On the verification of strictly deterministic behavior of Timed Finite State Machines

E.M. Vinarskii <vinevg2015@gmail.com>
V.A. Zakharov <zakh@cs.msu.su>
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia

Abstract. Finite State Machines (FSMs) are widely used as formal models for solving numerous tasks in software engineering, VLSI design, development of telecommunication systems, etc. To describe the behavior of a real-time system one could supply FSM model with clocks — a continuous time parameters with real values. In a Timed FSM (TFSM) inputs and outputs have timestamps, and each transition is equipped with a timed guard and an output delay to indicate time interval when the transition is active and how much time does it take to produce an output. A variety of algorithms for equivalence checking, minimization and test generation were developed for TFSMs in many papers. A distinguishing feature of TFSMs studied in these papers is that the order in which output letters occur in an output timed word does not depend on their timestamps. We think that such behavior of a TFSM is not realistic from the point of view of an outside observer. In this paper we consider a more advanced and adequate TFSM functioning; in our model the order in which outputs become visible to an outsider is determined not only by the order of inputs, but also by de lays required for their processing. When the same sequence of transitions is performed by a TFSM modified in a such way, the same outputs may follow in different order depending on the time when corresponding inputs become available to the machine. A TFSM is called strictly deterministic if every input timed word activates no more than one sequence of transitions (trace) and for any input timed word which activates this trace the letters in the output words always follows in the same order (but, maybe, with different timestamps). We studied the problem of checking whether a behavior of an improved model of TFSM is strictly deterministic. To this end we showed how to verify whether an arbitrary given trace in a TFSM is steady, i.e. preserves the same order of output letters for every input timed word which activates this trace. Further, having the criterion of trace steadiness, we developed an exhaustive algorithm for checking the property of strict determinacy of TFSMs. Exhaustive search in this case can hardly be avoided: we proved that determinacy checking problem for our model of TFSM is co-NP-hard.

Keywords: Timed Finite State Machines; strictly deterministic behavior

DOI: 10.15514/ISPRAS-2018-30(3)-22

For citation: Vinarskii E.M., Zakharov V.A. On the verification of strictly deterministic behaviour of Timed Finite State Machines. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 325-340. DOI: 10.15514/ISPRAS-2018-30(3)-22

1. Introduction

Finite State Machines (FSMs) are widely used as formal models for analysis and synthesis of information processing systems in software engineering, VLSI design, telecommunication, etc. The most attractive feature of this model of computation is its simplicity — many important synthesis and analysis problems (equivalence checking, minimization, test derivation, etc.) for classical FSMs can be solved in time which is almost linear or quadratic of the size of an FSM under consideration.

The concept of FSM is rather flexible. Since in many applications time aspects such as durations, delays, timeouts are very important, FSMs can be augmented with some additional features to describe the dependence of the behavior of a system on events occurring in real time. One of the most advanced timed extension of FSMs is the concept of Timed Automata which was developed and studied in [1]. Timed Automata are supplied with clocks (timers) for indicating real time moments, measuring durations of events, providing timeout effects. Transitions in such automata depends not only on the incoming of the outside messages and signals but also on the values of clocks. Further research showed that this model of computation is very expressive and captures many important features of real-time systems behavior. On the other side, Timed Automata in the full scope of their computing power are very hard for analysis and transformations. The reachability problem for Timed Automata is decidable [2], and, therefore, this model of computation is suitable for formal verification of real-time computer systems. But many other problems such as universality, inclusion, determinability, etc. are undecidable (see [2], [8]), and this hampers considerably formal analysis of Timed Automata.

When a Timed Automaton is capable to selectively reset timers, it can display rather sophisticated behavior which is very difficult for understanding and analysis. In some cases, such ability is very important; see, e.g. [9]. But a great deal of real-time programs and devices operate with timers much more simply: as soon as such a device switches to a new mode of operation (new state), it resets all timers. Timed Finite State Machines (TFSM) of this kind were studied in [5], [10], [13], [14]. TFSM has the only timer which it resets "automatically" as soon as it moves from one state to another. On the other hand, TFSMs, in contrast to Timed Automata introduced in [1], operate like transducers: they receive a sequence of input signals augmented with their timestamps (input timed word) and output a sequence of responses also labeled by timestamps (output timed word). The timestamps are real numbers which indicate the time when an input signal becomes available to a TFSM or an output response is generated. Transitions of a TFSM are equipped with time guards to indicate time intervals when transitions are active. Therefore, a reaction of a TFSM to an input signal depends not only on the signal but also on its timestamp. Some algorithms for equivalence checking, minimization and test generation were developed for TFSMs in [6], [5], [13], [14], [15]. It can be recognized that this model of TFSM combines a sufficient expressive power for modeling a wide class of real-time information processing systems and a developed algorithmic support.

As it was noticed above a behavior of a TFSM is characterized by a pair sequences: an input timed word and a corresponding output timed word. A distinguishing feature of TFSMs studied in [5], [10], [13], [14], [15] is that an output timed word is formed of timestamped output letters that follows in the same order as the corresponding input letters regardless of their timestamps. Meanwhile, suppose that a user of some file management system gives a command «Save» and immediately after that a command «Exit». Then if a file to be saved is small then the user will observe first a response «File is saved» and then a notification «File Management System is closed». But if a file has a considerable size then it takes a lot of time to close it. Therefore, it can happen that a user will detect first a notification «File Management System is closed» and then, some time later, he/she will be surprised to find an announcement «File is saved». Of course, the user may regard such behavior of the system enigmatic. But much worse if the order in which these notifications appear may vary in different sessions of the system. If a File Management System interacts with other service programs such an interaction will almost certainly lead to errors. However, if a behavior of TFSMs is defined as in the papers referred above then such a model can not adequately capture behavioral defects of real-time systems, similar to the one that was considered in the example.

To avoid this shortcoming of conventional TFSMs and to make their behavior more "realistic" from the point of view of an outside observer we offer some technical change to this model. We will assume that an output timed word consists of timestamped letters, and these letters always follow in ascending order of their timestamps regardless of an order in which the corresponding input letters entered a TFSM. In this model it may happen so that an input $\boldsymbol{b}$ follows an input $\boldsymbol{a}$ but a response to $\boldsymbol{b}$ appears before a response to $\boldsymbol{a}$ is computed. Clearly, the defect with File Management System discussed above becomes visible to an outside observer "through" the model of TFSMs thus modified.

At first sight, it may seem that this change only slightly complicates the analysis of the behavior of such models. But this is a false impression. In the initial model of TFSM the formation of an output timed word is carried out by local means for each state of the system. In our model this is a global task since to find the proper position of a timestamped output letter one should consider the run of TFSM as a whole. Therefore, even the problem of checking whether a behavior of an improved model of TFSM is deterministic can not be solved as easy and straightforwardly as in the case of the initial model of TFSM.

