. The first method is to use the method of Ural & al. [9] or one of its subsequent derivatives [1, 4, 5, 8] with the additional possibility to employ in transition implementation tests (TITs) also the so-called γ-sequences. The second method is the method of Ural and Zhang [10] enhanced with the possibility of employing γ-sequences in TITs.

This paper shows that the methods and three earlier similar methods [1, 8, 11] potentially introduce cyclic dependencies between the essential segments of the produced IOS, meaning that the IOS might fail to be a complete test under the default interpretation. It then proposes modifications provably preventing such cycles. All the methods assume that M is completely specified and strongly connected and possesses a distinguishing set (DS) and that N has at most as many states as M.

The paper is organized as follows. A practitioner interested exclusively in more proper *use* of a specific method can go immediately to the pertinent subsection of Section 5, in which the proposed modifications are summarized as straightforward instructions in the language of the source paper of the method. Sections 2 to 4 are intended for readers seeking *deeper understanding* of the methods and their problems. Section 2 introduces the basic notation and definitions. Section 3 gives an outline of the five methods and of the dependencies which they introduce. It is actually a description and proof of an abstract method of which the considered methods are supposed to be specialisations. In Section 4, we describe the weak points of the methods and the proposed modifications in detail. Section 6 concludes the paper.

2. Basic notation and definitions

Let ϵ denote an empty sequence. For a sequence $\sigma = o_1 \dots o_k$, let $|\sigma|$ denote its length k, $seg(\sigma)$ the set of all its segments $o_i \dots o_j$ with $1 \leq i \leq j \leq k$, $set(\sigma)$ the set $\{o_1, \dots, o_k\}$ and for every $0 \leq i \leq |\sigma|$, $pf(\sigma, i)$ its prefix $o_1 \dots o_i$. For two sequences $\sigma = o_1 \dots o_k$ and $\sigma' = o'_1 \dots o'_{k'}$, let $\sigma \cdot \sigma'$ denote the sequence $o_1 \dots o_k o'_1 \dots o'_{k'}$, whereas $\sigma \leq \sigma'$ denotes that σ is a prefix of σ'. For a sequence set Σ, let $seg(\Sigma)$ denote $\cup_{\sigma \in \Sigma} seg(\sigma)$.

DFSMs are machines exposed to inputs from their environment. An input is a member of the assumed universal input alphabet. A DFSM Q has a finite set $st(Q)$ of states in which it might reside, among them its initial state $init(Q)$. It also has a finite set $tr(Q)$ of transitions which it might execute. To execute a transition from a

Email address: monika.kapus-kolar@ijs.si (Monika Kapus-Kolar)

specific state s_i means to accept an input x and consequently issue the corresponding output y and enter the corresponding next state s_j. We denote such a transition t as $(s_i, s_j, x/y)$. If Q has no transition $(s_k, s_j, x/y)$ with $s_k \neq s_i$, we say that t is invertible.

An IOS is a sequence $x_1/y_1 \ldots x_k/y_k$ of k input/output pairs. For such a z, let $in(z)$ denote the input sequence $x_1 \ldots x_k$.

A transition sequence (TS) of a DFSM Q is a sequence $(s_1, s_2, x_1/y_1) \ldots (s_k, s_{k+1}, x_k/y_k)$ of transitions in $tr(Q)$. For such a τ, let $init(\tau)$ denote its initial state s_1, $fin(\tau)$ it final state s_{k+1} and $lab(\tau)$ its label $x_1/y_1 \ldots x_k/y_k$. An IOS z is a member of the language $lan(s_i)$ of a state s_i of a DFSM Q if Q has a TS τ with $init(\tau) = s_i$ and $lab(\tau) = z$. For a DFSM Q, let $lan(Q)$ denote $lan(init(Q))$ and $ios(Q)$ the IOS set $\cup_{s_i \in st(Q)} lan(s_i)$.

If a DFSM Q for every state s_i in $st(Q)$ and input x possesses a transition t with $init(t) = s_i$ and $in(lab(t)) = x$, it is completely specified. If it for every two different states s_i and s_j in $st(Q)$ possesses a TS τ with $(init(\tau) = s_i) \wedge (fin(\tau) = s_j)$, it is strongly connected.

An IOS z is a unique IOS (UIO) of (a specific state s_i of) a DFSM Q if s_i is the only state s in $st(Q)$ for which Q has a TS τ with $(init(\tau) = s) \wedge (lab(\tau) = z)$. An IOS z is a backward UIO (BUIO) of (a specific state s_i of) a DFSM Q if s_i is the only state s in $st(Q)$ for which Q has a TS τ with $(fin(\tau) = s) \wedge (lab(\tau) = z)$. For a DFSM Q, let $uio(Q)$ denote its UIO set. For a z in $uio(M)$, let $ts(z)$ denote the only TS τ of M with $lab(\tau) = z$.

A DS of M is a set for every state s_i in $st(M)$ comprising such a UIO z_i that for every other state s_j in $st(M)$, there is a k with $(in(pf(z_i, k)) = in(pf(z_j, k))) \wedge (pf(z_i, k) \neq pf(z_j, k))$.

