Microprocessors and Microsystems 38 (2014) 208–215

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

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

ARTICLE INFO

Article history:
Available online 6 February 2014

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

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.

© 2014 Elsevier B.V. All rights reserved.

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 2–11, 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 4–11, however, starts with an informal summary, as

* Tel.: +386 1 477 35 31.
E-mail address: monika.kapus-kolar@ijs.si

http://dx.doi.org/10.1016/j.micpro.2014.01.007
0141-9331/© 2014 Elsevier B.V. All rights reserved.

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 $\bar{o} = o_1, \ldots, o_k$, let $seg(\bar{o})$ denote the set of all its segments $o_i, \ldots, o_j$ with $1 \leqslant i \leqslant j \leqslant k$ and $set(\bar{o})$ the set of the objects present in the (multi)-set $\{o_1, \ldots, o_k\}$. For two sequences $\bar{o} = o_1, \ldots, o_k$ and $\bar{o}' = o'_1, \ldots, o'_{k'}$, let $\bar{o} \cdot \bar{o}'$ denote their concatenation $o_1, \ldots, o_k o'_1, \ldots, o'_{k'}$. 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 $w = e_1, \ldots, e_k$, let $init(w)$ denote its initial vertex $init(e_1)$, $fin(w)$ its final vertex $fin(e_k)$, $vr(w)$ its vertex set $\cup_{1 \leqslant i \leqslant k} \{init(e_i), fin(e_i)\}$ and $cost(w)$ its cost $\Sigma_{1 \leqslant i \leqslant k} cost(e_i)$. 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 $e'_1, \ldots, e'_k$ with e'_i for every $1 \leqslant i \leqslant k$ 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 $\bar{x} = x_1, \ldots, x_k$ is $\Sigma_{1 \leqslant i \leqslant k} cost(x_i)$.

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 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 $init(\tau) = init(Q), \tau$ is *rooted*. If Q has no other transition sequence τ' with $(ios(\tau') = ios(\tau)) \wedge (fin(\tau') = fin(\tau)), \tau$ 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^i_x 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^1_a, i.e. $(s_1, s_5, a/0)$, is invertible, whereas the transition sequence $t^1_b t^1_a$ 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 $ts(b/0a/1) = t^2_b t^3_a$, $a/0b/0$ 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 $D_5 = a/2$. $nxt(t^1_a t^5_b) = \{t^1_a, t^2_a, t^3_a, t^4_a, t^5_a, t^2_b, t^5_b\}$.

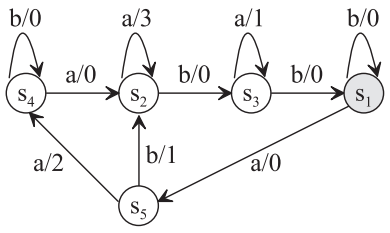

Fig. 1. An example M.

Download English Version:

https://daneshyari.com/en/article/460951

Download Persian Version:

https://daneshyari.com/article/460951

Daneshyari.com