It should be noticed that the property of deterministic behavior is very important in theory real-time machines. As it was said above, universality, inclusion and equivalence checking problems are undecidable for Timed Automata in general case [2] but all these problems have been shown to be decidable for deterministic Timed Automata [3], [11]. However, testing whether a Timed Automaton is determinable has been proved undecidable [8]. Understanding and coping with these weaknesses have attracted lots of research, and classes of timed automata have been exhibited, that can be effectively determinized [3], [12]. A generic construction that is applicable

to every Timed Automaton, and which, under certain conditions, yields a deterministic Timed Automaton, which is language-equivalent to the original timed automaton, has been developed in [4].

We studied the determinacy checking problem for improved TFSMs and present the results of our research in this paper. First, we offer a criterion to determine whether a given sequence of transition (trace) in a TFSM is steady, i.e. for any input timed word which activates this trace the letters of output words always follow in the same order (but, maybe, with different timestamps). Then, using this criterion we developed an exhaustive algorithm for checking the property of strict determinacy of TFSMs. This property means that every input timed word activates no more than one trace and all traces in a TFSM are steady. Exhaustive search, although been time consuming, can hardly be avoided in this case: we proved that determinacy checking problem for improved version of TFSMs is co-NP-hard by polynomially reducing to its complement the subset-sum problem [7] which is known to be NP-complete.

The structure of the paper is as follows. In Section II we define the basic notions and introduce an improved concept of TFSM (or, it would be better said, a concept of TFSM with an improved behavior). In Section III we present necessary and sufficient conditions for steadiness of traces in a TFSM and show how to use this criterion to check whether a given TFSM is strictly deterministic. Section IV contains the results on the complexity of checking the properties of strictly deterministic behavior of TFSM. In the Conclusion we briefly outline the consequences of our results and topics for further research.

2. Formatting overview

Consider two non-empty finite alphabets I and O; the alphabet I is an *input alphabet* and the alphabet O is an *output alphabet*. The letters from I can be regarded as control signals received by some real-time computing system, whereas the letters from O may be viewed as responses (actions) generated by the system. A finite sequence $w = i_1, i_2, \dots, i_n$ of input letters is called an *input word*, whereas a sequence $z = o_1, o_2, \dots, o_n$ of output letters is called an *output word*. As usual, the time domain is represented by the set of non-negative reals $\mathbb{R}_0^+$. The set of all positive real numbers will be denoted by $\mathbb{R}^+$. When such a system receives a control signal (a letter i) its output depends not only on the input signal i but also on

- a current internal state of the system,
- a time instance when $\boldsymbol{i}$ becomes available to a system, and
- time required to process the input (output delay).

These aspects of real-time behavior can be formalized with the help of timestamps, time guards and delays. A timestamp as well as a delay is a real number from $\mathbb{R}^+$. A *timestamp* indicates a time instance when the system receives an input signal or generates a response to it. A *delay* is time the system needs to generate an output response after receiving an input signal. A *time guard* is an interval $g = \langle u, v \rangle$, where $\langle \in \{(, [\}, \rangle \in \{),]\}$, and u, v are timestamps such that $0 < u < v$. Time intervals

indicate the periods of time when transitions of a system are active for processing input signals. As usual, the term *time sequences* is reserved for an increasing sequence of timestamps. For the sake of simplicity we will deal only with time guards of the form $(u, v]$: all the results obtained in this paper can be adapted with minor changes to arbitrary time guards.

Let $\boldsymbol{w = x_1, x_2, \dots x_n}$ and $\boldsymbol{\tau = t_1, t_2, \dots, t_n}$ be an input (output) word and a time sequence, respectively, of the same length. Then a pair $\boldsymbol{(w, \tau)}$ is called a *timed word*. Every pair of corresponding elements $\boldsymbol{x_j}$ and $\boldsymbol{t_j}$, $\boldsymbol{1 \le j \le n}$, indicates that an input signal (or an output response) $\boldsymbol{x_j}$ appears at time instance $\boldsymbol{t_j}$. In order to make this correspondence clearer we will often write timed words as sequences of pairs $\boldsymbol{(w, \tau) = (i_1, t_1), (i_2, t_2), \dots, (i_n, t_n)}$ whose components are input signals (or output responses) and their timestamps.

A *Finite State Machine* (*FSM*) over the alphabets $\boldsymbol{I}$ and $\boldsymbol{O}$ is a triple $\boldsymbol{M = \langle S, s_{in}, \rho \rangle}$ where $\boldsymbol{S}$ is a finite non-empty set of *states*, $\boldsymbol{s_{in}}$ is an *initial state*, $\boldsymbol{\rho \subseteq (S \times I \times O \times S)}$ is a *transition* relation. A transition $\boldsymbol{(s, i, o, s')}$ means that FSM $\boldsymbol{M}$ when being at the state $\boldsymbol{s}$ and receiving an input signal $\boldsymbol{i}$ moves to the state $\boldsymbol{s'}$ and generates the output response $\boldsymbol{o}$.

FSMs can not measure time and, therefore, they are unsuitable for modeling the behavior of real-time systems. The authors of [1] proposed to equip FSMs with clocks — variables which take non-negative real values. To manipulate with clocks machines use reset instructions, timed guards and output delays. Time guards indicate time intervals when transitions are active for processing input signals. An output delay indicates how much time does it take to process an input. Thus, every transition in such a machine is a quadruple $\boldsymbol{\langle input, timed\ guard, output, delay \rangle}$. Input signals and output responses are accompanied by timestamps. If an *input* is marked by a timestamp which satisfies the *time guard* then the transition fires, the machine moves to the next state and generates the *output*. This output is marked by a timestamp which is equal to the timestamp of the input plus the *delay*. For real-time machines of this kind usual problems from automata theory (equivalence and containment checking, minimization, etc.) may be set up and solved. The minimization problem for real-time machines is very important, since the complexity of many analysis and synthesis algorithms depend on the size of machines. In [14] this problem was studied under the so called "slow environment assumption": next input becomes available only after an output response to the previous one is generated.

In this paper, we consider a more advanced real-time machine; in this model the order in which outputs become visible to an outside observer is determined not only by the order in which inputs follow, but also by the delay required for their processing. When the same sequence of transitions is performed by such a machine the same outputs may follow in different order depending on the arriving time of the corresponding inputs. Our main goal is to develop equivalence checking and minimization algorithms for real-time machines of this kind. But, as the results of Automata Theory show, these problems may have efficient solution only for deterministic machines.

