On the global optimization of checking sequences for finite state machine implementations

Monika Kapus-Kolar*

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

Abstract

A checking sequence for a given domain of deterministic finite state machine implementations is an input sequence for which exactly the non-faulty members of the domain produce a non-faulty response. In the paper, we reconsider a popular family of methods which construct a checking sequence by performing its digraph-based global optimization. Recently, it was demonstrated that many of the methods are unsafe. As a remedy, a simple, but sufficient set of additional constraints on the structure of the employed digraph was introduced. In this paper, we show that the constraints sometimes ban also some of those originally considered checking sequence candidates which are sound. To safely restore the original power of the checking sequence construction approach, we perform its thorough re-engineering. This results in a very transparent and flexible generic method from which various methods of practical interest, both new ones and analogues of the traditional ones, can be derived simply by specialization.

Keywords: Conformance testing, Deterministic finite state machine, Checking sequence construction, Optimization.

*Tel.: +386 1 477 35 31

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

1. Introduction

For a given deterministic finite state machine (DFSM), call it *specification*, and a given domain of its *implementations*, a *checking sequence* is an input sequence to which exactly the non-faulty members of the domain respond as the DFSM would. The advantage of complete test suites consisting of a small number of longer sequences is that their application requires less or no resets of the implementation under test and that they tend to have a better performance on implementations outside their target domain [1]. On the other hand, single-sequence complete test suites are the most difficult to construct and optimize. Methods for their construction, hence, tend to target very narrow domains.

In the paper, we reconsider a popular family of checking sequence construction methods [2–11] which assume that the specification and the target implementations are complete DFSMs defined over the same input alphabet, that the specification is strongly connected and possesses a distinguishing set and that each of the target implementations has at most as many states as the specification. For the checking sequence under construction, the methods perform *digraph-based global optimization*. They start by constructing a digraph in which each of the considered candidate checking sequences is represented as a specific walk whose cost is the specified application cost of the sequence. In the digraph, the methods then look for one of the cheapest such walks.

The reason why we believe that the method family deserves reconsideration is the following: Recently, it was demonstrated that many of the methods are *unsafe* [12]. As a remedy, a simple, but sufficient set of additional constraints on the structure of the constructed digraph was introduced. We, however, discovered that

the solution is not satisfactory, for, as demonstrated in this paper, the constraints sometimes ban also some of those originally considered checking sequence candidates which are sound. This indicates that the checking sequence construction approach needs a *thorough re-engineering*. We accomplish it in the rest of the paper, in a way fully restoring the original power of the approach.

The re-engineering results in a very transparent and flexible *generic checking sequence construction method* from which various methods of practical interest, both new ones and analogues of those of [2–11], can be derived simply by specialization. To make the method even more flexible, so that it can consider an even larger class of candidate checking sequences, we make more flexible also the employed templates for state recognizers and transition implementation tests.

In the Sections 2 to 11, the new generic method is developed and proven step-by-step, in a highly formal and structured way, as to satisfy those diligent readers who seek deep understanding of every detail of the method, for example for its safe direct use or for the development of its new simple-to-use specializations. Each of the Sections 4 to 11, however, starts with an informal summary, as to provide an informal introduction for the detailed readers and to satisfy also practitioners currently using the methods of [2–11] and seeking just a general idea of the new method. The latter are advised to read the sections only up to the phrase "Speaking formally" and to refer to the Sections 2 and 3 only when needing to refresh their memory about a technical term. For the detailed readers, the two sections are indispensable for a proper start, for they define the employed formal notation. Section 12 comprises a discussion and conclusions.

In the following, let M denote the specification DFSM, N its implementation under test, the term "input" a member of their common input alphabet, $\mathcal{D}$ the

distinguishing set of M selected as the basis of checking sequence construction, and G the constructed digraph.

2. Sequences and directed (multi)graphs

Let ϵ denote an empty sequence. For a sequence $\bar{o} = o_1 \dots o_k$, let $seg(\bar{o})$ denote the set of all its segments $o_i \dots o_j$ with $1 \leq i \leq j \leq k$ and $set(\bar{o})$ the set of the objects present in the (multi)set $\{o_1, \dots, o_k\}$. For two sequences $\bar{o} = o_1 \dots o_k$ and $\bar{o}' = o'_1 \dots o'_{k'}$, let $\bar{o} \cdot \bar{o}'$ denote their concatenation $o_1 \dots o_k o'_1 \dots o'_{k'}$. For a sequence $\bar{o} = \bar{o}' \cdot \bar{o}''$, let $\bar{o} - \bar{o}''$ denote $\bar{o}'$.

A *directed (multi)graph* H consists of a set $vr(H)$ of *vertices* and a set $ed(H)$ of *directed edges* which connect them. An edge e has an initial vertex $init(e)$, a final vertex $fin(e)$, a label $lab(e)$ and a cost $cost(e)$. We will specify such an edge as $(init(e), fin(e), lab(e), cost(e))$. A directed (multi)graph can have multiple edges with the same specification. An edge having the same specification as some other is its copy. A *digraph* is a directed (multi)graph in which no edge is a copy of another. For a given directed (multi)graph vertex v, let $in(v)$ denote the set of its incoming edges and $out(v)$ the set of its outgoing edges. A given directed (multi)graph H is *edge-induced* if $vr(H) = \cup_{e \in ed(H)} \{init(e), fin(e)\}$ and *symmetric* if $|in(v)| = |out(v)|$ for every vertex v in $vr(H)$. The cost of a given directed (multi)graph H is $\Sigma_{e \in ed(H)} cost(e)$.

A *walk* of a given directed (multi)graph is a sequence of its consecutive edges. For a walk $w = e_1 \dots e_k$, let $init(w)$ denote its initial vertex $init(e_1)$, $fin(w)$ its final vertex $fin(e_k)$, $vr(w)$ its vertex set $\cup_{1 \leq i \leq k} \{init(e_i), fin(e_i)\}$ and $cost(w)$ its cost $\Sigma_{1 \leq i \leq k} cost(e_i)$. If $k = 0$, $init(w)$ is supposed to be known from the context. If $(k > 0) \wedge (init(w) = fin(w)) \wedge (|vr(w)| = k)$, w is a *cycle*. A *copy* of w is any directed

(multi)graph walk $e'_1 \dots e'_k$ with e'_i for every $1 \leq i \leq k$ a copy of e_i. A directed (multi)graph H is *acyclic* if it has no cycle and *strongly connected* if for every two vertices v and v' in $vr(H)$, it has a walk w with $(init(w) = v) \wedge (fin(w) = v')$.

A *reduction* of a directed (multi)graph H is an edge-induced directed (multi)graph H' with $ed(H') \subseteq ed(H)$. For a directed (multi)graph H and an edge set $E \subseteq ed(H)$, let $H[E]$ denote that reduction of H whose edge set is E. A given directed (multi)graph is a *component* of a directed (multi)graph H if it is its strongly connected reduction and a reduction of no larger strongly connected reduction of H. A *symmetric augmentation* of an edge set E in an edge set E' is a symmetric edge-induced directed (multi)graph whose edge set is E enhanced with zero or more copies of edges in E'.

3. The specification and its implementation under test

For an input x, let $cost(x)$ denote the cost of its application, presumably a non-zero positive real. The default $cost(x)$ is 1. The cost $cost(\bar{x})$ of a given input sequence $\bar{x} = x_1 \dots x_k$ is $\Sigma_{1 \leq i \leq k} cost(x_i)$.