A digraph G consists of a set $vr(G)$ of vertices and a set $ed(G)$ of edges. Every edge in $ed(G)$ is a (v, v', l) with v its initial vertex in $vr(G)$, v' its final vertex in $vr(G)$ and l its label. A walk of a digraph G is a sequence $(v_1, v_2, l_1) \ldots (v_k, v_{k+1}, l_k)$ of edges in $ed(G)$. For such a w, let $init(w)$ denote its initial vertex v_1, $fin(w)$ its final vertex v_{k+1} and $lab(w)$ its label. If for every $1 \leq i \leq k$, l_i is a sequence, $lab(w)$ is $l_1 \cdot \ldots \cdot l_k$. For a digraph G, let $wk(G)$ denote the set of all its walks.

A test is any IOS in $ios(M)$ supposed to be observed on N. A test z is complete, i.e., $in(z)$ is a checking sequence, if $z \in lan(N)$ implies $lan(N) = lan(M)$.

3. An outline of the methods

The methods start by choosing a DS of M, call it $\mathcal{D}$. For a state s_i in $st(M)$, let d_i denote the UIO in $\mathcal{D} \cap lan(s_i)$, in the case of $s_i = init(M)$ also called d_{init}.

A state s_i in $st(M)$ is said to be properly implemented in N, denoted as $ok(s_i)$, if $d_i \in uio(N)$. In that case, let s'_i denote the only state s in $st(N)$ with $d_i \in lan(s)$. A transition $t = (s_i, s_j, x/y)$ in $tr(M)$ is said to be properly implemented in N, denoted as $ok(t)$, if $ok(s_i) \wedge ok(s_j) \wedge ((s'_i, s'_j, x/y) \in tr(N))$. M is said to be properly implemented by N, denoted as $ok(M)$, if $(d_{init} \in lan(N)) \wedge (\forall s_i \in st(M) : ok(s_i)) \wedge (\forall t \in tr(M) : ok(t))$. The generated test, call it z^*, is supposed to satisfy $(z^* \in lan(N)) \Rightarrow ok(M)$, where $ok(M)$ by $|st(N)| \leq |st(M)|$ implies $lan(N) = lan(M)$.

To make z^* complete, the methods secure $d_{init} \leq z^*$ and that $seg(z^*)$ comprises a sufficient set of special-purpose *subtests*. Among the subtests, some are selected for inclusion in advance, call the set of all such tests $\mathcal{A}$, and some *during* the construction of z^*, from a pool, call it C, of candidate tests.

The primary purpose of the tests in $\mathcal{A}$ is to check $ok(s_i)$ for every state s_i in $st(M)$. For that purpose, the methods secure $\mathcal{D} \subseteq seg(\mathcal{A})$. On the other hand, the purpose of each individual test in C is to check $ok(t)$ for a specific transition t in $tr(M)$. Let Θ denote the set of those transitions in $tr(M)$ for which there are implementation tests (ITs) specified in C, with Θ' denoting the set $tr(M) \setminus \Theta$. For the transitions in Θ', the set of the specified candidate ITs is a subset C' of $seg(\mathcal{A})$. The methods secure that every transition t in $tr(M)$ has an IT from $C \cup C'$ in $seg(z^*)$.

Every conceived specification of a candidate IT z for a transition t in $tr(M)$ is virtually a *triplet* (z, t, θ) with θ the set of those transitions t' in $tr(M)$ on whose $ok(t')$ the soundness of the TIT *presumably* depends. In most cases, there are at least some t' in θ for which the soundness *actually* depends on $ok(t')$. Every such triplet ξ, hence, introduces on $tr(M)$ the dependency relation $\{(t, t') | t' \in \theta\}$, call it $\mathcal{R}_\xi$. Let Ξ denote the set of the conceived triplets and $\mathcal{R}$ the *cumulative dependency relation* $\cup_{\xi \in \Xi} \mathcal{R}_\xi$.

The methods secure that every triplet (z, t, θ) in Ξ satisfies $(((\mathcal{A} \cup \{z\}) \subseteq ios(N)) \wedge \forall t' \in \theta : ok(t')) \Rightarrow ok(t)$. Their proof for $(z^* \in lan(N)) \Rightarrow ok(M)$ is then virtually as follows:

Proof. For every transition t in $tr(M)$, choose in Ξ any triplet (z_t, t, θ_t) with $z_t \in seg(z^*)$. Order the transitions in $tr(M)$ into a sequence $t_1 \ldots t_{|tr(M)|}$ for every $1 \leq i \leq |tr(M)|$ satisfying $\theta_{t_i} \subseteq \{t_1, \ldots, t_{i-1}\}$.

Suppose that $z^* \in lan(N)$, which implies $(\mathcal{A} \cup \{z_t | t \in tr(M)\}) \subseteq ios(N)$ and, by $d_{init} \leq z^*$, $d_{init} \in lan(N)$. By $(\mathcal{A} \subseteq ios(N)) \wedge (\mathcal{D} \subseteq seg(\mathcal{A})) \wedge (|st(N)| \leq |st(M)|)$, $ok(s_i)$ for every s_i in $st(M)$.

If for an $1 \leq i \leq |tr(M)|$, $ok(t_j)$ for every $1 \leq j < i$, $((\mathcal{A} \cup \{z_{t_i}\}) \subseteq ios(N)) \wedge \forall 1 \leq j < i : ok(t_j)$ by $\theta_{t_i} \subseteq \{t_1, \ldots, t_{i-1}\}$ implies $((\mathcal{A} \cup \{z_{t_i}\}) \subseteq ios(N)) \wedge \forall t \in \theta_{t_i} : ok(t)$ and, hence, $ok(t_i)$. For every $1 \leq i \leq |tr(M)|$, $ok(t_i)$, hence, follows by induction, implying $ok(t)$ for every t in $tr(M)$. □

The proof has a weak point. It can happen that an unfortunate choice of triplets in its first step prevents completion of its second step, because of a cycle in $\mathcal{R}$. If $\mathcal{R}$ is acyclic, however, the proof is sound. The methods virtually assume that $\mathcal{R}$ is acyclic, but in the next section, we for each of them give an example where this is not true, meaning that they cannot be trusted in the general case. As a remedy, we then propose modifications provably making $\mathcal{R}$ acyclic.