Thus, our first step toward the solution of these problems is to find a way to check if the behavior of a machine is deterministic.

But there is also another reason to study the problem of checking the determinism of the behavior of real-time machines. Unlike traditional discrete models of computation, the behavior of real-time machines depends not only on the control signals as such, but also on the time of their arrival. However, the latter factor has a greater degree of uncertainty. In most cases, in practice, it is desirable to reduce the effect of this uncertainty to a minimum. Therefore, the determinacy checking problem for real-time machines can be considered as a special version of the verification problem — checking that the time factor does not have an unforeseen influence on the behavior of the system.

Formally, by Timed FSM (TFSM) over the alphabets $\boldsymbol{I}$ and $\boldsymbol{O}$ we mean a quadruple $\boldsymbol{M = (S, s_{in}, G, \rho)}$ where:

- $\boldsymbol{S}$ is a finite non-empty *set of states*,
- $\boldsymbol{s_{in}}$ is *an initial state*.
- $\boldsymbol{G}$ is a set of *timed guards*,
- $\boldsymbol{\rho \subseteq (S \times I \times O \times S \times G \times \mathbb{R}^+)}$ is a *transition relation*.

A transition (s, i, o, s', g, d) should be understood as follows. Suppose that TFSM receives the input letter i marked by a timestamp t when being at the state s. If the previous letter has been delivered to the TFSM at time $\hat{t}$ such that $\Delta t = t - \hat{t} \in g$ then the TFSM moves to the state s' and outputs the letter o marked with the timestamp $\tau = t + d$. When algorithmic and complexity issues of TFSM's analysis and synthesis are concerned then we assume that time guards and delays are rational numbers, and the size of a TFSM is the length of a binary string which encodes all transitions in the TFSM.

A *trace* $\boldsymbol{tr}$ in TFSM $\boldsymbol{M}$ is a sequence of transitions $\boldsymbol{(s_0, a_1, b_1, s_1, (u_1, v_1], d_1), \dots, (s_{n-1}, a_n, b_n, s_n, (u_n, v_n], d_n)}$, where every state $\boldsymbol{s_j}$, $\boldsymbol{0 < j < n}$, is an arrival state of one transition and a departure state of the next transition. We say that the trace $\boldsymbol{tr}$ *converts* an input timed word $\boldsymbol{\alpha = (a_1, t_1), (a_2, t_2), \dots, (a_n, t_n)}$ to the timed output word $\boldsymbol{\beta = (b_{j_1}, \tau_1), (b_{j_2}, \tau_2), \dots, (b_{j_n}, \tau_n)}$, iff

- $t_j - t_{j-1} \in (u_j, v_j]$ holds for all $j, 1 \le i \le n$ (it is assumed that $t_0 = 0$);
- β is such a permutation of the sequence $\gamma = (b_1, t_1 + d_1), (b_2, t_2 + d_2), \dots, (b_n, t_n + d_n)$ that the second components of the pairs $\tau_1, \tau_2, \dots, \tau_n$ constitute a time sequence.

Clearly, for every trace tr and an input timed word α its conversion β (if any) is determined uniquely; such a conversion will be denoted as $conv(tr, \alpha)$. If $conv(tr, \alpha)$ is defined then we say that the input timed word α *activates* the trace

tr. We will say that the output word $b_{j_1}, b_{j_2}, \dots, b_{j_n}$ is a *plain response* to the input timed word α on the trace tr; it will be denoted as $resp(tr, \alpha)$.

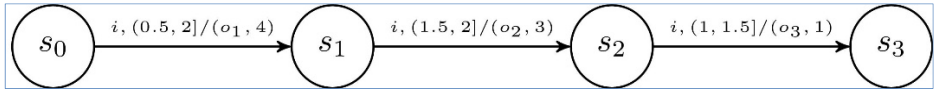

Fig.1 TFSM $\boldsymbol{M}$

Consider, for example, a TFSM $\boldsymbol{M}$ depicted in Fig. 1 and a trace

$$\boldsymbol{tr} = (\boldsymbol{s_0}, \boldsymbol{i}, \boldsymbol{s_1}, \boldsymbol{o_1}, (\mathbf{0.5}, \mathbf{2}], \mathbf{4}), (\boldsymbol{s_1}, \boldsymbol{i}, \boldsymbol{s_2}, \boldsymbol{o_2}, (\mathbf{1.5}, \mathbf{2}], \mathbf{3}), (\boldsymbol{s_2}, \boldsymbol{i}, \boldsymbol{s_3}, \boldsymbol{o_3}, (\mathbf{1}, \mathbf{1.5}], \mathbf{1})$$

in this TFSM. Then this trace

1. accepts an input timed word $\alpha_1 = (i, 1), (i, 2.7), (i, 4.1)$ and converts it to the output timed word $\beta_1 = (o_1, 5), (o_3, 5.1), (o_2, 5.7)$; thus, the plain response of M to α_1 is $w_1 = o_1, o_3, o_2$;
2. accepts an input timed word $\alpha_2 = (i, 1.5), (i, 3.2), (i, 4.3)$ and converts it to the output timed word $\beta_2 = (o_3, 5.3), (o_1, 5.5), (o_2, 6.2)$, and the plain response of M to α_2 is $w_2 = o_3, o_1, o_2$ which is different from w_1;

does not accept an input timed word $\alpha_3 = (i, 2.3), (i, 4), (i, 6)$.

3. Steady traces and strictly deterministic TFSMs

As can be seen from the above example, a pair of input timed words that differ only in timestamps of input signals may activate the same trace in a TFSM, although plain responses of TFSM to these words are different. Generally speaking, there is nothing unusual in this: in real-time models not only the input signals, but also the values of timers influence a run of a model. Nevertheless, in many applications it is critically important to be sure that the behavior of a real-time system is predictable: once a system choose a mode of computation (i.e. a trace in TFSM) it will behave in a similar way (i.e. give the same plain response) in all computations of this mode. Traditionally, computer systems in which for any input data the processing mode is uniquely determined by the system are called deterministic. But for our model of real-time systems this requirement should be clarified and strengthened. For this purpose, we introduce the notion of steady traces and the property of strict determinacy of a real-time system.