A DFSM Q is a machine possessing a finite set $st(Q)$ of *states* in which it might reside, among them its *initial state* $init(Q)$, and a finite set $tr(Q)$ of *transitions* which it is willing to execute. Every transition in $tr(Q)$ is an $(s, s', x/y)$ with s the state from which it is executed, x the input by which it is provoked, y the output which it produces and s' the state to which it leads, with no other transition defined for x applied in s. If for every state in $st(Q)$, every input has a corresponding transition in $tr(Q)$, Q is *complete*.

By executing a transition $(s, s', x/y)$, a DFSM executes from the state s the *input/output* (I/O) x/y. The *I/O sequences* executable from a given DFSM state s

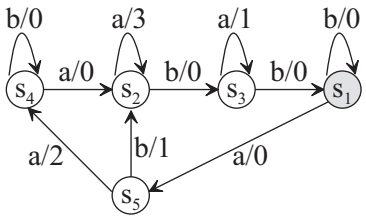

Figure 1: An example M

constitute its *language* $lan(s)$. For a given DFSM Q, let $lan(Q)$ denote $lan(init(Q))$ and $ios(Q)$ the I/O-sequence set $\cup_{s \in st(Q)} lan(s)$. For an I/O sequence $\bar{z}$, let $is(\bar{z})$ denote its input sequence.

A *transition sequence* of a given DFSM Q is a sequence of its consecutive transitions. For such a τ, let $init(\tau)$ denote its initial state, $fin(\tau)$ its final state, $is(\tau)$ its input sequence, $ios(\tau)$ its I/O sequence and $cost(\tau)$ its cost $cost(is(\tau))$. If $init(\tau) = init(Q)$, τ is *rooted*. If Q has no other transition sequence τ' with $(ios(\tau') = ios(\tau)) \wedge (fin(\tau') = fin(\tau))$, τ is *invertible*. For a zero-length transition sequence, the initial state is assumed to be known from the context. A given DFSM Q is *strongly connected* if for every two states s and s' in $st(Q)$, it has a transition sequence τ with $(init(\tau) = s) \wedge (fin(\tau) = s')$.

In the following, we assume that $st(M)$ is an $\{s_1, \ldots, s_n\}$ with $n > 1$ and $init(M) = s_1$. For a transition $(s_i, s_j, x/y)$ in $tr(M)$, let t_x^i be a shorter name.

Example 1. The M in Fig. 1 is a complete and strongly connected DFSM operating over the input alphabet $\{a, b\}$. Its state set is $\{s_1, s_2, s_3, s_4, s_5\}$. Upon receiving a in s_1, it emits 0 and enters s_5. The thereby executed transition t_a^1, i.e. $(s_1, s_5, a/0)$, is invertible, whereas the transition sequence $t_b^1 t_a^1$ is not.

An I/O sequence $\bar{z}$ is a *unique I/O sequence* (UIO) of (a specific state s of) a given DFSM Q if s is the only state s' in $st(Q)$ with $\bar{z} \in lan(s')$. An I/O sequence

$\bar{z}$ is a *backward UIO* (BUIO) of (a specific state s of) a given DFSM Q if s is the only state s' in $st(Q)$ for which Q has a transition sequence τ with $(fin(\tau) = s') \wedge (ios(\tau) = \bar{z})$. For a DFSM state s, let $uio(s)$ denote its UIO set and $buio(s)$ its BUIO set. For a UIO $\bar{z}$ of M, let $ts(\bar{z})$ denote the only transition sequence τ of M with $ios(\tau) = \bar{z}$.

If in two DFSM states, application of a given input sequence results in two different output sequences, the input sequence *separates* the states. A distinguishing set of a given DFSM Q is a set for every state s in $st(Q)$ comprising exactly one of its UIOs, a $\bar{z}_s$, with the property that for every two different states s and s' in $st(Q)$, there is a separating input sequence that is a common prefix of $is(\bar{z}_s)$ and $is(\bar{z}_{s'})$. For a state s_i in $st(M)$, let D_i denote the only member of $\mathcal{D} \cap lan(s_i)$.

For a transition sequence τ of M, let $nxt(\tau)$ denote the set consisting of those transitions t in $tr(M)$ for which M has a transition sequence $\tau' \cdot t$ with $ios(\tau')$ a prefix of $ios(\tau)$ and $is(\tau' \cdot t)$ a prefix of $is(\tau)$.

Example 2. In the M in Fig. 1, ab separates s_1 and s_4, $b/0a/1$ is a UIO of s_2, with $ts(b/0a/1) = t_b^2 t_a^3$, $a/0b/0$ is a BUIO of s_3 and $\{a/0b/1, a/3, a/1, a/0b/0, a/2\}$ is a distinguishing set. If $\mathcal{D}$ is the distinguishing set, then $D_1 = a/0b/1$, $D_2 = a/3$, $D_3 = a/1$, $D_4 = a/0b/0$ and $D_5 = a/2$. $nxt(t_a^1 t_b^5) = \{t_a^1, t_a^2, t_a^3, t_a^4, t_a^5, t_b^2, t_b^5\}$.

A state s_i in $st(M)$ is said to be *properly implemented* in N, denoted as $ok(s_i)$, if $st(N)$ comprises a state, call it s_i', with $D_i \in uio(s_i')$. 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 $ok(s_i) \wedge ok(s_j) \wedge ((s_i', s_j', x/y) \in tr(N))$. For a subset Δ of $tr(M)$, let $ok(\Delta)$ denote $\wedge_{t \in \Delta} ok(t)$. M is said to be properly implemented by N, denoted as $ok(M)$, if $ok(tr(M)) \wedge (init(N) = s_1')$.

For the input sequence which a checking sequence construction method generates, as the input sequence of some I/O sequence $\bar{z}$ in $lan(M)$, let σ denote $seg(\bar{z})$.

4. Primary state recognizers

Recall that $\mathcal{D}$ is such a set of I/O sequences, one for every state of M, that in the case of their execution on N, one knows that each of them has been executed from a different state, which brings the states of N into a one-to-one correspondence with the states of M and confirms that the sequences are UIOs (i.e., I/O sequences recognizing the state from which they are executed) also in the implementation. Besides, one wants to confirm that the destination state of the sequences or their selected extensions executable on M is the expected one, so that the (extensions of) the sequences in $\mathcal{D}$ (call their set $\mathcal{T}$) become confirmed BUIOs (i.e., I/O sequences recognizing the state to which they lead). One, hence, chooses such a set, call it $\mathcal{A}$, of I/O sequences executable on M that in the case of the sequences executed on N, both the UIOs in $\mathcal{D}$ and the BUIOs in $\mathcal{T}$ are confirmed. The format of $\mathcal{A}$ is not prescribed, just make sure that the set comprises also a UIO of the initial state of M.

Speaking formally, the methods of [2–11] for every state s_i in $st(M)$ besides D_i choose yet another special-purpose UIO, a $T_i = D_i \cdot T'_i$, with $\{T_i | s_i \in st(M)\}$ a $\mathcal{T}$. The default $\mathcal{T}$ is $\mathcal{D}$. For every s_i in $st(M)$, with $fin(ts(T_i))$ an s_j, they secure the checking of $D_i \in uio(s'_i)$ and $T_i \in buio(s'_j)$. In general, the checks for $\mathcal{D}$ and $\mathcal{T}$ would be secured by securing that among the subsets of σ, there is also a specific subset of $ios(M)$, in the following called $\mathcal{A}$, with $ok'(\Delta)$ for a given subset Δ of $tr(M)$ denoting $(\mathcal{A} \subseteq ios(N)) \wedge ok(\Delta)$. In the methods, $\mathcal{A} = \{T_i \cdot T_j | (s_i \in st(M)) \wedge (fin(ts(T_i)) = s_j)\}$. We observe that the checks remain secured also if $\mathcal{A}$

is simplified into $\{T_i \cdot D_j | (s_i \in st(M)) \wedge (fin(ts(T_i)) = s_j)\}$, the new default $\mathcal{A}$. In specific cases, $\mathcal{A}$ can be safely simplified even further.

