Testing Components of Interacting Timed Finite State Machines

Maxim Gromov, Aleksandr Tvardovskii, Nina Yevtushenko
Tomsk State University, Tomsk, Russia
maxim.leo.gromov@gmail.com, tvardal@mail.ru, nyevtush@gmail.com

Abstract

In this paper, we address the problem of deriving test suites for checking components of interacting finite state machines with timed guards (TFSMs). Given a component TFSM, a corresponding test is derived for the composition of TFSMs under the assumption that all other components are fault-free.

1. Introduction

A big body of current research work is devoted to test derivation based on timed finite transition systems [see, for example, 1-10]. When deriving a test suite with the guaranteed fault coverage, the test derivation problem is usually reduced to generating tests against an appropriate Finite State Machine (FSM) model [1-2], for which such test derivation methods exist. Moreover, since complex systems are usually compositions of some simpler components, tests should be derived for a composition of interacting machines.

In this paper, we consider FSMs with time guards and output delays (TFSMs). FSM abstractions for TFSMs are proposed in [9] and a technique for deriving a composed machine for a system of interacting TFSMs is elaborated in [10]. Correspondingly, when deriving tests with the guaranteed fault coverage for TFSM compositions, the existing FSM based methods can be applied [11, 12]. In order to minimize the length of test suites while preserving their high quality, tests are iteratively derived for each component TFSM under the assumption that all other components are fault-free [13, 14]. In other words, we consider the composition of two TFSMs: one component TFSM is assumed to be fault-free while the other component TFSM should be tested. A test suite with the guaranteed fault coverage is derived for the composition with respect to the equivalence relation and then is minimized while preserving the fault coverage for the component under test.

Why not to test a component TFSM in isolation? If a component is tested in isolation, conforming component implementations can be claimed as nonconforming, since the equivalence relation is too strong for testing components of interacting machines [12]. We need to consider a so-called external equivalence relation and there are no methods for deriving tests with the guaranteed fault coverage with respect to such relation. Moreover, it can happen that for a component TFSM there is no direct access to its input and/or output. For this reason, similar to testing in context, we still use a composed TFSM for test derivation and discuss how a test suite can be minimized when the context is assumed to be fault-free.

The contributions of the paper can be summarized as follows:

- Given a system of two interacting TFSMs, the composed TFSM is constructed. A test suite is derived for the FSM abstraction of the composed machine and converted into a test suite that contains input and output timed sequences.
- We show how this test suite can be minimized while preserving the test coverage for a component TFSM under test.

The structure of the paper is as follows. Section 2 contains the preliminaries for FSMs with time guards (TFSMs) and the method for deriving a composed FSM is briefly described followed by a small illustration example. In Section 3, we discuss how test suites for an embedded component can be derived, and Section 4 concludes the paper.

2. Preliminaries

In this paper, we consider initialized FSMs which correspond to sequential systems with a reliable reset input [15]. An FSM $\mathtt{S}$ has a finite non-empty set S of states with the designated initial state. The set I is the set of *inputs* (impacts) and set O is the set of *outputs* (responses). The *behavior* (*transition*) relation represents transitions of the system. Given a transition (s, i, o, s') and FSM $\mathtt{S}$ at state s, the FSM moves to state s' and produces the output response o when an input i is applied. A *trace* of the FSM is a sequence of consecutive input/output pairs starting at the initial state.

A *timed* FSM (TFSM) is an FSM annotated with a *clock* [9, 10] and time guards associated with each transition. The reset operation resets the value of clock variable t to zero when an input is submitted

978-1-5090-0693-9/16/$31.00 ©2016 IEEE

and after the output is produced. An *input (time) guard* g_i describes the time domain when a transition can be executed. A time guard is an interval $\lceil min, max \rceil$, $\lceil \in \{(, [\}, \rceil \in \{,),]\}$, where *min* is a non-negative integer while *max* is the infinity or a non-negative integer such that $min \leq max$. An *output delay* describes a transition time: it is a non-negative integer that represents the number of ticks needed for producing an output after an input is applied. Consider a transition $(s, i, o, s', g_i, d) \in S \times I \times O \times S \times \Pi \times Z$. This transition describes the situation when TFSM S being at state s accepts input i applied at time $t \in g$ measured from the moment TFSM S entered state s. The TFSM S produces output o after d ticks counted from the moment when the input has been applied, moves to state s' and the clock is set to zero.