A trace $\boldsymbol{tr}$ in TFSM $\boldsymbol{M}$ is called *steady* if $\boldsymbol{resp}(\boldsymbol{tr}, \boldsymbol{\alpha_1}) = \boldsymbol{resp}(\boldsymbol{tr}, \boldsymbol{\alpha_2})$ holds for every pair of input timed words $\boldsymbol{\alpha_1}$ and $\boldsymbol{\alpha_2}$ that activate $\boldsymbol{tr}$. Thus, the order of the output letters generated by a steady trace does not depend on the small deviations of the timestamps of the input signals. A TFSM $\boldsymbol{M} = (\boldsymbol{S}, \boldsymbol{s_{in}}, \boldsymbol{G}, \boldsymbol{\rho})$ is called *deterministic* iff for every pair of transitions $(\boldsymbol{s}, \boldsymbol{i_1}, \boldsymbol{o_1}, \boldsymbol{s'}, (\boldsymbol{u_1}, \boldsymbol{v_1}], \boldsymbol{d_1})$ and $(\boldsymbol{s}, \boldsymbol{i_2}, \boldsymbol{o_2}, \boldsymbol{s''}, (\boldsymbol{u_2}, \boldsymbol{v_2}], \boldsymbol{d_2})$ in $\boldsymbol{\rho}$ either $\boldsymbol{i_1} \neq \boldsymbol{i_2}$, or $(\boldsymbol{u_1}, \boldsymbol{v_1}] \cap (\boldsymbol{u_2}, \boldsymbol{v_2}] = \emptyset$. This requirement means that every timestamped input letter can activate no more than one transition from an arbitrary given state $\boldsymbol{s}$. It also implies that every input timed word

can activate no more than one trace in $\boldsymbol{M}$. A deterministic TFSM is called *strictly deterministic* iff every initial trace in $\boldsymbol{M}$ which starts from the initial state $\boldsymbol{s_{in}}$ is steady. It is easy to see that TFSM, depicted in Fig. 1, is not strictly deterministic.

The Strict Determinacy Checking Problem (in what follows, SDCP) is that of checking, given a TFSM, if it is strictly deterministic. It is easy to check whether a TFSM is deterministic by considering one by one all pairs of transitions that emerge from the same state. But local means alone are not enough to check whether a given trace in a TFSM is steady. A simple criterion for steadiness of traces is presented as a Theorem below.

Let a sequence of transitions

$$(\boldsymbol{s_0}, \boldsymbol{i_1}, \boldsymbol{s_1}, \boldsymbol{o_1}, \langle \boldsymbol{u_1}, \boldsymbol{v_1} \rangle, \boldsymbol{d_1}), \dots, (\boldsymbol{s_{n-1}}, \boldsymbol{i_n}, \boldsymbol{s_n}, \boldsymbol{o_n}, \langle \boldsymbol{u_n}, \boldsymbol{v_n} \rangle, \boldsymbol{d_n})$$

be a trace $\boldsymbol{tr}$ in a TFSM $\boldsymbol{M}$. Then the following theorem holds.

Theorem 1. *A trace* $\boldsymbol{tr}$ *is steady iff for all pairs of integers* $\boldsymbol{k}, \boldsymbol{m}$ *such that* $\mathbf{1} \leq \boldsymbol{k} < \boldsymbol{m} \leq \boldsymbol{n}$ *at least one of the two inequalities* $\boldsymbol{d_k} - \boldsymbol{d_m} \leq \sum_{j=k+1}^{m} \boldsymbol{u_j}$ *or* $\boldsymbol{d_k} - \boldsymbol{d_m} > \sum_{j=k+1}^{m} \boldsymbol{v_j}$ *holds.*

Proof. (⇒) Suppose that there exists a pair $\boldsymbol{k}, \boldsymbol{m}$ such that $\mathbf{1} \leq \boldsymbol{k} < \boldsymbol{m} \leq \boldsymbol{n}$, and a double inequality holds:

$$\sum_{j=k+1}^{m} \boldsymbol{u_j} < \boldsymbol{d_k} - \boldsymbol{d_m} \leq \sum_{j=k+1}^{m} \boldsymbol{v_j}\,.$$

Then we use two positive numbers $\boldsymbol{r} = \boldsymbol{d_k} - \boldsymbol{d_m} - \sum_{j=k+1}^{m} \boldsymbol{u_j}$ and $\boldsymbol{\varepsilon} = \frac{\boldsymbol{r}}{\boldsymbol{n}}$ and consider a behaviour of a TFSM $\boldsymbol{M}$ in the input timed words

$$\boldsymbol{\alpha'} = (\boldsymbol{i_1}, \boldsymbol{v_1}), \dots, (\boldsymbol{i_k}, \sum_{j=1}^{k} \boldsymbol{v_j}), (\boldsymbol{i_{k+1}}, \sum_{j=1}^{k} \boldsymbol{v_j} + \boldsymbol{u_{k+1}} + \boldsymbol{\varepsilon}), \dots, (\boldsymbol{i_m}, \sum_{j=1}^{k} \boldsymbol{v_j} + \sum_{j=k+1}^{m} \boldsymbol{u_j} + \boldsymbol{\varepsilon}),$$

$$\boldsymbol{\alpha''} = (\boldsymbol{i_1}, \boldsymbol{v_1}), \dots, (\boldsymbol{i_k}, \sum_{j=1}^{k} \boldsymbol{v_j}), (\boldsymbol{i_{k+1}}, \sum_{j=1}^{k+1} \boldsymbol{v_j}), \dots, (\boldsymbol{i_m}, \sum_{j=1}^{m} \boldsymbol{v_j}).$$

It is easy to see that both words activate $\boldsymbol{tr}$.

The trace $\boldsymbol{tr}$ converts the timed input word $\boldsymbol{\alpha_1}$ to the timed output word

$$\boldsymbol{conv}(\boldsymbol{tr}, \boldsymbol{\alpha'}) = \dots, (\boldsymbol{o_m}, \boldsymbol{T'_m}), \dots, (\boldsymbol{o_k}, \boldsymbol{T'_k}), \dots$$

such that $\boldsymbol{T'_m} = \sum_{j=1}^{k} \boldsymbol{v_j} + \sum_{j=k+1}^{m} (\boldsymbol{u_j} + \boldsymbol{\varepsilon}) + \boldsymbol{d_m}$, and $\boldsymbol{T'_k} = \sum_{j=1}^{k} \boldsymbol{v_j} + \boldsymbol{d_k}$. In this timed output word, the output letter $\boldsymbol{o_k}$ follows the output letter $\boldsymbol{o_m}$ since

$$\boldsymbol{T'_k} - \boldsymbol{T'_m} = \boldsymbol{d_k} - \boldsymbol{d_m} - \sum_{j=k+1}^{m} \boldsymbol{u_j} + (\boldsymbol{m} - \boldsymbol{k})\boldsymbol{\varepsilon} = \boldsymbol{r} - \frac{\boldsymbol{r}(\boldsymbol{m} - \boldsymbol{k})}{\boldsymbol{n}} > \mathbf{0}.$$

Hence, $\boldsymbol{resp}(\boldsymbol{tr}, \boldsymbol{\alpha'}) = \dots, \boldsymbol{o_m}, \dots, \boldsymbol{o_k}, \dots$.