4. The weak points of the methods and suggestions for modification

4.1. Introduction

Section 4 is organized as follows. In Section 4.3, we describe the TIT types which the methods virtually use, after in Section 4.2 describing the UIOs and BUIOs which they use for state recognition within TITs. Section 4.4 explains how to read the digraph which the methods allowing $\Theta' \neq \emptyset$ construct as an encoding of the TITs conceived for the members of Θ'. Section 4.5 explains how to read the digraph which the considered methods construct as an encoding of the TITs conceived for the members of Θ. Section 4.6 shows that each of the methods sometimes specifies a cycle in $\mathcal{R}$. The modifications which we propose as a remedy are described and proven sufficient in Section 4.7.

4.2. The employed kinds of UIOs and BUIOs

In the source papers of the methods, the members of $\mathcal{D}$ and $\mathcal{D}$ itself are not given any special names, whereas the members of $\mathcal{A}$ are called α-sequences, α'-sequences or α-elements. The methods construct members of $\mathcal{A}$ as concatenations of, as they call them, T-sequences - selected members of $uio(M)$ owning a member of $\mathcal{D}$ as a prefix. Call the set of the employed T-sequences $\mathcal{T}$.

Recall that $\mathcal{A} \subseteq ios(N)$ implies that every d_i in $\mathcal{D}$ is in N a UIO of s'_i. Besides, the methods secure that for every IOS z in $\mathcal{T}$, $seg(\mathcal{A})$ comprises a $z \cdot z'$ with $z' \in \mathcal{D}$, and that every IOS in $\mathcal{A}$ has a member of $\mathcal{D}$ as a prefix and a member of $\mathcal{T}$ as a suffix. Consequently, the IOSs in $\mathcal{D} \cup \mathcal{T} \cup \mathcal{A}$ are within TITs useful both for forward and for backward state recognition.

Some of the methods [1, 3, 8] use for state recognition within TITs also members z of $uio(M)$ conceived as a γ-sequence, i.e., as a $z' \cdot z''$ with $z'' \in \mathcal{D}$ and z', call it $ip(z)$ - the invertible prefix of z, the label of a non-empty TS of M consisting exclusively of invertible transitions. Let Γ denote the set of the γ-sequences conceived for application within ITs for transitions in Θ. Let Γ' denote the set of the γ-sequences z with $ip(z)$ a true suffix of a member of $\mathcal{T}$.

Let $\mathcal{U}$ denote the superset of Γ comprising also every other auxiliary UIO conceived for application within ITs for transitions in Θ (in the only one among the methods which allows arbitrary UIOs for the purpose [11], the set of their *input* sequences is called $\mathcal{U}$). Let $\mathcal{Z}$ denote the set $\mathcal{D} \cup \mathcal{T} \cup \mathcal{A} \cup \mathcal{U} \cup \Gamma'$.

To trust that a UIO z of a state s_i of M is in N a UIO of s'_i, the methods require $ok(t)$ for the transitions t in a subset of $tr(M)$ which we call $req(z)$, with $req(\mathcal{U})$ denoting $\cup_{z \in \mathcal{U}} req(z)$. For a z in $\mathcal{Z}$, they virtually compute $req(z)$ as follows:

1. If $z \in (\mathcal{D} \cup \mathcal{T} \cup \mathcal{A})$, $req(z) = \emptyset$.
2. If $z \in (\Gamma \cup \Gamma')$, $req(z)$ comprises every transition $(s_i, s_j, x/y)$ in $tr(M)$ with $x \in set(in(ip(z)))$.
3. If $z \in (\mathcal{U} \setminus \Gamma)$, $req(z)$ comprises every transition $(s_i, s_j, x/y)$ in $tr(M)$ with $x \in set(in(z))$.

To trust that a BUIO z of a state s_i of M is in N a BUIO of s'_i, the methods require $ok(t)$ for the transitions t in a subset of $tr(M)$ which we call $breq(z)$. For a z in $\mathcal{Z}$, they virtually compute $breq(z)$ as follows:

1. If $z \in (\mathcal{T} \cup \mathcal{A})$, $breq(z) = \emptyset$.
2. If $z \in (\mathcal{D} \setminus \mathcal{T})$, $breq(z) = set(ts(z))$.
3. If z is an $ip(z) \cdot z'$ in $\Gamma \cup \Gamma'$, $breq(z) = breq(z')$.
4. If $z \in (\mathcal{U} \setminus \Gamma)$, $breq(z) = req(z)$.

4.3. The employed kinds of TITs

In the methods, every IT for a transition t in $tr(M)$ is conceived as a $tit1(\tau, t, \tau')$, as we call the first TIT type, or as a $tit2(\tau, \tau', t, \tau'')$, as we call the second TIT type.