Given a TFSM $S = (S, I, O, \lambda_S, s_0)$, a pair (i, t) where $i \in I$ and t is real, is a *timed input* that indicates that an input i is applied to TFSM S at time instance t. A *timed output* is defined in the same way. A sequence of timed inputs $(i_1, t_1) \ldots (i_l, t_l)$ is a *timed input* sequence, a sequence of timed outputs $(o_1, t'_1) \ldots (o_l, t'_l)$ is a *timed output* sequence. A sequence $\alpha = (i_1, t_1)/(o_1, t'_1) \ldots (i_l, t_l)/(o_l, t'_l)$ of pairs of timed inputs and outputs is a *timed trace* of TFSM S at state s if the TFSM has a sequence of transitions $(s_j, i_j, o_j, s_{j+1}, g_i, d_j)$ such that $s_1 = s$ and for each $j = 1, \ldots, l$, it holds that $t_j \in g_j$. TFSM traces can be adequately described using a classical FSM that has timed inputs $(i, 0)$, $(i, (0, 1))$, …, (i, B), $(i, (B, \infty))$ where B is the largest finite boundary of time guards [9]. If B is rather small, for example, equals two or three as it happens for real time systems, then the number of inputs is not dramatically increased and FSM based test derivation methods can be used for deriving test suites against TFSMs.

In this paper, we consider complete and deterministic TFSMs, i.e., for each state s, input i and value t of the clock there exists exactly one transition (s, i, o, s', g, d) such that $t \in g$. Two complete deterministic TFSMs are *equivalent* if they have the same behavior, i.e., the TFSMs have the same sets of timed traces at the initial states.

Test cases are timed input sequences derived from the given TFSM specification to determine whether an implementation TFSM under test conforms to the specification. In this paper, an implementation TFSM *conforms* to the specification if it is equivalent to the specification TFSM, i.e., the implementation TFSM has the expected output response to each timed input sequence. If the observed output is unexpected or it is not produced at a specified time instance, then the implementation has a fault, i.e., it is a *faulty* or a *nonconforming* implementation.

In this section, we briefly describe an algorithm for deriving a composed machine for the parallel composition of two TFSMs [10]. For the sake of simplicity, we assume that only one component (*Context*) has external inputs and outputs (Figure 1). The machines communicate in a dialogue mode by exchanging internal actions (messages). The next external input can be applied only after the composition has produced an external output to the previous external input. If an infinite dialogue occurs between components then there is a *live-lock* [13-15] that is usually considered as a design error.

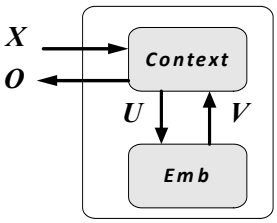
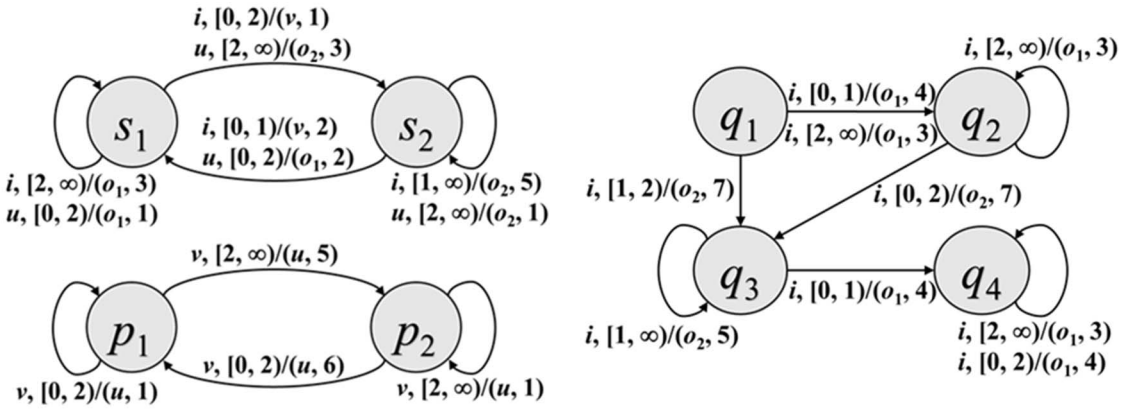