On the other hand, the trace $\boldsymbol{tr}$ converts the timed input word $\boldsymbol{\alpha''}$ to the timed output word

$$\boldsymbol{conv(tr, \alpha'') = \cdots, (o_k, T''_k), \ldots, (o_m, T''_m), \ldots}$$

such that $\boldsymbol{T''_k = \sum_{j=1}^{k} v_j + d_k}$ and $\boldsymbol{T''_m = \sum_{j=1}^{m} v_j + d_m}$. In this timed output word the output letter $\boldsymbol{o_m}$ follows the output letter $\boldsymbol{o_k}$ since

$$\boldsymbol{T''_m - T''_k = d_m - d_k = \sum_{j=k+1}^{m} v_j \geq 0}$$

Therefore, $\boldsymbol{resp(tr, \alpha'') = \cdots, o_k, \ldots, o_m, \ldots}$.

Thus, we got evidence that the trace $\boldsymbol{tr}$ is not steady.

($\Leftarrow$) Suppose that the trace $\boldsymbol{tr}$ is not steady. Then there exists a pair of timed input words $\boldsymbol{\alpha' = (i_1, t'_1), \ldots, (i_n, t'_n)}$ and $\boldsymbol{\alpha'' = (i_1, t''_1), \ldots, (i_n, t''_n)}$ such that both words activate the trace $\boldsymbol{tr}$ and $\boldsymbol{resp(tr, \alpha') \neq resp(tr, \alpha'')}$. Consequently, there exists a pair of output letters $\boldsymbol{o_m}$ and $\boldsymbol{o_k}$ such that

$$\boldsymbol{conv(tr, \alpha') = \cdots, (o_k, T'_k), \ldots, (o_m, T'_m), \ldots}$$
$$\boldsymbol{conv(tr, \alpha'') = \cdots, (o_m, T''_m), \ldots, (o_k, T''_k), \ldots}\ .$$

Such permutation of output letters is possible iff the following inequalities hold

$$\boldsymbol{t'_k + d_k = T'_k < T'_m = t'_m + d_m,}$$
$$\boldsymbol{t''_k + d_k = T''_k > T''_m = t''_m + d_m}\ .$$

But since both input timed words $\boldsymbol{\alpha'}$ and $\boldsymbol{\alpha''}$ activate $\boldsymbol{tr}$, we have the following chain of inequalities:

$$\boldsymbol{\sum_{j=k+1}^{m} u_j < T''_m - T''_k < d_k - d_m < T'_m - T'_k \leq \sum_{j=k+1}^{m} v_j.}$$

Thus, if $\boldsymbol{tr}$ is not steady then there exists a pair of integers such that $\boldsymbol{1 \leq k < m \leq n}$ and

$$\boldsymbol{\sum_{j=k+1}^{m} u_j < d_k - d_m \leq \sum_{j=k+1}^{m} v_j}$$

holds.

End proof.

Now, having the criterion for steadiness of traces, we can give a solution to SDCP for TFSMs. Let TFSM $\boldsymbol{M = (S, s_{in}, G, \rho)}$ be a deterministic TFSM. Denote by $\boldsymbol{u_{min}}$ the greatest lower bound of all left boundaries used in the time guards of $\boldsymbol{M}$. In our model of TFSM $\boldsymbol{u_{min} > 0}$. Let $\boldsymbol{d_{min}}$ and $\boldsymbol{d_{max}}$ be the minimum and the maximum output delays occurred in the transitions of $\boldsymbol{M}$. A theorem below gives necessary and sufficient conditions for the behaviour of $\boldsymbol{M}$ to be strictly deterministic.

Theorem 2. *A deterministic TFSM* $\boldsymbol{M}$ *is strictly deterministic iff all its traces of length* $\boldsymbol{p}$*, where* $\boldsymbol{p = \lceil \frac{d_{max} - d_{min}}{u_{min}} \rceil}$*, are steady.*

Proof. The necessity of conditions is obvious.

We prove the sufficiency of conditions by contradiction. Suppose that all traces of length less or equal $\boldsymbol{p}$ are steady but TFSM $\boldsymbol{M}$ is not. Then there exists such a trace $\boldsymbol{tr}$ in $\boldsymbol{M}$ which is not steady. Then, by Theorem 1, this trace is a sequence of transitions $\boldsymbol{(s_{j-1}, i_j, s_j, b_j, (u_j, v_j], d_j)}$, $\boldsymbol{1 \leq j \leq n}$, such that for some pair of integers $\boldsymbol{m}$ and $\boldsymbol{k}$, where $\boldsymbol{1 \leq k < m \leq n}$, two inequalities

$$\boldsymbol{\sum_{j=k+1}^{m} u_j \leq d_k - d_m \leq \sum_{j=k+1}^{m} v_j}$$

hold. It should be noticed, that, by the same Theorem 1, the trace $\boldsymbol{tr'}$ which includes only the transitions $\boldsymbol{(s_{j-1}, i_j, s_j, b_j, (u_j, v_j], d_j)}$, $\boldsymbol{m \leq j \leq k}$, is not steady as well. Hence, $\boldsymbol{m - k > p}$, and we have the following sequence of inequalities

$$\boldsymbol{d_{max} - d_{min} \geq d_m - d_k \geq \sum_{j=k+1}^{m} u_j > p * u_{min}}$$

which contradicts our choice of $\boldsymbol{p = \lceil \frac{d_{max} - d_{min}}{u_{min}} \rceil}$.

End of proof.

As it follows from Theorems 1 and 2, to guarantee that a given TFSM $\boldsymbol{M = (S, s_{in}, G, \rho)}$ is strictly deterministic it is sufficient to consider all traces $\boldsymbol{(s_0, a_1, b_1, s_1, (u_1, v_1], d_1), \ldots, (s_{n-1}, a_n, b_n, s_n, (u_n, v_n], d_n)}$ in $\boldsymbol{M}$, whose length $\boldsymbol{n}$ does not exceed the value $\boldsymbol{p = \lceil \frac{d_{max} - d_{min}}{u_{min}} \rceil}$ defined in Theorem 2, and for every such trace check that one of the inequalities $\boldsymbol{d_1 - d_n < \sum_{j=2}^{n} u_j}$ or $\boldsymbol{d_1 - d_n > \sum_{j=2}^{n} v_j}$ holds. Thus, we arrive at

Corollary 1. *Strict Determinacy Checking Problem for TFSMs is decidable.*

4. Strict Determinacy Checking Problem for TFSMs is co-NP-hard

Clearly, the decision procedure, based on Theorem 2, is time consuming since p may be exponential of the size of M and the number of traces of length p in TFSM M is exponential of p. In this section we show that such an exhaustive search can hardly be avoided because SDCP for improved version of TFSMs is co-NP-hard.