Example 3. For a transition $(s_i, s_i, x/y)$ in $tr(M)$ with $D_i = T_i = x/yx/y$, $T_i \cdot D_i$ in the proposed $\mathcal{A}$ can be safely replaced with $x/yx/yx/y$.

An alternative option is to deliberately choose (for example because a powerful ad hoc checking sequence construction method suggests it) a richer $\mathcal{A}$, so that $\mathcal{A} \subseteq ios(N)$ can be of more help in the checking of transition implementations. In any case, we assume $\mathcal{A} \cap uio(s_1) \neq \emptyset$.

5. Conditionally safe state recognizers

In this section, we propose more flexible templates for conditionally safe state recognizers. In the next two paragraphs, we informally describe the main precondition relaxations proposed.

In the method of [8], one is allowed to employ also those UIOs of M which are not in $\mathcal{D}$, but only after implementation checking has already been secured for the members of $\mathcal{A}$ and for a specific set of transitions of M. The set comprises every transition of M whose input is one of those in the UIO. We observe that a smaller, but still sufficient set can be computed as follows: For every state of M, consider application of the input sequence of the UIO up to and including the point when the end of the sequence is reached or the resulting output sequence starts differing from that of the UIO. The applications result in a set of transition sequences. The transitions which one looks for are the members of the sequences.

In the method of [10], application of a UIO of M which is not in $\mathcal{D}$ is legal provided that the corresponding transition sequence of M is a sequence τ of invertible

transitions followed by a transition sequence τ' corresponding to a member of $\mathcal{D}$, but only if implementation checking has already been secured for the members of $\mathcal{A}$ and for a specific set of transitions of M. According to a correction in [12], the set comprises every transition of M whose input is one of those in the input sequence of τ. We observe that a smaller, but still sufficient set can be computed as follows: For every state of M, consider application of the input sequence of τ up to and including the point when the end of the sequence is reached or the resulting output sequence starts differing from that of τ. The applications result in a set of transition sequences. The transitions which one looks for are the members of the sequences, with the possibility that in the case of τ just a single transition, the transition is exempted from the set. As another relaxation, we observe that the transitions in τ need not be invertible, provided that τ as a whole is invertible. Besides, the I/O sequence of τ' need not be a member of $\mathcal{D}$, provided that it is some other UIO of M which can be trusted under the adopted assumptions.

Speaking formally, once $\mathcal{A} \subseteq \sigma$ is secured, the UIOs in $\mathcal{D}$ and the BUIOs in $\mathcal{T}$ become safe for state recognition in transition implementation tests. Additionally employed UIOs and BUIOs of individual states s_i in $st(M)$ are in the methods of [2–11] trusted as UIOs or BUIOs of s'_i, respectively, under the condition that the checking of $ok'(\Delta)$ is already secured for a specific subset Δ of $tr(M)$. We call such conditionally safe UIOs and conditionally safe BUIOs (Δ, i)-UIOs and (Δ, i)-BUIOs, respectively.

For a subset Δ of $tr(M)$ and a state s_i in $st(M)$, a (Δ, i)-UIO is any UIO $\bar{z}$ of s_i with $(\bar{z} \in ios(N)) \wedge ok'(\Delta)$ a sufficient condition for $\bar{z} \in uio(s'_i)$. In particular, every D_i (and, hence, T_i) is an $(\emptyset, i)$-UIO. We observe that the conditions under which the methods virtually trust that for a transition sequence τ of M, a state s_i

in $st(M)$ and a subset Δ of $tr(M)$, $ios(\tau)$ is a (Δ, i)-UIO remain sufficient if they are relaxed to the requirement that τ is a $\tau_1 \cdot \tau_2 \cdot \tau_3$, with $init(\tau_2)$ an s_j, satisfying the following:

(1) $ios(\tau_2) \in uio(s_j)$.

(2) $(ios(\tau_2) = D_j) \vee (nxt(\tau_2) \subseteq \Delta)$.

(3) τ_1 is invertible and $init(\tau_1) = s_i$.

(4) If τ_1 is just a single transition, a t, then $(nxt(\tau_1) \setminus \{t\}) \subseteq \Delta$ else $nxt(\tau_1) \subseteq \Delta$.

Theorem 1. The above conditions are sufficient.

Proof. Suppose that $ok'(\Delta)$ and N has a walk $\tau'_1 \cdot \tau'_2 \cdot \tau'_3$ with $ios(\tau'_i) = ios(\tau_i)$ for $1 \leq i \leq 3$. If $ios(\tau_2) = D_j$ then, by $\mathcal{A} \subseteq ios(N)$, $ios(\tau_2) \in uio(s'_j)$ and, hence, $init(\tau'_2) = s'_j$. Otherwise, by $(nxt(\tau_2) \subseteq \Delta) \wedge ok'(\Delta) \wedge (ios(\tau_2) \in uio(s_j))$, $ios(\tau_2) \in uio(s'_j)$ and, hence, $init(\tau'_2) = s'_j$. By (3), (4) and $(init(\tau'_2) = s'_j) \wedge ok'(\Delta)$, $init(\tau'_1) = s'_i$ and, hence, $ios(\tau) \in uio(s'_i)$. □

For a subset Δ of $tr(M)$ and a state s_i in $st(M)$, a (Δ, i)-BUIO is any BUIO $\bar{z}$ of s_i with $(\bar{z} \in ios(N)) \wedge ok'(\Delta)$ a sufficient condition for $\bar{z} \in buio(s'_i)$. In particular, every T_j with $fin(ts(T_j)) = s_i$ is an $(\emptyset, i)$-BUIO. We observe that the conditions under which the methods virtually trust that for a transition sequence τ of M, a state s_i in $st(M)$ and a subset Δ of $tr(M)$, $ios(\tau)$ is a (Δ, i)-BUIO remain sufficient if they are relaxed to the requirement that τ is a $\tau_1 \cdot \tau_2 \cdot \tau_3$, with $init(\tau_2)$ an s_j and $fin(\tau_2)$ an s_k, satisfying the following:

(1) $ios(\tau_2)$ is a (Δ, j)-UIO.

(2) $(ios(\tau_2) = T_j) \vee (set(\tau_2) \subseteq \Delta)$.

(3) $(fin(\tau_3) = s_i) \wedge (set(\tau_3) \subseteq \Delta)$.

Theorem 2. The above conditions are sufficient.

Proof. Suppose that $ok'(\Delta)$ and N has a walk $\tau'_1 \cdot \tau'_2 \cdot \tau'_3$ with $ios(\tau'_i) = ios(\tau_i)$ for $1 \leq i \leq 3$. If $ios(\tau_2) = T_j$ then, by $\mathcal{A} \subseteq ios(N)$, $ios(\tau_2) \in buio(s'_k)$ and, hence, $fin(\tau'_2) = s'_k$. Otherwise, by $ok'(\Delta)$ and $ios(\tau_2)$ a (Δ, j)-UIO, $ios(\tau_2) \in uio(s'_j)$ and, hence, $init(\tau'_2) = s'_j$ and, hence, by $(set(\tau_2) \subseteq \Delta) \wedge ok(\Delta)$, $fin(\tau'_2) = s'_k$. By (3) and $(fin(\tau'_2) = s'_k) \wedge ok(\Delta)$, $fin(\tau'_3) = s'_i$ and, hence, $ios(\tau) \in buio(s'_i)$. □

6. Candidate transition implementation tests

With the above defined primary and conditionally safe state recognizers, one can construct (conditionally safe) transition implementation tests. In this section, we propose for them a more flexible template. Its novelty is in (only slightly, to keep it simple) more flexible combining of forward and backward state recognition.