Figure 1. Composition of two TFSMs

States of the composition are 4-tuples: two items represent states of the component TFSMs while the pair of timed notations indicates how long a corresponding component is at the current state when an input is applied. The initial pair of states is annotated with the time instances (0, 0). For each external timed input, we construct a sequence of consecutive transitions which occur in the composition until an external output is produced. Once an external output is obtained, all the time delays of the transition sequence are summarized and this sum is the output delay for the corresponding transition. As an example, consider the composition of two TFSMs S (*Context*) and P (*Emb*) in Figure 2a [10]; the composed TFSM is shown in Figure 2b.

Consider the initial state $(s_1, 0, p_1, 0)$ and a transition $(s_1, (i, 1), (v, 1), s_2)$. When a timed input $(i, 1)$ is applied to S, the component TFSM P is at state $(p_1, 1)$. The TFSM S produces v after one tick, reaches state $(s_2, 0)$ and the timed input $(v, 1)$ is applied to the component TFSM P at state $(p_1, 1)$. The TFSM P executes the transition $(p_1, (v, 2), (u, 5), p_2)$ and the input $(u_1, 5)$ is applied to S at state $(s_2, 0)$. The component TFSM S executes the transition $(s_2, (u, 5), (o_2, 1), s_2)$ and produces the external output $(o_2, 7)$ where $7 = 1 + 5 + 1$ is the sum of all delays occurred in the composition after applying the external input $(i, 1)$. Therefore, after producing the external output $(o_2, 7)$ the composition reaches state $(s_2, 0, p_2, 1)$. Correspondingly, the composed TFSM has a transition $(q_1, (i, 1), (o_2, 7), q_3)$ where $q_1 = (s_1, 0, p_1, 0)$ and $q_3 = (s_2, 0, p_2, 1)$.



a) **b)**

Figure 2. a) Component TFSMs S (context) and P (the embedded component); b) The composed TFSM

3. Test suites with the guaranteed fault coverage for the embedded component

As the composed machine is a TFSM, conformance tests can be derived in the following ways.

1. If the number of faults in the component TFSM under test is not big then all possible component implementations can be explicitly enumerated (white box approach). For each component implementation, the composed TFSM with other components is derived. When deriving a composed TFSM it can happen that the composition falls into a live-lock. In this case, an input sequence that causes a live-lock is added to the test suite. If the composed TFSM is complete and deterministic then a distinguishing sequence for two composed TFSMs is constructed and the sequence is added to the test suite (if such a sequence exists). A distinguishing sequence can be derived based on the FSM abstractions of composed TFSMs or directly based on the product of two composed TFSMs.

2. If the number of component faults is rather big then black-box test derivation methods can be applied [11, 12]. The only problem is that in this case, the information that the context is fault-free is not taken into account. Correspondingly, some FSM based methods for minimizing a test suite should be utilized. First, all the input sequences which do not involve the interaction with the component under test can be eliminated as they use only the fault-free context. Secondly, given an external test case, an automata representation for the set of internal traces (traces of the embedded component) that are detected by this test case can be constructed. At the next step, a minimal set of test cases that detect each faulty internal trace can be constructed.

3. Another option is to consider an implementation system as a grey box where only a component TFSM under test is unknown. In this case, a *mutation machine* can be constructed for the FSM abstraction of the composed TFSM and a number of methods for test derivation against a mutation machine (or a fault function) can be applied which return tests with the desirable fault coverage. For example, if we assume that all the transitions in the embedded component are processed slower than prescribed by its specification, then the FSM abstraction of the specification TFSM has to be distinguished from FSM abstractions of all TFSMs where at some transition the output is produced later than in the specification TFSM. Such faults can be detected by a transition tour of the specification TFSM.

We have experimented with a transition tour of a composed system of two TFSMs using their microcontroller implementations. According to our experiments, many hardware faults of the embedded component can be detected with such test suite; this fact has been checked by inserting appropriate faults into the embedded component TFSM.