We are aimed to show that the complement of SDCP is NP-hard. To this end we consider the Subset-Sum Problem (see [7]) which is known to be NP-complete and demonstrate that this problem can be reduced in polynomial time to the complement of SDCP for TFSMs.

The Subset-Sum Problem (SSP) is that of checking, given a set of integers $\boldsymbol{Q}$ and an integer $\boldsymbol{L}$, whether there is any subset $\boldsymbol{Q', Q' \subseteq Q}$, such that the sum of all its elements

is equal to $\boldsymbol{L}$. More formally, the variant of the SSP we are interested in is defined as follows. Let $\boldsymbol{Q} = \boldsymbol{m_1}, \boldsymbol{m_2}, \dots, \boldsymbol{m_N}$ be a sequence of positive integers, and $\boldsymbol{L}$ be also a positive integer. A solution to $(\boldsymbol{Q}, \boldsymbol{L})$-instance of SSP is a binary tuple $\boldsymbol{z} = \langle \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_N} \rangle$ such that $\sum_{j=1}^{N} \boldsymbol{\sigma_j}\, \boldsymbol{m_j} = \boldsymbol{L}$. In [7] it was proved that the problem of checking the existence of a solution to a given $(\boldsymbol{Q}, \boldsymbol{L})$-instance of SSP is NP-complete.

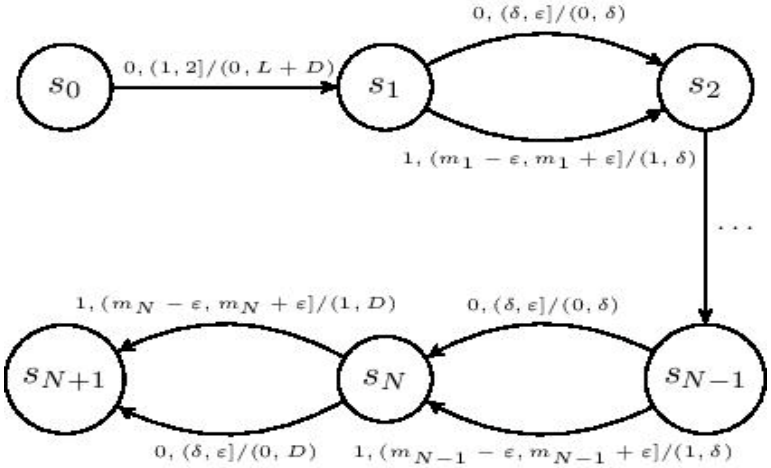

Fig.2 TFSM M

Now, given a $(\boldsymbol{Q}, \boldsymbol{L})$-instance of SSP, we show how to build a deterministic TFSM $\boldsymbol{M_{Q,L}}$ such that it has an initial trace which is *not* strictly determined iff this instance of SSP has a solution. Let $\boldsymbol{D} = \sum_{j=1}^{N} \boldsymbol{m_j}$, and $\boldsymbol{\varepsilon}$ and $\boldsymbol{\delta}$ be positive rational numbers such that $\boldsymbol{\varepsilon} = \boldsymbol{o}(\boldsymbol{1/N^2})$ and $\boldsymbol{\delta} = \boldsymbol{o}(\boldsymbol{\varepsilon/N^2})$. Consider a TFSM depicted in Fig. 2. This machine operates over alphabets $\boldsymbol{I} = \boldsymbol{O} = \{\boldsymbol{0}, \boldsymbol{1}\}$. It has $\boldsymbol{N+2}$ states $\boldsymbol{s_0}, \boldsymbol{s_1}, \dots, \boldsymbol{s_n}, \boldsymbol{s_{N+1}}$. The only transition $(\boldsymbol{s_0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{s_1}, (\boldsymbol{1}, \boldsymbol{2}], \boldsymbol{L} + \boldsymbol{D})$ leads from the initial state $\boldsymbol{s_0}$ to $\boldsymbol{s_1}$. From each state $\boldsymbol{s_j}, \boldsymbol{1} \le \boldsymbol{j} < \boldsymbol{N}$, two transitions $(\boldsymbol{s_j}, \boldsymbol{1}, \boldsymbol{1}, \boldsymbol{s_{j+1}}, (\boldsymbol{m_j} - \boldsymbol{\varepsilon}, \boldsymbol{m_j} + \boldsymbol{\varepsilon}], \boldsymbol{\delta})$ and $(\boldsymbol{s_j}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{s_{j+1}}, (\boldsymbol{\delta}, \boldsymbol{\varepsilon}], \boldsymbol{\delta})$ lead to the state $\boldsymbol{s_{j+1}}$. The state $\boldsymbol{s_N}$ is different: two transitions $(\boldsymbol{s_N}, \boldsymbol{1}, \boldsymbol{1}, \boldsymbol{s_{N+1}}, (\boldsymbol{m_N} - \boldsymbol{\varepsilon}, \boldsymbol{m_N} + \boldsymbol{\varepsilon}], \boldsymbol{D})$ and $(\boldsymbol{s_N}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{s_{N+1}}, (\boldsymbol{\delta}, \boldsymbol{\varepsilon}], \boldsymbol{D})$ lead this state to $\boldsymbol{s_{N+1}}$.

First, we make some observations.

1) Since all transitions outgoing from the states $\boldsymbol{s_j}, \boldsymbol{1} \le \boldsymbol{j} < \boldsymbol{N}$, have the same delay $\boldsymbol{\delta}$, every trace from a state $\boldsymbol{s_k}$ to a state $\boldsymbol{s_\ell}$, where $\boldsymbol{0} < \boldsymbol{k} < \boldsymbol{\ell} \le \boldsymbol{N}$, is strictly deterministic.

2) Since $\boldsymbol{\delta} = \boldsymbol{o}(\boldsymbol{1/N^4})$ and $\boldsymbol{0} < \boldsymbol{\varepsilon} = \boldsymbol{o}(\boldsymbol{1/N^2})$, for every $\boldsymbol{k}, \boldsymbol{1} < \boldsymbol{k} \le \boldsymbol{N}$, and a binary tuple $\boldsymbol{z} = \langle \boldsymbol{\sigma_k}, \boldsymbol{\sigma_{k+1}}, \dots, \boldsymbol{\sigma_N} \rangle$ the inequalities

$$\boldsymbol{\delta - D < 0 < N\delta \le} \sum_{\boldsymbol{j=k+1}}^{\boldsymbol{N}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} - \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\delta})$$

hold. By Theorem 1, this implies that every trace from a state $\boldsymbol{s_k}, \boldsymbol{1} \le \boldsymbol{k} \le \boldsymbol{N}$, to the state $\boldsymbol{s_{N+1}}$ is strictly deterministic.