For a $tit1(\tau, t, \tau')$, one assumes $(\tau \cdot t \cdot \tau' \in wk(M)) \wedge (lab(\tau') \in \mathcal{Z}) \wedge \exists d_i \in \mathcal{D} : (d_i \leq lab(\tau \cdot t \cdot \tau')) \wedge \forall t' \in (set(\tau) \cup req(lab(\tau'))) : ok(t')$, so that $tit1(\tau, t, \tau')$ denotes a TIT whose specification is the triplet $(lab(\tau \cdot t \cdot \tau'), t, set(\tau) \cup req(lab(\tau')))$.

For a $tit2(\tau, \tau', t, \tau'')$, one assumes $(\tau \cdot \tau' \cdot t \cdot \tau'' \in wk(M)) \wedge (\{lab(\tau), lab(\tau'')\} \subseteq \mathcal{Z}) \wedge \forall t' \in (breq(lab(\tau)) \cup set(\tau') \cup req(lab(\tau''))) : ok(t')$, so that $tit2(\tau, \tau', t, \tau'')$ denotes a TIT whose specification is the triplet $(lab(\tau \cdot \tau' \cdot t \cdot \tau''), t, breq(lab(\tau)) \cup set(\tau') \cup req(lab(\tau'')))$.

4.4. The candidate TITs specified for the members of Θ'

Those of the methods which allow $\Theta' \neq \emptyset$ virtually construct C' in parallel with Θ' and encode them in a digraph G'_S, the final version of a digraph G_S [1, 8]. Every vertex v in $vr(G'_S)$ represents a transition, call it t_v, in Θ' and virtually contributes to C' every $tit1(\tau, t_v, \tau' \cdot \tau'')$ which satisfies $(lab(\tau \cdot t_v \cdot \tau') \in \mathcal{T}) \wedge (lab(\tau'') \in \mathcal{D})$ and, depending on the method, $\tau' = \epsilon$ or $lab(\tau' \cdot \tau'') \in \Gamma'$.

4.5. The candidate TITs specified for the members of Θ

For transitions in Θ, the methods represent candidate ITs as specific walks in a separate digraph, which we call $\mathcal{G}$. Of the five methods, only one, the second of Duan and Chen [3], explicitly pursues subtest overlapping. In it, $\mathcal{G}$ is called G^* and has a slightly different structure [3, 10] than in the remaining methods, which call the digraph G'. Hence in the following, G^* and G', respectively, denote $\mathcal{G}$ of the first and of the second form.

$\mathcal{G}$ is the digraph in which the methods choose a walk, call it w^*, and interpret it as a TS, call it τ^*, of M. For G^*, z^* is computed as $lab(\tau^*)$. For G', $fin(\tau^*)$ is always $init(M)$ and z^* is computed as $lab(\tau^*) \cdot d_{init}$.

The specialty of G^* are edges whose label is an input sequence preceded by a minus sign, call them negative edges. Let $ed^+(\mathcal{G})$ denote the set of the non-negative edges in $ed(\mathcal{G})$. Every walk w of $\mathcal{G}$ consisting exclusively of the edges in $ed^+(\mathcal{G})$ is a representation of a specific TS of M, call it $ts(w)$. In the case of $ts(w) = \epsilon$, $init(ts(w))$ is a state in $st(M)$ known from the context and $fin(ts(w)) = init(ts(w))$. If for an e in $ed^+(\mathcal{G})$, $ts(e)$ comprises just one element, call the transition $t(e)$. To simplify the discussion, we in the following pretend that in G^*, $lab(e)$ of every e in $ed^+(\mathcal{G})$ is, as in G', $lab(ts(e))$, although it is actually $in(lab(ts(e)))$.

$ed^+(\mathcal{G})$ is the union of the following disjoint sets, for E the edge set $\{(v_i, v_j, x/y) | (s_i, s_j, x/y) \in tr(M)\}$:

1. The set of the edges representing individual transitions in Θ and supposed to be present in $set(w^*)$. We call it E'_C (alternatively called E_C if $\Theta' = \emptyset$), as it is originally called in G'. In G^*, it is the set of those edges which end in a vertex $v^*_{e,2}$ with $e \in E$.
2. The set of the edges e representing (as $lab(e)$) individual IOSs in $\mathcal{A}$. We call it $E_{\alpha'}$, as it is originally called in G'. In G^*, it is the set of those edges which end in a vertex $v^*_{\rho,2}$ with $\rho \notin E$.
3. The set of the edges e representing (as $lab(e)$) individual IOSs in $\mathcal{T}$. We call it E_T, as it is originally called in G'. In G^*, it is the set of those edges which end in a $v^*_{e,\beta}$ vertex.
4. The set of the edges e representing (as $lab(e)$) individual IOSs in $\mathcal{U}$. We call it E_U (alternatively called E_Γ if $\mathcal{U} = \Gamma$), as it is originally called in G'. In G^*, it is the set of those edges which end in a vertex $v^*_{e,z}$ with $z \notin \{1, 2, \beta\}$. Let Θ^U denote the transition set $\{t(e) | (e \in E'_C) \wedge \exists e' \in E_U : (fin(e) = init(e'))\}$.
5. The set, call it $E^\dagger$, of the remaining edges in $ed^+(\mathcal{G})$ representing individual transitions. In G', $E^\dagger$ is the union of the sets originally called $E_C \setminus E'_C$ and E''. In G^*, $E^\dagger = E$.
6. The set, call it E_ϵ, of the edges e with $lab(e) = \epsilon$. For G^*, E_ϵ is not clear from the source paper of the method [3], but by analogy with the basic method [10], it is virtually as described in Section 5.4.