Speaking formally, as we construct a checking sequence indirectly, by constructing the corresponding rooted transition sequence of M, we define (conditionally safe) transition implementation tests (TITs) as candidates for special-purpose segments of the transition sequence. For a subset Δ of $tr(M)$ and a transition t in $tr(M)$, a (Δ, t)-TIT is any transition sequence τ of M with $(ios(\tau) \in ios(N)) \wedge ok'(\Delta)$ a sufficient condition for $ok(t)$. We observe that the conditions under which the methods of [2–11] virtually trust that for a transition t in $tr(M)$ and a subset Δ of $tr(M)$, a transition sequence τ of M is a (Δ, t)-TIT remain sufficient if they are relaxed to the requirement that τ is a $\tau_1 \cdot \tau_2 \cdot t \cdot \tau_3 \cdot \tau_4$, with $init(\tau_2)$ an s_h, $init(t)$ an s_i, $init(\tau_3)$ an s_j and $init(\tau_4)$ an s_k, satisfying the following:

(1) If $ios(\tau_1 \cdot \tau_2)$ is not a (Δ, i)-BUIO, then $ios(\tau_2 \cdot t \cdot \tau_3 \cdot \tau_4)$ is a (Δ, h)-UIO and $set(\tau_2) \subseteq \Delta$.

(2) If $ios(\tau_3 \cdot \tau_4)$ is not a (Δ, j)-UIO, then $ios(\tau_1 \cdot \tau_2 \cdot t \cdot \tau_3)$ is a (Δ, k)-BUIO and $ios(\tau_3) \cdot D_k$ is a (Δ, j)-UIO.

Theorem 3. The above conditions are sufficient.

Proof. Suppose that $ok'(\Delta)$ and N has a walk $\tau'_1 \cdot \tau'_2 \cdot t' \cdot \tau'_3 \cdot \tau'_4$ with $ios(\tau'_i) = ios(\tau_i)$ for $1 \leq i \leq 4$ and $ios(t') = ios(t)$. If $ios(\tau_1 \cdot \tau_2)$ is a (Δ, i)-BUIO then, by $ok'(\Delta)$, $ios(\tau_1 \cdot \tau_2) \in buio(s'_i)$ and, hence, $init(t') = s'_i$. Otherwise, by $ok'(\Delta)$ and $ios(\tau_2 \cdot t \cdot \tau_3 \cdot \tau_4)$ a (Δ, h)-UIO, $ios(\tau_2 \cdot t \cdot \tau_3 \cdot \tau_4) \in uio(s'_h)$ and, hence, $init(\tau'_2) = s'_h$ and, hence, by $(set(\tau_2) \subseteq \Delta) \wedge ok(\Delta)$, $init(t') = s'_i$.

If $ios(\tau_3 \cdot \tau_4)$ is a (Δ, j)-UIO then, by $ok'(\Delta)$, $ios(\tau_3 \cdot \tau_4) \in uio(s'_j)$ and, hence, $fin(t') = s'_j$. Otherwise, by $ok'(\Delta)$ and $ios(\tau_1 \cdot \tau_2 \cdot t \cdot \tau_3)$ a (Δ, k)-BUIO, $ios(\tau_1 \cdot \tau_2 \cdot t \cdot \tau_3) \in buio(s'_k)$ and, hence, $init(\tau'_4) = s'_k$ and, hence, N has a walk $t' \cdot \tau'_3 \cdot \tau'_5$ with $ios(\tau'_5) = D_k$ and, hence, by $ok'(\Delta)$ and $ios(\tau_3) \cdot D_k$ a (Δ, j)-UIO, $ios(\tau_3) \cdot D_k \in uio(s'_j)$ and, hence, $fin(t') = s'_j$. □

Example 4. Suppose that $st(M) = \{s_1, s_2, s_3\}$ and $\tau \cdot t_a^1 \cdot t_b^2 \cdot t_c^3 \cdot \tau'$ is a transition sequence of M. If $ios(\tau \cdot t_a^1 \cdot t_b^2 \cdot t_c^3) \in \mathcal{T}$ and $ios(t_a^1 \cdot t_b^2 \cdot t_c^3 \cdot \tau') \in \mathcal{D}$ and t_c^3 is invertible, the transition sequence is a $(\{t_a^1, t_c^1, t_c^2\}, t_b^2)$-TIT.

7. Candidate rooted transition sequences

If a transition sequence of M is to be trusted to correspond to a checking sequence, it must correspond to a UIO of the initial state of M, it must cover every member of $\mathcal{A}$ and a sufficient set of transition implementation tests and the dependency relation between the constituent tests must be acyclic.

Speaking formally, the conditions under which the methods of [2–11] trust that the input sequence of a rooted transition sequence τ of M is a checking sequence are virtually (a specialization of) the following:

(1) $\mathcal{A} \subseteq seg(ios(\tau))$ and $ios(\tau) \in uio(s_1)$.

(2) There exist such a subset Δ of $tr(M)$ and permutation $t_1, \ldots, t_{|tr(M)\setminus\Delta|}$ of the transitions in $tr(M) \setminus \Delta$ that $\mathcal{A} \subseteq ios(N)$ is a sufficient condition for $ok(\Delta)$ and for every $1 \leq i \leq |tr(M) \setminus \Delta|$, $seg(\tau)$ comprises a (Δ_i, t_i)-TIT with $\Delta_i \subseteq (\Delta \cup \{t_1, \ldots, t_{i-1}\})$.

Theorem 4. The above conditions are sufficient.

Proof. Suppose that $ios(\tau) \in lan(N)$ and, hence, $ok'(\Delta)$. By (2), this by induction on i secures $ok(t_i)$ for $1 \leq i \leq |tr(M) \setminus \Delta|$. N is, hence, isomorphic to M. Hence, by $ios(\tau) \in (uio(s_1) \cap lan(N))$, $init(N) = s'_1$ and, hence, $ok(M)$. □

In the following, a rooted transition sequence of M which satisfies the above constraints is called a *checking rooted transition sequence* (CRTS).

8. An outline of the employed digraph

The recommended structure of the digraph G is exemplified in Fig. 2, for the M in Fig. 1. Every vertex of the digraph corresponds to a specific state of M. In the central part of the digraph, the one in the grey area, every walk starting in the top row of vertices and ending in the bottom one represents some member of $\mathcal{A}$ or a (conditionally safe) candidate implementation test for an individual transition of M. In both cases, the concatenation of the labels of the consecutive edges of the walk is the corresponding transition sequence of M.

The rest of G specifies the possibilities for the interconnection of the transition sequences. On the left, we see a part isomorphic to M without its loop transitions. From every vertex of the part, there is a link to every vertex in the top row of vertices in the central part of G which corresponds to the same state. From every

vertex in the bottom row of vertices in the central part of G, there is a link to the corresponding vertex of the left part. The left part of G, together with the two kinds of links, specifies how the transition sequences specified in the central part can be interconnected via auxiliary transition sequences.