3) For the same reason the inequalities

$$\boldsymbol{D + L - \delta} > \sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} \boldsymbol{m_j} \; + \; \boldsymbol{k\varepsilon} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{k}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} + \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\varepsilon})$$

hold for every $\boldsymbol{k}, \boldsymbol{1} \le \boldsymbol{k} < \boldsymbol{N}$, and a binary tuple $\boldsymbol{z} = \langle \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_k} \rangle$. By Theorem 1, this guarantees that every initial trace leading to a state $\boldsymbol{s_k}, \boldsymbol{1} \le \boldsymbol{k} \le \boldsymbol{N}$ is strictly deterministic.

As for the initial traces that lead to the state $\boldsymbol{s_{N+1}}$, due to our choice of $\boldsymbol{\varepsilon}$ and $\boldsymbol{\delta}$, we can trust the following chain of reasoning. By definition, a $(\boldsymbol{Q}, \boldsymbol{L})$-instance of SSP has a solution $\boldsymbol{z} = \langle \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_N} \rangle$ iff $\sum_{j=1}^{N} \boldsymbol{\sigma_j}\, \boldsymbol{m_j} = \boldsymbol{L}$. The latter is possible iff two following inequalities hold:

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{\sigma_j}\, \boldsymbol{m_j} \; - \; \boldsymbol{\varepsilon} + \boldsymbol{N\delta} < \boldsymbol{L} < \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{\sigma_j}\, (\boldsymbol{m_j}) \; + \boldsymbol{N\varepsilon} \qquad (1)$$

By taking into account the relationships below

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} - \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\delta}) < \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{\sigma_j}\, \boldsymbol{m_j} \; - \; \boldsymbol{\varepsilon} + \boldsymbol{N\delta}$$

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{\sigma_j}\, (\boldsymbol{m_j}) \; + \boldsymbol{N\varepsilon} = \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} + \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\varepsilon}),$$

we can conclude that (1) holds iff another pair of inequalities hold:

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} - \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\delta}) < \boldsymbol{L} < \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} (\boldsymbol{\sigma_j}(\boldsymbol{m_j} + \boldsymbol{\varepsilon}) + (\boldsymbol{1} - \boldsymbol{\sigma_j})\boldsymbol{\varepsilon})$$

But in the context of observations 1) – 3) above, the latter inequalities, as it follows from Theorem 1, provide the necessary and sufficient conditions that the initial trace in TFSM $\boldsymbol{M_{Q,L}}$ activated by the input word $\boldsymbol{z} = \langle \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \dots, \boldsymbol{\sigma_N} \rangle$ is not strictly deterministic.

Thus, a $(\boldsymbol{Q}, \boldsymbol{L})$-instance of SSP has a solution iff TFSM $\boldsymbol{M_{Q,L}}$ is not strictly deterministic.

The considerations above bring us to

Theorem 3. *SDCP for TFSMs is co-NP-hard.*

5. Conclusion

The main contributions of this paper are

1. the development of a modified version of TFSM which, in our opinion, provides a more adequate model of real-time computing systems;

2. the introduction of the notion of strict deterministic behaviour of TFSM and setting up the Strict Determinacy Checking Problem (SDCP) for a modified version of TFSMs;
3. the establishing of an effectively verifiable criterion for the strict determinacy property of TFSMs;
4. the proving that SDCP for TFSMs is co-NP-hard.

However, some problems concerning strict deterministic behavior of TFSMs still remain open. They will be topics for our further research.

1. In Sections [Sect3] and [Sect4] it was shown that SDCP for TFSMs is co-NP-hard and in the worst case it can be solved in double exponential time by means of a naive exhaustive searching algorithm based on Theorems 1 and 2. We think that this complexity upper bound estimate is too much high. The question arises, for what complexity class F SDCP for TFSMs is a F-complete problem. By some indications we assume that SDCP for TFSMs is PSPACE-complete problem.
2. As it can be seen from the proof of Theorem 3, SDCP for TFSMs is intractable only if timed parameters of transitions (time guards and delays) depend on the number of states in TFSM. But this is not a typical phenomenon in real-time systems since in practice the performance of individual components of a system does not depend on the size of the system. Therefore, it is reasonable to confine ourselves to considering only such TFSMs, in which the time guards and the delays are chosen from some fixed finite set. As it follows from Theorem 2, for this class of TFSMs SDCP is decidable in polynomial time. One may wonder what is the degree of such a polynomial, or, in other words, how efficiently the strict determinacy property can be checked for TFSMs corresponded to real systems.
3. In the model of TFSM besides the usual transitions there are also possible timeout transitions. A timeout transition fires when a timestamped input letter $(\boldsymbol{i}, \boldsymbol{t})$ can not activate any usual transition from a current state. In it was shown that in some cases such timeout transitions can not be replaced by any combination of ordinary transitions. In the future we are going to study how SDCP can be solved for TFSMs with timeouts.

Acknowledgments

The authors of the article express their deep gratitude to V.V. Podymov and the anonymous reviewers for their valuable comments and advice on improving the article. This work was supported by the Russian Foundation for Basic Research, Grant N 18-01-00854.

References

[1]. Alur R., Dill D. A Theory of Timed Automata. Theoretical Computer Science, vol. 126, 1994, pp. 183-235.

[2]. Alur R., Madhusudan P. Decision Problems for Timed Automata: A Survey. In Proceedings of the 4-th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'04), 2004, pp. 1-24.

[3]. Alur R., Fix L., Henzinger Th. A. A Determinizable Class of Timed Automata. In Proceedings of the 6-th International Conference on Computer Aided Verification (CAV'94), 1994, p 1-13.

[4]. Baier C., Bertrand N., Bouyer P., Brihaye T. When are Timed Automata Determinizable? In Proceedings of the 36-th International Colloquium on Automata, Languages, and Programming (ICALP 2009), 2009, p. 43-54.

[5]. Bresolin D., El-Fakih K., Villa T., Yevtushenko N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. In Proceedings of the International Conference GANDALF, 2014, p. 203-216.

[6]. Cardell-Oliver R. Conformance Tests for Real-Time Systems with Timed Automata Specifications. Formal Aspects of Computing, vol. 12, no. 5, 2000, p. 350–371.