As originally in G', let E'' denote the set of those edges in $E^\dagger$ which potentially participate in a walk specifying a candidate TIT. For G^*, it is fair to say that E'' is $E^\dagger \cap set(w^*)$, because in the method which uses G^*, the choice of $E^\dagger \cap set(w^*)$ is not arbitrary. Let Θ'' denote the set $\{t(e) | e \in E''\}$.

G' virtually specifies the following candidate TITs:

1. Every walk w of G' which is of the form $e_1 \cdot w' \cdot e_2 \cdot w'' \cdot e_3$ with $(\{e_1, e_3\} \subseteq (E_{\alpha'} \cup E_T \cup E_U)) \wedge (set(w') \subseteq E'') \wedge (e_2 \in E'_C) \wedge (set(w'') \subseteq E_\epsilon)$ specifies the candidate TIT $tit2(ts(e_1), ts(w'), t(e_2), ts(e_3))$ and, hence, contributes to Ξ the triplet $(lab(w), t(e_2), breq(lab(e_1)) \cup set(ts(w')) \cup req(lab(e_3)))$.
2. Every walk w of G' which is of the form $e_1 \cdot w' \cdot e_2$ with $(e_1 \in (E_{\alpha'} \cup E_T \cup E_U)) \wedge (set(w') \subseteq E'') \wedge (e_2 \in E'_C) \wedge (fin(t(e_2)) = init(M))$ specifies the candidate TIT $tit2(ts(e_1), ts(w'), t(e_2), ts(d_{init}))$ and, hence, contributes to Ξ the triplet $(lab(w) \cdot d_{init}, t(e_2), breq(lab(e_1)) \cup set(ts(w')))$.

The method using G^* assumes $(\mathcal{T} = \mathcal{D}) \wedge (\mathcal{U} = \Gamma)$, implying $breq(z) = \emptyset$ for every z in $\mathcal{U}$. Hence, G^* virtually specifies the following candidate TITs:

1. Every walk w of G^* which is of the form $e_1 \cdot w' \cdot e_2 e_3$ with $(\{e_1, e_3\} \subseteq (E_{\alpha'} \cup E_T \cup E_U)) \wedge (set(w') \subseteq (E'' \cup E_\epsilon)) \wedge (e_2 \in E'_C) \wedge \forall d_i \in \mathcal{D} : (d_i \not\leq lab(e_2 e_3))$ specifies the candidate TIT $tit2(ts(e_1), ts(w'), t(e_2), ts(e_3))$ and, hence, contributes to Ξ the triplet $(lab(w), t(e_2), set(ts(w')) \cup req(lab(e_3)))$.
2. Every walk w of G^* which is of the form $e_1 e_2$ with $(e_1 \in E'_C) \wedge (e_2 \in (E_T \cup E_U)) \wedge \exists d_i \in \mathcal{D} : (d_i \leq lab(e_1 e_2))$ specifies the candidate TIT $tit1(\epsilon, t(e_1), ts(e_2))$ and, hence, contributes to Ξ the triplet $(lab(w), t(e_1), req(lab(e_2)))$.

4.6. Examples of cycles in $\mathcal{R}$

Example 1. In Section 3.3 of the paper of Chen & al. [1], the transition $t = (s_1, s_2, a/0)$ of M_0 is in Θ', with $ts(d_1) = tt't$ for t' the transition $(s_2, s_1, b/1)$. The only TIT considered for t is $tit1(tt', t, ts(d_2))$, contributing the triplet $(lab(ts(d_1) \cdot ts(d_2)), t, \{t, t'\})$, implying $(t, t) \in \mathcal{R}$. $\mathcal{R}$ fails to be acyclic because the method fails to include into the initial version of G_S an edge specifying the self-dependency of t. The problem is present also in the method of Tekle & al. [8] and in the first method of Duan and Chen [3].

Example 2. In the paper of Chen & al. [1], the G' in Fig. 3 comprises the walk $w = (v_5, v'_1, \bar{T}_5)(v'_1, v'_2, a/2)(v'_2, v_1, b/1)(v_1, v'_2, \bar{T}_1)$ with $((v_5, v'_1, \bar{T}_5) \in E_T) \wedge ((v'_1, v'_2, a/2) \in E'') \wedge ((v'_2, v_1, b/1) \in E'_C) \wedge ((v_1, v'_2, \bar{T}_1) \in E_T)$, with $t((v'_1, v'_2, a/2))$ the transition $t = (s_1, s_2, a/2)$ and with $t((v'_2, v_1, b/1))$ the transition $t' = (s_2, s_1, b/1)$. The walk specifies the triplet $(lab(w), t', \{t\})$, implying $(t', t) \in \mathcal{R}$.

On the other hand, t is in Θ', with the only candidate IT the one from Example 1, implying $(t, t') \in \mathcal{R}$. $\mathcal{R}$ fails to be acyclic because a specific transition in Θ'', namely t, is a member of Θ'. The problem is present also in the method of Tekle & al. [8] and in the first method of Duan and Chen [3].