The remaining edges, of which only an illustrative one is depicted in Fig. 2, specify for the transition sequences specified in the central part of G the possibilities for the interconnection with overlapping. The label of such an edge is a transition sequence of M preceded by a minus. Whenever in a walk of G, the edge links a pair of edges, the transition sequence is the presumed overlapping of the transition sequence specified by the previous edge and that specified by the following one. To the cost of the transition sequence specified by the walk, such an edge, hence, contributes the negative cost of the transition sequence in its label. For the other kinds of edges, whose label is a (possibly empty) transition sequence, the cost is that of the sequence.

Speaking formally, a CRTS in general consists of (possibly overlapping) *essential segments* and of segments serving for their interconnection, call them *transfer transition sequences*. None of the advanced ones among the methods of [2–11] takes care that in the constructed digraph, the walks representing candidate transfer transition sequences consist solely of edges not belonging to any walk representing a candidate essential segment. In other words, it is possible that in their digraph, the part specifying candidate transfer transition sequences intersects with the part specifying candidate essential segments, which is a problem, for the latter is currently supposed to respect rigorous constraints [12].

Example 5. Take the M in Fig. 1 with $\mathcal{D} = \mathcal{T} = \{a/0b/1, a/3, a/1, a/0b/0, a/2\}$. According to the method of [7] enhanced with the redundant-TIT elimination of

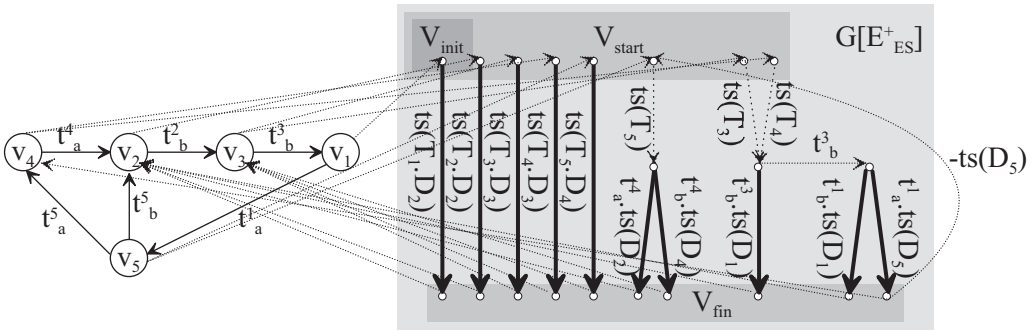

Figure 2: An example G, with the edges in E^-, except for an illustrative one, invisible

[4], a CRTS for the case is constructed as a specific kind of composition of the transition sequences $t_a^1 t_b^5 t_a^2$, $t_a^2 t_a^2$, $t_a^3 t_a^3$, $t_a^4 t_b^2 t_a^3$, $t_a^5 t_a^4 t_b^2$, $t_a^1 t_a^5$, $t_b^1 t_a^1 t_b^5$, $t_b^3 t_a^1 t_b^5$, $t_a^4 t_a^2$ and $t_b^4 t_a^4 t_b^2$, with no TITs explicitly provided for the transitions t_a^2, t_a^3, t_a^5, t_b^2 and t_b^5. If, however, the digraph originally constructed to implement the strategy is corrected as suggested in [12], the five transitions are no longer available for the interconnection of the transition sequences, but without employing t_b^2 for the purpose, the interconnection is impossible. In the G partially depicted in Fig. 2, the central part specifies, under the interpretation formalized below, exactly the relevant ones of the candidate essential segments specified in the original digraph, whereas the rest of G specifies all the originally specified options for their interconnection, thereby solving the problem.

As evident from Fig. 2, we suggest that $ed(G)$ consists of three disjoint sets of edges, an E^+_{ES} with $G[E^+_{ES}]$ specifying candidate essential segments, an E^+_{TTS} with $G[E^+_{TTS}]$ specifying candidate transfer transition sequences, and an E^- specifying the possible *essential segments overlaps*, with $E^+_{ES} \uplus E^+_{TTS}$ an E^+. Every edge e in $ed(G)$ is associated with a transition sequence $ts(e)$ of M, with $lab(e) = ts(e)$ and $cost(e) = cost(ts(e))$ if in E^+ and with $lab(e) = -ts(e)$ and $cost(e) = -cost(ts(e))$

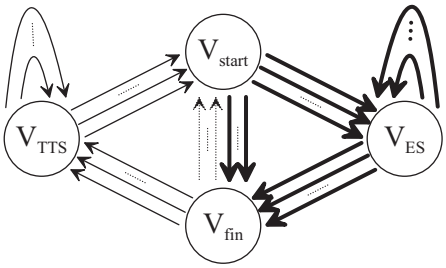

Figure 3: The recommended G sketched with the edges in E^+_{ES} bold, the edges in E^- dotted and the edges in E^+_{TTS} ordinary

if in E^-. For a walk $w = e_1 \dots e_k$ of $G[E^+]$, let $lab(w)$ denote its label $lab(e_1) \cdot \ldots \cdot lab(e_k)$.

The vertices of G are the endpoints of its edges. According to their position in G, we partition them into the sets V_{start}, V_{ES}, V_{fin} and V_{TTS}, in a way evident from Fig. 3. A walk w of G specifies a (conditionally safe) candidate essential segment if it is a walk of $G[E^+_{ES}]$ with $init(w)$ in V_{start} and $fin(w)$ in V_{fin}. A walk w of G specifies a candidate transfer transition sequence if it is a walk of $G[E^+_{TTS}]$ with $init(w)$ in V_{fin} and $fin(w)$ in V_{start}. In both cases, the specified transition sequence is $lab(w)$.

For every vertex v in V_{start}, it is assumed that $ts(e)$ is the same for every edge e in $out(v)$. For every vertex v in V_{fin}, it is assumed that $ts(e)$ is the same for every edge e in $in(v)$. It is assumed that V_{start} comprises a vertex v with $ios(ts(e)) \in uio(s_1)$ for the edges e in $out(v)$, with V_{init} denoting the set of all such vertices.

E^+_{TST} comprises:

1. for every transition $t = (s_i, s_j, x/y)$ in $tr(M)$ with $s_i \neq s_j$, an edge $(v_i, v_j, t, cost(t))$,
2. for every vertex v in V_{start}, with $init(ts(e))$ of the edges e in $out(v)$ an s_i, an edge $(v_i, v, \epsilon, 0)$ and

3. for every vertex v in V_{fin}, with $fin(ts(e))$ of the edges e in $in(v)$ an s_i, an edge $(v, v_i, \epsilon, 0)$.

E^- comprises every feasible edge $(v, v', -\tau, -cost(\tau))$ satisfying the following:

(1) $v \in V_{fin}$ and $v' \in V_{start}$.

(2) For the edges e in $in(v)$, τ is a non-empty suffix of $lab(e)$.

(3) For the edges e in $out(v')$, τ is a prefix of $lab(e)$.

(4) The length of τ is maximized.

For a walk $w = w_1 e_1 \ldots w_k e_k w_{k+1}$ of G with every w_i a non-empty walk of $G[E^+]$ and every e_i an edge in E^-, let $lab(w)$ denote its label $(lab(w_1) - ts(e_1)) \cdot \ldots \cdot (lab(w_k) - ts(e_k)) \cdot lab(w_{k+1})$, with every $ts(e_i)$ the shared part of the consecutive (partially) overlapping segments $lab(w_i)$ and $lab(w_{i+1})$.

9. The central part of the digraph

The proposed structure of the central part of the digraph G is exemplified in Fig. 5, for the M in Fig. 4. The undirected lines in Fig. 5 are not a part of the depicted digraph and indicate which of the depicted vertices actually represent the same vertex of the digraph. We see that the digraph consists of five parts. They are depicted in the order in which they have been conceived. The leftmost part specifies the members of $\mathcal{A}$. Each of the remaining parts specifies candidate implementation tests for some transitions of M not covered by any previously conceived part, where some transitions are covered (it does not matter how) already by the leftmost part.