[7]. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. 35.5: The subset-sum problem. Introduction to Algorithms (2-nd ed.), 2001.

[8]. Finkel O. Undecidable Problems about Timed Automata. In Proceedings of 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06), 2006, p. 187-199.

[9]. Fletcher J. G., Watson R. W. Mechanism for Reliable Timer-Based Protocol. Computer Networks, vol. 2, 1978, pp. 271-290.

[10]. Merayo M.G., Nuunez M., Rodriguez I. Formal Testing from Timed Finite State Machines. Computer Networks, vol. 52, no 2, 2008, pp. 432-460.

[11]. Ouaknine J., Worrell J. On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap. In Proceedings of the 19-th Annual Symposium on Logic in Computer Science (LICS'04), 2004, pp. 54-63.

[12]. Suman P.V., Pandya P.K., Krishna S.N., Manasa L. Timed Automata with Integer Resets: Language Inclusion and Expressiveness. In Proceedings of the 6-th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'08), 2008, pp. 78–92.

[13]. Tvardovskii A., Yevtushenko N. Minimizing Timed Finite State Machines. Tomsk State University Journal of Control and Computer Science, No 4 (29), 2014, pp. 77-83 (in Russian).

[14]. Tvardovskii A., Yevtushenko N., M. Gromov. Minimizing Finite State Machines with Time Guards and Timeouts. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 4, 2017, pp. 139-154 (in Russian).

[15]. Zhigulin M., Yevtushenko N., Maag S., Cavalli A. FSM-Based Test Derivation Strategies for Systems with Timeouts. In Proceedings of the 11-th International Conference on Quality Software, 2011, p. 141-149.

К проверке строго детерминированного поведения временных конечных автоматов

Е.М.Винарский <vinevg2015@gmail.com>
В.А. Захаров < zakh@cs.msu.su >.
Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1

Аннотация. Конечные автоматы широко применяются в качестве математических моделей при решении многочисленных задач в области программирования, проектирования микроэлектронных схем и телекоммуникационных систем. Для описания поведения систем реального времени модель конечного автомата может быть расширена добавлением в неё часов - параметра непрерывного времени, моделируемого вещественной переменной. В автоматах реального времени для входных и выходных сигналов указывается время их поступления и выдачи, а переходы автомата снабжены описанием задержек, связанных с ожиданием входных сигналов и формированием выходных сигналов. Так же, как и для классических автоматов дискретного времени, задача минимизации конечных автоматов реального времени возникает во многих приложениях этой модели вычислений. Для классической модели автоматов реального времени эта задача уже подробно рассмотрена. В нашей работе мы предлагаем более сложную модель: в ней порядок следования выходных сигналов определяется не только порядком поступления входных сигналов, но также и задержкой, связанной с их обработкой. В этой модели при выполнении одной и той же последовательности переходов выходные сигналы могут выдаваться в разном порядке в зависимости от времени поступления входных сигналов. В новой модели автоматов реального времени решению задачи минимизации должно предшествовать изучение вопроса строгой детерминированности - однозначности поведения автомата на одних и тех же последовательностях переходов. В представленной статье приведены и обоснованы необходимые и достаточные условия строгой детерминированности автоматов реального времени, а также исследованы вопросы, связанные с решением задачи минимизации этой разновидности автоматов.

Ключевые слова: конечные временные автоматы; строго детерминированное поведение

DOI: 10.15514/ISPRAS-2018-30(3)-22

Для цитирования: Винарский Е.М., Захаров В.А. К проверке строго детерминированного поведения временных конечных автоматов. Труды ИСП РАН, том 30, вып. 3, 2018 г., стр. 325-340 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(3)-22

Список литературы

[1]. Alur R., Dill D. A Theory of Timed Automata. Theoretical Computer Science, vol. 126, 1994, pp. 183-235.

[2]. Alur R., Madhusudan P. Decision Problems for Timed Automata: A Survey. In Proceedings of the 4-th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'04), 2004, pp. 1-24.

[3]. Alur R., Fix L., Henzinger Th. A. A Determinizable Class of Timed Automata. In Proceedings of the 6-th International Conference on Computer Aided Verification (CAV'94), 1994, p 1-13.

[4]. Baier C., Bertrand N., Bouyer P., Brihaye T. When are Timed Automata Determinizable? In Proceedings of the 36-th International Colloquium on Automata, Languages, and Programming (ICALP 2009), 2009, p. 43-54.

[5]. Bresolin D., El-Fakih K., Villa T., Yevtushenko N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power. In Proceedings of the International Conference GANDALF, 2014, p. 203-216.

[6]. Cardell-Oliver R. Conformance Tests for Real-Time Systems with Timed Automata Specifications. Formal Aspects of Computing, vol. 12, no. 5, 2000, p. 350–371.

[7]. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. 35.5: The subset-sum problem. Introduction to Algorithms (2-nd ed.), 2001.

[8]. Finkel O. Undecidable Problems about Timed Automata. In Proceedings of 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06), 2006, p. 187-199.

[9]. Fletcher J. G., Watson R. W. Mechanism for Reliable Timer-Based Protocol. Computer Networks, vol. 2, 1978, pp. 271-290.

[10]. Merayo M.G., Nuunez M., Rodriguez I. Formal Testing from Timed Finite State Machines. Computer Networks, vol. 52, no 2, 2008, pp. 432-460.

[11]. Ouaknine J., Worrell J. On the Language Inclusion Problem for Timed Automata: Closing a Decidability Gap. In Proceedings of the 19-th Annual Symposium on Logic in Computer Science (LICS'04), 2004, pp. 54-63.

[12]. Suman P.V., Pandya P.K., Krishna S.N., Manasa L. Timed Automata with Integer Resets: Language Inclusion and Expressiveness. In Proceedings of the 6-th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'08), 2008, pp. 78–92.

[13]. А.С. Твардовский, Н.В. Евтушенко. К минимизации автоматов с временными ограничениями. Вестник Томского государственного университета. Управление, вычислительная техника и информатика, vol. 29, no 4, 2014, pp. 77-83.

[14]. Твардовский А.С., Евтушенко Н.В., Громов М.Л. Минимизация автоматов с таймаутами и временными ограничениями. Труды ИСП РАН, том 29, вып. 4, 2017 г., стр. 139-154. DOI: 10.15514/ISPRAS-2017-29(4)-8..

[15]. Zhigulin M., Yevtushenko N., Maag S., Cavalli A. FSM-Based Test Derivation Strategies for Systems with Timeouts. In Proceedings of the 11-th International Conference on Quality Software, 2011, p. 141-149.