Example 3. In the paper of Yalcin and Yenigün [11], the G' in Fig. 5 comprises the walk $w = (v_3, v'_2, \bar{T}_3)(v'_2, v'_3, a/0)(v'_3, v_2, b/1)(v_2, v'_1, \bar{T}_2)$ with $((v_3, v'_2, \bar{T}_3) \in E_T) \wedge ((v'_2, v'_3, a/0) \in E'') \wedge ((v'_3, v_2, b/1) \in E'_C) \wedge ((v_2, v'_1, \bar{T}_2) \in E_T)$, with $t((v'_3, v_2, b/1))$ the transition $t = (s_3, s_2, b/1)$ and with $t((v'_2, v'_3, a/0))$ the transition $t' = (s_2, s_3, a/0)$. The walk specifies the triplet $(lab(w), t, \{t'\})$, implying $(t, t') \in \mathcal{R}$.

On the other hand, the G' comprises the walk $w' = (v_3, v'_2, \bar{T}_3)(v'_2, v^U_3, a/0)(v^U_3, v'_2, b/1)$ with $((v_3, v'_2, \bar{T}_3) \in E_T) \wedge ((v'_2, v^U_3, a/0) \in E'_C) \wedge ((v^U_3, v'_2, b/1) \in E_U)$, with $t((v'_2, v^U_3, a/0)) = t'$ and with $t \in req(b/1)$. The walk specifies a triplet $(lab(w'), t', \{t, \ldots\})$, implying $(t', t) \in \mathcal{R}$. $\mathcal{R}$ fails to be acyclic because a specific transition in Θ'', namely t', is a member of Θ^U. The problem is present also in the first method of Duan and Chen [3].

Example 4. In the paper of Yalcin and Yenigün [11], the G' in Fig. 5 comprises the walk $w = (v^U_3, v'_2, b/1)(v'_2, v_3, b/0)(v_3, v'_2, \bar{T}_3)$ with $((v^U_3, v'_2, b/1) \in E_U) \wedge ((v'_2, v_3, b/0) \in E'_C) \wedge ((v_3, v'_2, \bar{T}_3) \in E_T)$, with $t((v'_2, v_3, b/0))$ the transition $t = (s_2, s_3, b/0)$ in $breq(b/1)$. The walk specifies a triplet $(lab(w), t, \{t, \ldots\})$, implying $(t, t) \in \mathcal{R}$. $\mathcal{R}$ fails to be acyclic because for a specific edge e in E^U with $breq(lab(e)) \neq \emptyset$, $fin(e) \in \{init(e') | e' \in (E'_C \cup E'')\}$. The problem is present also in the first method of Duan and Chen [3].

Example 5. If G^* is the digraph which Duan and Chen [3] partially present in Fig. 9 of their paper, $wk(G^*)$ comprises the walk $w = (v^*_{\rho_0,1}, v^*_{\rho_0,2}, d_0 \cdot d_1)(v^*_{\rho_0,2}, v_1, \epsilon)(v_1, v_2, a/1)(v_2, v^*_{e_7,1}, \epsilon)(v^*_{e_7,1}, v^*_{e_7,2}, b/0)(v^*_{e_7,2}, v^*_{e_7,\beta}, d_0)$ of G^*, with $((v^*_{\rho_0,1}, v^*_{\rho_0,2}, d_0 \cdot d_1) \in E_{\alpha'}) \wedge ((v^*_{\rho_0,2}, v_1, \epsilon) \in E_\epsilon) \wedge ((v_1, v_2, a/1) \in E) \wedge ((v_2, v^*_{e_7,1}, \epsilon) \in E_\epsilon) \wedge ((v^*_{e_7,1}, v^*_{e_7,2}, b/0) \in E'_C) \wedge ((v^*_{e_7,2}, v^*_{e_7,\beta}, d_0) \in E_T)$, with $t((v_1, v_2, a/1))$ the transition $t = (s_1, s_2, a/1)$ and with $t((v^*_{e_7,1}, v^*_{e_7,2}, b/0))$ the transition $t' = (s_2, s_0, b/0)$. If the walk is a segment of w^*, so that $(v_1, v_2, a/1) \in E''$, it specifies the triplet $(lab(w), t', \{t\})$, implying $(t', t) \in \mathcal{R}$.

On the other hand, the G^* comprises the walk $w' = (v^*_{e_3,1}, v^*_{e_3,2}, a/1)(v^*_{e_3,2}, v^*_{e_3,\gamma}, b/0 \cdot d_0)$, with $((v^*_{e_3,1}, v^*_{e_3,2}, a/1) \in E'_C) \wedge ((v^*_{e_3,2}, v^*_{e_3,\gamma}, b/0 \cdot d_0) \in E_U)$, with $t((v^*_{e_3,1}, v^*_{e_3,2}, a/1)) = t$ and with $t' \in req(b/0 \cdot d_0)$. The walk specifies a triplet $(lab(w'), t, \{t', \ldots\})$, implying $(t, t') \in \mathcal{R}$. $\mathcal{R}$ fails to be acyclic because a specific transition in Θ'', namely t, is a member of Θ^U.

4.7. Proposed modifications

Modification 1. If not assuming $\Theta' = \emptyset$, modify the G'_S-construction procedure as follows (recall Example 1), with Ξ' denoting the set of all the triplets specifying a $tit1(\tau, t, \tau' \cdot \tau'')$ with $(lab(\tau \cdot t \cdot \tau') \in \mathcal{T}) \wedge (lab(\tau'') \in \mathcal{D}) \wedge ((\tau' = \epsilon) \vee (lab(\tau' \cdot \tau'') \in \Gamma'))$:

1. Initialize $vr(G_S)$ to $\{v_{t,\theta} | \exists (z, t, \theta) \in \Xi'\}$ and $ed(G_S)$ to $\{(v_{t,\theta}, v_{t',\theta'}, \epsilon) | (\{v_{t,\theta}, v_{t',\theta'}\} \subseteq vr(G_S)) \wedge (t \in \theta')\}$.
2. When removing vertices of G_S to change it into an acyclic G'_S, the set to be maximized and adopted as Θ' is $\{t | \exists v_{t,\theta} \in vr(G'_S)\}$. The set of the triplets which G'_S contributes to Ξ is then $\{(z, t, \theta) | ((z, t, \theta) \in \Xi') \wedge (v_{t,\theta} \in vr(G'_S))\}$.