In the general case, there can be any number of parts, provided that their collection covers the entire transition set of M. The incremental approach to the

design of the central part of G is recommended because it allows that in the design of any part, one safely assumes correct implementation of $\mathcal{A}$ and of every transition of M covered by a previously conceived part. Thanks to this, a part can sometimes specify an efficient candidate transition implementation test which otherwise could not be trusted.

The leftmost part consists of edges representing individual members of $\mathcal{A}$. The structure recommended for the remaining parts is exemplified in the fourth part of the digraph depicted in Fig. 5. The core of such a part (the dotted edges) consists of edges individually representing the relevant ones among those non-loop transitions of M which are covered by a previously conceived part or are among the target transitions of the part. Entering the core, there are one or more (dashed) edges representing candidate BUIOs safe under the assumptions adopted for the part. Leaving the core, there are (bold) edges representing individual target transitions of the part. Emanating from the final vertex of every such edge, there are one or more (ordinary) edges representing candidate UIOs safe under the assumptions adopted for the part. If a bold edge is followed by a single ordinary edge, the two edges can be merged into a bold one, as in the second part of the digraph depicted in Fig. 5.

If a part has a single target transition and the transition is invertible and every other transition with the same input is covered by a previously conceived part, its core and its BUIO edges can be omitted, as in the rightmost part of the digraph depicted in Fig. 5. If a part has a single target transition and specifies for it a single candidate implementation test, the candidate can be of any desired kind and represented by a single (bold) edge, as in the third part of the digraph depicted in Fig. 5. The edges of the leftmost part of the central part of G are also assumed

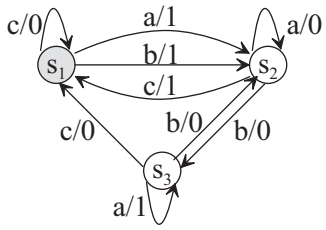

Figure 4: An example M

to be bold. Being bold gives an edge a special status in the final step of checking sequence construction.

Speaking formally, in the methods of [2–11], one of the necessary conditions for trusting that the label of a given walk w of G is a CRTS (call such walks *CRTS walks*) is also that for a specific subset of $ed(G)$, call it E^{acyc}, the digraph $G[set(w) \cap E^{acyc}]$ is *acyclic*. Every edge in E^{acyc} represents an individual transition in $tr(M)$. According to [12], the transitions cannot be freely employed in the (conditionally safe) candidate TITs specified in G, which is a problem, for it decreases the chances that G has a cheap CRTS walk.

Example 6. Take the M in Fig. 4 with $\mathcal{D} = \mathcal{T} = \{a/1a/0, a/0, a/1a/1\}$. According to the method of [8], the UIO of s_1 employed in a candidate TIT for the transition t_c^3 can be either D_1 or the shorter $b/1$. If, however, the G constructed according to the strategy is corrected as suggested in [12], the latter is no longer possible. The $G[E_{ES}^+]$ in Fig. 5, constructed as suggested below, circumvents the problem, simplifies $\mathcal{A}$ and candidate TITs for t_c^2 and eliminates more redundant TITs than the method of [5]. Unlike the corrected version of the method of [8], it also manages to employ $b/1$ as a short BUIO.

We suggest that one partitions $tr(M)$ into a $\Delta_0 \uplus \ldots \uplus \Delta_p$ with $\mathcal{A} \subseteq ios(N)$ a sufficient condition for $ok(\Delta_0)$, with $\uplus_{0 \leq j \leq i} \Delta_j$ a Δ_i' for $0 \leq i \leq p$. E_{ES}^+ is then

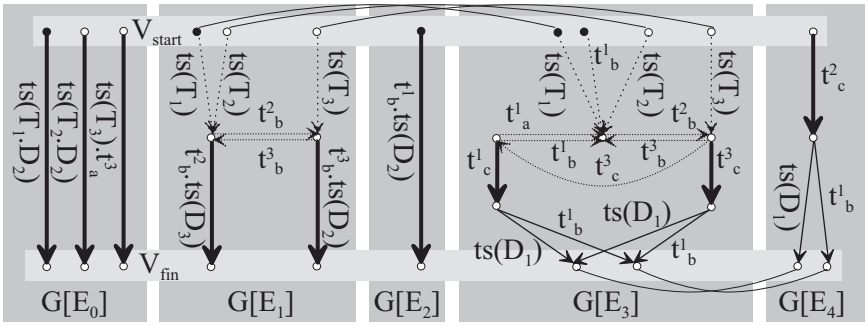

Figure 5: An example $G[E^{+}_{ES}]$ for $tr(M) = \{t^1_a, t^2_a, t^3_a\} \uplus \{t^2_b, t^3_b\} \uplus \{t^1_b\} \uplus \{t^1_c, t^3_c\} \uplus \{t^2_c\}$, with the edges in E^{trg} bold, the edges in E^{acyc} dotted, the edges in E^{buio} dashed, the edges in E^{uio} ordinary and the vertices in V_{init} black. The undirected lines are not a part of the digraph and indicate which of its depicted vertices are actually identical.

conceived as an $E_0 \cup \ldots \cup E_p$ with $G[E_0]$ specifying $\mathcal{A}$ and every $G[E_i]$ with $1 \leq i \leq p$ specifying candidate TITs for the transitions in Δ_i. The default p for the case of $\Delta_0 \neq tr(M)$ is 1.

We suggest that for $1 \leq i \leq p$, E_i is conceived as an $E^{buio}_i \uplus E^{core}_i \uplus E^{trg}_i \uplus E^{uio}_i$, with $\cup_{1 \leq j \leq p} E^{buio}_j$ an E^{buio}, $\cup_{1 \leq j \leq p} E^{core}_j$ our E^{acyc}, $E_0 \cup (\cup_{1 \leq j \leq p} E^{trg}_j)$ an E^{trg} and $\cup_{1 \leq j \leq p} E^{uio}_j$ an E^{uio}.

In the following, if we call a vertex u_τ or w_τ (with τ a transition sequence of M), it belongs to V_{start} or V_{fin}, respectively. Otherwise, it belongs to V_{ES}.

The edges in E_0 represent individual members of $\mathcal{A}$, where the edge corresponding to a given $\bar{z}$ in $\mathcal{A}$ is specified as $(u_\tau, w_\tau, \tau, cost(\tau))$ with τ a selected transition sequence of M with $ios(\tau) = \bar{z}$.

For $1 \leq i \leq p$, we define three alternative forms of $G[E_i]$, sketched in Fig. 6. The *first option* is that Δ_i comprises a single transition, a t, and E_i comprises a single edge, an $e = (u_\tau, w_\tau, \tau, cost(\tau))$ with τ a (Δ'_{i-1}, t)-TIT, with the singleton walk set $\{e\}$ a W_t. In Fig. 5, an example of such a $G[E_i]$ is $G[E_2]$. Preferably,

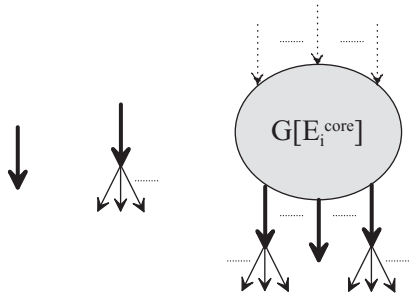

Figure 6: The three alternative forms of $G[E_i]$ for $i > 0$ sketched with the edges in E_i^{trg} bold, the edges in E_i^{buio} dashed and the edges in E_i^{uio} ordinary

select e among the members of $E_0 \cup (\cup_{1 \leq j \leq i-1} E_j^{trg})$.