4. Conclusions

In this paper, we briefly discussed how a test suite with the guaranteed fault coverage can be reduced for a system of interacting TFSMs when only some components can be faulty. Since for each TFSM there exists an FSM abstraction, FSM based test derivation methods can be used for this purpose. In order to minimize a test suite and still preserve the high fault coverage, a test suite can be derived with respect to single component faults, i.e., we can assume that only one component TFSM can be faulty. Some components can be more reliable than others and then only tests for unreliable components

need to be taken into account. When using a white box testing approach, all the mutants of a component of our interest are generated and a distinguishing sequence for the specification and mutant composed TFSMs is constructed. When talking about black box test derivation approaches, an initial test suite can be derived for the composed timed machine. At the next step, this test suite can be optimized by deleting test cases which detect same faulty implementations of a component of our interest. An interesting way is to use an FSM abstraction of a mutation composed TFSM; to the best of our knowledge, there are no results related to mutation TFSMs. Experimental results clearly show that a test suite derived as a transition tour of the composed system of interacting TFSMs can detect many hardware faults of the embedded component. However, when deriving a composed TFSM, the state explosion problem can occur and it would be interesting to study the opportunity for deriving tests with the guaranteed fault coverage for interacting TFSMs without the explicit construction of the composed TFSM.

5. Acknowledgement

This work is partly supported by RSF Project No. 16-49-03012.

6. References

[1] Springintveld, J., Vaandrager, F., D'Argenio, P. Testing Timed Automata. *Theoretical Computer Science*, vol. 254, 2001, issues 1-2, pp. 225-257.

[2] En-Nouary, A., Dssouli, R., Khendek, F. Timed Wp-Method: Testing Real-Time Systems. *IEEE Transactions on Software Engineering*, 2002, vol. 28, issue 11, pp. 1023-1038.

[3] Larsen, K.G., Mikucionis, M., Nielsen, B., Skou, A. Testing real-time embedded software using UPPAAL-TRON: an industrial case study. *Proceedings of the 5th ACM international conference on Embedded software*, 2005, pp. 299-306.

[4] Rütz, C., Schmaltz, J. An experience report on an industrial case-study about timed model-based testing with UPPAAL-TRON. *Proceedings of the IEEE 4th International Conference on Software Testing, Verification and Validation Workshops*, 2011, pp. 39-46.

[5] Merayo, M.G., Nunez, M., Rodriguez, I. Formal Testing from Timed Finite State Machines. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2008, vol. 52, issue 2, pp. 432-460.

[6] Schmaltz, J., Tretmans, J. On conformance testing for timed systems. *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems*, 2008, pp. 250-264.

[7] Krichen, M., Tripakis, S. Conformance testing for real-time systems. *Formal Methods in System Design*, 2008, vol. 34, issue 3, pp. 238-304.

[8] El-Fakih, K., Yevtushenko, N., Simão, A. A practical approach for testing timed deterministic finite state machines with single clock. *Science of Computer Programming*, 2014, vol. 80, pp. 343-355.

[9] Bresolin, D., El-Fakih, K., Villa, T., Yevtushenko, N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. *Proceedings of the 7th International Symposium on Games, Automata, Logics and Formal Verification*, 2014, pp. 203-216.

[10] Gromov, M., Tvardovskii A., Yevtushenko N. Testing Systems of interacting Timed Finite State Machines with the Guaranteed Fault Coverage. *Proceedings of the 17th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices*, 2016, pp. 96-99.

[11] Chow, T. S. Testing software Design Modeled by Finite State Machines. *IEEE Transactions on Software Engineering*, 1978, vol. 4, issue 3, pp. 178-187.

[12] Dorofeeva, R., El-Fakih, K., Maag, S., Cavalli, A., Yevtushenko N. FSM-based conformance testing methods: A survey annotated with experimental evaluation. *Information & Software Technology*, 2010, vol. 52, issue 12, pp. 1286-1297.

[13] Petrenko, A., Yevtushenko, N., Bochmann, G. v., Dssouli R. Testing in context: framework and test derivation. *Computer communications*, vol. 19, pp. 1236-1249, 1996.

[14] Lima, L.P., Cavalli, A.R. A pragmatic approach to generating test sequences for embedded systems. *Proceedings of the 10th International Workshop on Protocol Conformance Testing*, 1997, pp. 125-140.

[15] Villa, T., Yevtushenko, N., Brayton, K.R., Mishchenko, A., Petrenko, A., Sangiovanni-Vincentelli, A. The Unknown Component Problem: Theory and Applications, Springer, 2012, 313 p.