Modification 2. Secure $(\Theta'' \cup req(\mathcal{U})) \cap (\Theta' \cup \Theta^U) = \emptyset$ (recall the Examples 2, 3 and 5), but, unlike the first method of Duan and Chen [3], allow $\Theta'' \cap (\cup_{z \in \Gamma} set(ts(ip(z)))) \neq \emptyset$.

Modification 3. In G', for every edge e in E_U with $breq(lab(e)) \neq \emptyset$, with $fin(e)$ originally a v'_i, change $fin(e)$ into v_i, a vertex not in $\{init(e') | e' \in (E'_C \cup E'')\}$ (recall Example 4).

Theorem 1. The modifications secure $(z^* \in lan(N)) \Rightarrow ok(M)$.

Proof. With Mod. 3, every (z, t, θ) in Ξ satisfies $(t \in \Theta') \vee (\theta \subseteq (\Theta'' \cup req(\mathcal{U})))$. Hence, by $(\Theta'' \cup req(\mathcal{U})) \cap \Theta' = \emptyset$, $\mathcal{R}$ comprises no (t, t') with $(t \in \Theta) \wedge (t' \in \Theta')$,

implying that any cycle involving a member of Θ' is one comprising only members of Θ', but with Mod. 1, there is no such cycle.

Any cycle is, hence, in $\{(t,t')|((t,t') \in \mathcal{R}) \wedge (\{t,t'\} \subseteq \Theta)\}$. By $(\Theta'' \cup req(\mathcal{U})) \cap \Theta^U = \emptyset$, the relation comprises no (t,t') with $t' \in \Theta^U$, implying that any cycle is in $\{(t, t')|((t,t') \in \mathcal{R}) \wedge (\{t,t'\} \subseteq (\Theta \setminus \Theta^U))\}$. With Mod. 3 and every (z,t,θ) in Ξ with $t \in (\Theta \setminus \Theta^U)$ satisfying $\theta \subseteq \Theta''$, this relation is, however, acyclic, because the methods secure acyclicity of its superrelation $\{(t(e'),t(e))|\exists e \cdot w \cdot e' \in wk(\mathcal{G}) : ((e \in E'') \wedge (set(w) \subseteq (E'' \cup E_\epsilon)) \wedge (e' \in E'_C))\}$. Hence, the entire $\mathcal{R}$ is acyclic. The rest of the proof is the proof from Section 3. □

5. The modifications summarized by method

For each of the methods, the modifications are described in the language of its source paper.

5.1. The method of Chen & al. [1] or Tekle & al. [8]

For a path $\bar{P}$ in G, let $dep(\bar{P})$ denote the set of those edges in E whose input symbol is in the input portion of $label(\bar{P})$. Modify the computation of L as follows:

1. Initially, let V_S comprise every $v_{e_l,\eta}$ for which there is an $i \in \{1,\ldots,n\}$ with $\bar{R}_i$ (alternatively called ρ_i [1]) an $e_1 \ldots e_h$ with $l \in \{1,\ldots,h\}$ (alternatively with $l = h$ [1]) and $\eta = \{e_1,\ldots,e_{l-1}\} \cup dep(e_{l+1} \ldots e_h)$ and the edges $e_{l+1},\ldots,e_h$ nonconverging.
2. Initially, let E_S comprise every $(v_{e,\eta}, v_{e',\eta'})$ with $(\{v_{e,\eta}, v_{e',\eta'}\} \subseteq V_S) \wedge (e \in \eta')$.
3. When removing vertices of G_S to change it into an acyclic (V'_S, E'_S), the set to be maximized and adopted as L is $\{e|\exists v_{e,\eta} \in V'_S\}$.

Secure $(v'_i, v'_j, x/y) \notin E''$ for every $(v_i, v_j, x/y)$ in L.

5.2. The method of Yalcin and Yenigün [11]

Modify the definition of E_U to $\{(v_i^U, v_j, \bar{U}_i/\lambda(s_i,\bar{U}_i))|(\bar{U}_i \in \mathcal{U}) \wedge (s_j = \delta(s_i,\bar{U}_i))\}$. Secure $(v'_i, v'_j, x/y) \notin E''$ for every $(v'_i, v_j^U, x/y)$ in E_C.

5.3. The first method of Duan and Chen [3]

Let $dep(\Gamma)$ denote the set of those edges in E whose input symbol is in the input portion of the label of the invertible path of a member of Γ.

If the basic method is that of Chen & al. [1] or Tekle & al. [8], modify the computation of L as in Section 5.1 while also securing $dep(\Gamma) \cap L = \emptyset$. Otherwise, let L denote $\emptyset$.

Modify the definition of E^Γ to $E^\Gamma = \{((start(W_i))^\gamma, (end(W_i))', \gamma_i)|\gamma_i$ is in Γ and owns a T-sequence as a suffix$\} \cup \{((start(W_i))^\gamma, end(W_i), \gamma_i)|\gamma_i$ is in Γ and does not own a T-sequence as a suffix$\}$.