The *second option* is that Δ_i comprises a single transition, an invertible t with $fin(t)$ an s_j, $(nxt(t) \setminus \{t\}) \subseteq \Delta'_{i-1}$, E_i^{buio} and E_i^{core} are empty, E_i^{trg} comprises a single edge, an $e = (u_t, q_t, t, cost(t))$, E_i^{uio} is non-empty and every edge in E_i^{uio} is a $(q_t, w_\tau, \tau, cost(\tau))$ with τ a selected transition sequence of M with $ios(\tau)$ a (Δ'_{i-1}, j)-UIO, with the walk set $\{e \cdot e' | e' \in E_i^{uio}\}$ a W_t. In Fig, 5, an example of such a $G[E_i]$ is $G[E_4]$.

The *third option* (the default choice) is that $G[E_i]$ is structured as sketched in Fig. 6 on the right and defined in the following:

(1) E_i^{buio} is non-empty and every edge in it is a $(u_\tau, v_j^i, \tau, cost(\tau))$ with τ a selected transition sequence of M, $s_j = fin(\tau)$ and $ios(\tau)$ a (Δ'_{i-1}, j)-BUIO. The default candidates for $ios(\tau)$ are the members of $\mathcal{T}$.

(2) Every edge in E_i^{core} is a $(v_j^i, v_k^i, t, cost(t))$ with t a selected transition in Δ'_i, $s_j = init(t)$, $s_k = fin(t)$ and $s_j \neq s_k$. The default candidates for t are all those transitions in Δ'_i which are not a loop.

(3) E_i^{trg} consists of one edge per each individual transition t in Δ_i, with $init(t)$ an s_j and $fin(t)$ an s_k. The edge can be either $(v_j^i, q_t, t, cost(t))$ with $|out(q_t)| > 1$

or (the default option) a $(v^i_j, w_{t\cdot\tau}, t\cdot\tau, cost(t\cdot\tau))$ with τ a selected transition sequence of M with $ios(\tau)$ a (Δ'_{i-1}, k)-UIO, where the default τ is $ts(D_k)$.

(4) E^{uio}_i consists of two or more edges per each individual transition t in Δ_i whose edge in E^{trg}_i ends in q_t. Every edge for such a t, with $fin(t)$ an s_j, is a $(q_t, w_\tau, \tau, cost(\tau))$ with τ a selected transition sequence of M with $ios(\tau)$ a (Δ'_{i-1}, j)-UIO. The only default candidate for τ is $ts(D_j)$.

(5) For a transition t in Δ_i, let W_t denote the set of those walks of $G[E_i]$ which start in V_{start}, end in V_{fin} and traverse the edge in E^{trg}_i whose label begins with t. It is assumed that every edge in E_i belongs to a walk in W_t of some t in Δ_i. This is trivial to achieve at least if $\Delta'_i = tr(M)$.

Example 7. In Fig. 5, the third kind of structuring has been employed for $G[E_1]$ and $G[E_3]$. In E^{core}_3, the only useful edges are those labelled t^2_b and t^3_c. If its useless members are deleted, its t^2_b edge can be merged with each of the preceding edges in E^{buio}_3.

10. Candidate walks

For an explanation of which walks in the digraph G are candidates for selection, because they are trusted to correspond to a checking sequence, let us reconsider the example G partially depicted in Fig. 2. In the top row of vertices in the central part of G (the grey area), every vertex by construction has the property that its outgoing edges all have the same label. Of special interest are those for which the label corresponds to a UIO of the initial state of M, for these are the vertices of G in which candidate walks are allowed to start. The vertices in which they are allowed to end are those in the bottom row of vertices in the central part of G.

For an explanation of the remaining two conditions which a walk of G has to satisfy to be a candidate walk, reconsider the example central part of G depicted in Fig. 5. Recall that the central part of G has bold and non-bold edges. A candidate walk traverses every bold edge at least once. Recall also that the central part of G is a series of parts, of which all but the first one are allowed to have a core. In the figure, the edges of the cores are the dotted ones and in the fourth part, we see that cores can have cycles. It can happen that a walk of G comprises every edge of such a cycle, but then it does not qualify for a candidate walk.

Speaking formally, we define that a walk w of G is a CRTS walk if $(init(w) \in V_{init}) \wedge (fin(w) \in V_{fin}) \wedge (E^{trg} \subseteq set(w))$ and the digraph $G[set(w) \cap E^{acyc}]$ is acyclic.

Theorem 5. The above conditions are sufficient.

Proof. By $(init(w) \in V_{init}) \wedge (fin(w) \in V_{fin})$, $lab(w)$ of such a w is a transition sequence of M (recall the last paragraph of Section 8), a τ.

For every $\bar{z}$ in $\mathcal{A}$, $set(w)$ by $E_0 \subseteq E^{trg} \subseteq set(w)$ comprises an edge e with $ios(lab(e)) = \bar{z}$. Hence, $\mathcal{A} \subseteq seg(ios(\tau))$.

By $init(w) \in V_{init}$, w starts with an edge e satisfying $ios(lab(e)) \in uio(s_1)$. Hence, $ios(\tau) \in uio(s_1)$.

$\mathcal{A} \subseteq ios(N)$ is a sufficient condition for $ok(\Delta_0)$. For every $1 \leq i \leq p$ and transition t in Δ_i, take any walk w_t in $seg(w) \cap W_t$, with $\Delta'_{i-1} \cup \{t' | (t' \in tr(M)) \wedge \exists e \in (set(w_t) \cap E_i^{core}) : (ts(e) = t')\}$ a Δ''_t and $lab(w_t)$ a (Δ''_t, t)-TIT. Its existence is implied by $(init(w) \in V_{init}) \wedge (fin(w) \in V_{fin}) \wedge (E^{trg} \subseteq set(w))$.

For every $1 \leq i \leq p$, by $G[set(w) \cap E^{acyc}]$ acyclic, $G[set(w) \cap E_i^{core}]$ is acyclic. There, hence, exists such a permutation $t^i_1, \ldots, t^i_{|\Delta_i|}$ of the transitions in Δ_i that for

every $1 \leq j \leq |\Delta_i|$, $\Delta''_{t^i_j} \subseteq (\Delta'_{i-1} \cup \{t^i_1, \ldots, t^i_{j-1}\})$. Hence, $t^1_1, \ldots, t^1_{|\Delta_1|}, \ldots, t^p_1, \ldots, t^p_{|\Delta_p|}$ is such a permutation $t_1, \ldots, t_{|tr(M)\setminus\Delta_0|}$ of the transitions in $tr(M)\setminus\Delta_0$ that for every $1 \leq i \leq |tr(M)\setminus\Delta_0|$, $seg(\tau)$ comprises a (Δ''_{t_i}, t_i)-TIT with $\Delta''_{t_i} \subseteq (\Delta_0 \cup \{t_1, \ldots, t_{i-1}\})$. □

11. The search for a minimum-cost candidate walk

Following an idea of [6], we suggest that the search for a minimum-cost candidate walk of G proceeds as follows:

1. Construct a digraph which, as sketched in Fig. 7, consists of the edges of G, of an additional bold edge, of edges connecting the final vertex of the bold edge to those vertices of G in which a candidate walk can start, and of edges connecting to the starting vertex of the bold edge those vertices of G in which a candidate walk can end.
2. From copies of edges of the above digraph, construct one of the minimum-cost symmetric digraphs comprising, among other edges, exactly one copy of the additional bold edge and at least one copy of each individual bold edge of G.
3. If the digraph consists of multiple components, connect them by adding copies of adequate cycles of the digraph G.
4. In the resulting digraph, construct a walk starting and ending in the initial vertex of the additional bold edge and traversing every edge exactly once. By removing its first two edges and its last one, one obtains an optimized candidate walk of G.