Modify the definition of E_C to $\{(v'_i, v_j^\gamma, x/y)|((v_i, v_j, x/y) \in (E \setminus dep(\Gamma))) \wedge (v_j^\gamma \in V^\Gamma)\} \cup \{(v'_i, v_j, x/y)|((v_i, v_j, x/y) \in dep(\Gamma)) \vee (v_j^\gamma \notin V^\Gamma)\}$.

Secure $(v'_i, v'_j, x/y) \notin E''$ for every $(v_i, v_j, x/y)$ in L and $(v'_i, v_j^\gamma, x/y)$ in E_C, but not necessarily for every $(s_i, s_j, x/y)$ in the invertible path of a member of Γ.

If the basic method is that of Hierons and Ural [4], implement the correction from the Erratum [5].

5.4. The second method of Duan and Chen [3]

As G^* comprises also the edges from E, whose label is an input/output pair, we for the sake of compatibility assume that the label of any newly introduced edge, originally defined as an input sequence (possibly preceded with a minus sign), is actually the corresponding IOS (inheriting the minus sign, if any).

For $dep(\Gamma)$ as in Section 5.3, modify the constraint for creating a vertex v^*_{e,γ_i} with $(e \in E) \wedge (\gamma_i \in \Gamma) \wedge (end(e) = start(W_i))$ to $e \notin dep(\Gamma)$.

For the edges in G^* labelled ε, we assume that they include the following [10], with v^* denoting the special vertex of G^* created as its root:

1. For every path $\rho = (v_0, v_i, z)$ in A', the edge $(v^*, v^*_{\rho,1}, \varepsilon)$.
2. For every edge e in E with $D_0/\lambda(s_0, D_0)$ for every edge $(v^*_{e,2}, v^*_{e,q}, z)$ in G^* a prefix of $label(e)z$, the edge $(v^*, v^*_{e,1}, \varepsilon)$.
3. For every path $\rho = (v_i, v_j, z)$ in A', the edges $(v_i, v^*_{\rho,1}, \varepsilon)$ and $(v^*_{\rho,2}, v_j, \varepsilon)$.
4. For every edge $e = (v_i, v_j, z)$ in E, with $\delta(s_j, D_j)$ an s_k, the edges $(v_i, v^*_{e,1}, \varepsilon)$ and $(v^*_{e,\beta}, v_k, \varepsilon)$ and for every vertex v^*_{e,γ_i} in G^* with $\gamma_i \in \Gamma$, the edge $(v^*_{e,\gamma_i}, end(W_i), \varepsilon)$.

Secure that the generated rooted path of G^* ends in V and traverses no edge in $dep(\Gamma)$.

6. Final remarks

The idea of the considered methods is to include in the constructed IOS a sufficient set of tests, specific segments whose observation on the system under test in the context of the other tests implies specific desirable properties of the system. We believe that testing specialists should concentrate primarily on identifying small candidate sets of short tests with a provably sufficient collective power, for once a pool of such sets is available, optimally deciding which of them and how the constructed IOS should cover is a routine task that requires

no domain-specific knowledge and can, hence, be left to mathematicians. Only with such separation of concerns, which the considered methods fail to achieve, test construction becomes sufficiently transparent to be reliable and truly opens to new, more efficient kinds of tests and even to new deduction patterns for testing power assessment [2, 6, 7].

References

[1] J. Chen, R. M. Hierons, H. Ural, H. Yenigün, Eliminating redundant tests in a checking sequence, Proc. IFIP Int'l Conf. Testing of Communicating Systems, pp. 146-158, May-June 2005.

[2] M. E. Dincturk, A Two Phase Approach for Checking Sequence Generation, M.Sc. Thesis, Sabancı University, August 2009.

[3] L. Duan, J. Chen, Exploring alternatives for transition verification, J. Syst. Software 82(9) (Sept. 2009) 1388-1402.

[4] R. M. Hierons, H. Ural, Reduced length checking sequences, IEEE Trans. Computers 51(9) (Sept. 2002) 1111-1117.

[5] R. M. Hierons, H. Ural, Erratum: Reduced length checking sequences, IEEE Trans. Computers 58(2) (Feb. 2009) 287.

[6] M. Kapus-Kolar, A Better Procedure and a Stronger State-Recognition Pattern for Checking Sequence Construction, Jožef Stefan Institute Technical Report #10574, 2010.

[7] M. Kapus-Kolar, New state-recognition patterns for conformance testing of finite state machine implementations, submitted for publication, 2010.

[8] K. T. Tekle, H. Ural, M. C. Yalcin, H. Yenigün, Generalizing redundancy elimination in checking sequences, Proc. Int'l Symp. Computer and Information Sciences, pp. 915-925, Oct. 2005.

[9] H. Ural, X. Wu, F. Zhang, On minimizing the lengths of checking sequences, IEEE Trans. Computers 46(1) (Jan. 1997) 93-99.

[10] H. Ural, F. Zhang, Reducing the length of checking sequences by overlapping, Proc. IFIP Int'l Conf. Testing of Communicating Systems, pp. 274-288, May 2006.

[11] M. C. Yalcin, H. Yenigün, Using distinguishing and UIO sequences together in a checking sequence, Proc. IFIP Int'l Conf. Testing for Communicating Systems, pp. 259-273, May 2006.