Speaking formally, in the first step, one constructs a digraph G' defined as follows:

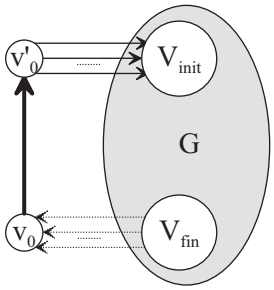

Figure 7: G' sketched with e_0 bold, the edges in E_{fin} dotted and the edges in E_{init} ordinary

$vr(G') = vr(G) \uplus \{v_0, v'_0\}$

$ed(G') = ed(G) \uplus \{e_0\} \uplus E_{init} \uplus E_{fin}$

$e_0 = (v_0, v'_0, \epsilon, 0)$

$E_{init} = \{(v'_0, v, \epsilon, 0) | v \in V_{init}\}$

$E_{fin} = \{(v, v_0, \epsilon, 0) | v \in V_{fin}\}$

For a given edge set E, let $org(E)$ denote the set of those edges (originals) in $ed(G')$ of which E comprises a copy. In the second step, one constructs a strongly connected symmetric augmentation G'' of $E^{trg} \uplus \{e_0\}$ in $ed(G') \setminus \{e_0\}$, as follows:

1. Initialize G'' to a minimum-cost symmetric augmentation of $E^{trg} \uplus \{e_0\}$ in $ed(G') \setminus \{e_0\}$. This is the most demanding step of the procedure, but can be accomplished in polynomial time [13].
2. While G'' has more than one component, merge two of its components, a G_1 and a G_2, as follows:
   a) In G, select a minimum-cost cycle w_c with $((vr(w_c) \cap vr(G_1)) \neq \emptyset) \wedge ((vr(w_c) \cap vr(G_2)) \neq \emptyset)$ and $G[(org(ed(G'')) \cup set(w_c)) \cap E^{acyc}]$ acyclic.
   b) Enhance G'' with a new copy of every edge in w_c.

In the last step, one constructs in G'' a walk starting and ending in v_0 and traversing every edge in $ed(G'')$ exactly ones. Construction of such an *Euler tour*

is trivial. The tour is a copy of a walk $e_0 \cdot e_1 \cdot w \cdot e_2$ of G' with e_1 in E_{init}, e_2 in E_{fin} and w, as we now prove, a CRTS walk.

Theorem 6. The above procedure is sound.

Proof. The initial version of G'' satisfies the following:

(1) Every edge is a copy of an edge in $ed(G')$.

(2) The directed (multi)graph is symmetric.

(3) Exactly one edge is a copy of e_0. Hence, exactly one edge is a copy of an edge in E_{init} and exactly one edge is a copy of an edge in E_{fin}, for otherwise the directed (multi)graph would not be symmetric.

(4) For every edge in E^{trg}, there is at least one copy.

(5) $G[org(ed(G'')) \cap E^{acyc}]$ is acyclic, because for any cycle in the digraph, of cost more than zero, G'' would have a copy whose edges could be removed from $ed(G'')$ to contradictorily obtain an even cheaper symmetric augmentation of $E^{trg} \uplus \{e_0\}$ in $ed(G') \setminus \{e_0\}$.

In any subsequent merging of two components, with a copy of a cycle w_c of G, all the five properties of G'' are preserved, because no edge is removed, every new edge is a copy of an edge in $ed(G)$, the new edges form a cycle and $G[(org(ed(G'')) \cup set(w_c)) \cap E^{acyc}]$ is acyclic. Hence, the final version of G'' also possesses the five properties, which makes w a CRTS walk. □

12. Discussion and conclusions

In the paper, we re-engineered an established approach to checking sequence construction with global optimization. By the re-engineering, we restored its original power, which has recently been diminished by a non-optimal correction of

detected flaws. With the new method, one can, hence, exploit test overlapping and alternative kinds of transition implementation tests at least as much as originally in the methods of [2–11]. Experimental studies on how much this can contribute to the length optimization of the checking sequence are available in [10, 14]. The chances strongly depend on the choice of the candidate transition implementation tests, whose key part are the employed state recognizers. Each of the studies targets a specific promising kind of state recognizers allowed by the methods of [2–11].

The new method is a semantic generalization of the methods and, unlike them, open to new kinds of transition implementation tests, possibly based on new kinds of state recognition patterns [15]. It can readily consider as candidate checking sequence segments also promising segments of checking sequences generated by other methods, such as, for example, the so called incremental methods [16–19]. Efficient exploitation of the new options is for further study.

Another virtue of the new method is that in its every step, one is forced to be fully aware of what one is doing and for what purpose. Most importantly, it commands explicit consideration of any dependencies between the considered candidate transition implementation tests. In the methods of [2–11], this is not always the case, which is why the problematic ones, if uncorrected, sometimes generate a sequence which combines tests in a cyclic dependency relation and, hence, cannot be trusted [12].

Finally, the new method can be regarded as a template for new simple-to-use more specific methods, for note that not everybody is skilled enough to employ it directly. The identification of its specializations of practical interest other than analogues of the methods of [2–11] is another task for the future.

References

[1] A.T. Endo, A. Simão, Evaluating test suite characteristics, cost and effectiveness of DFSM-based testing methods, Inf. Soft. Tech. 55(6) (June 2013) 1045-1062.

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

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

[4] 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.

[5] 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.

[6] R. M. Hierons, H. Ural, Optimizing the length of checking sequences, IEEE Trans. Comput. 55(5) (May 2006) 618-629.

[7] 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.

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

[9] R. M. Hierons, G.-V. Jourdan, H. Ural, H. Yenigün, Using adaptive distinguishing sequences in checking sequences, Proc. ACM Symp. Applied Computing, pp. 682-687, March 2008.

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

[11] R.M. Hierons, H. Ural, Generating a checking sequence with a minimum number of reset transitions, Aut. Soft. Eng. 17(3) (Sept. 2010) 217-250.

[12] M. Kapus-Kolar, On "Exploring alternatives for transition verification", J. Syst. Soft. 85(8) (Aug. 2012) 1744-1748.

[13] A.V. Aho, A.T. Dahbura, D. Lee, M.U. Uyar, An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours, IEEE Trans. Comm. 39(11) (1991) 1604-1615.

[14] G.-V. Jourdan, H.Ural, H. Yenigün, J.C. Zhang, Lower bounds on lengths of checking sequences, Form. Asp. Comp. 12(6) (Nov. 2010) 667-679.

[15] M. Kapus-Kolar, New state-recognition patterns for conformance testing of finite state machine implementations, Comput. Stand. Int. 34(4) (June 2012) 390-395.

[16] A. Simão, A. Petrenko, Generating checking sequences for partial reduced finite state machines, Proc. IFIP Int'l Con. Testing of Software and Communicating Systems, pp. 153-168, June 2008.

[17] A. Simão, A. Petrenko, Checking sequence generation using state distin-

guishing subsequences, Proc. IEEE Int'l Workshops Software Testing, Verification, and Validation, pp. 48-56, April 2009.

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

[19] A. Petrenko, A. Simão, N. Yevtushenko, Generating checking sequences for nondeterministic finite state machines, Proc. IEEE Int'l Conf. Software Testing, Verification and Validation, pp. 310-319, April 2012.