УДК 519.713.8

On some properties of timed finite state machines

Vinarskii E.M., Zakharov V.A. (Lomonosov Moscow State University)

Sequential reactive systems are formal models of programs that interact with the environment by receiving inputs and producing corresponding outputs. Such formal models are widely used in software engineering, computational linguistics, telecommunication, etc. In real life, the behavior of a reactive system depends not only on the flow of input data, but also on the time the input data arrive and the delays that occur when generating responses. To capture these aspects, a timed finite state machine (TFSM) is used as a formal model of a real-time sequential reactive system. However, in most of previously known works, this model was considered in simplified semantics: transduction relations of TFSMs are defined in such a way that the responses in the output stream, regardless of their timestamps, follow in the same order in which the corresponding inputs are delivered to the machine. This simplification makes the model easier to analyze and manipulate, but it misses many important aspects of real-time computation. In this paper we study a more realistic semantics of TFSMs and show how to represent it by means of Labeled Transition Systems. This opens up a possibility to apply traditional formal methods for verifying more subtle properties of real-time reactive behavior which were previously ignored.

***Keywords**: Timed finite state machines, transduction relation, safety property, Labeled Transition System, bisimulation*

1. Introduction

Timed finite state machines (TFSMs) are, perhaps, the most simple extensions of finite state machines (FSMs) which are widely used for modelling and analysis of real-time reactive systems. There are several known ways to generalize the concept of finite state machines for modeling real time computations. One of the most advanced is the concept of Timed Automata (TA) [1]. In [2–4] it was shown that TAs supplied with clocks (timers), timed guards at their transitions and timed invariants at their states capture many important aspects of real-time computations. However, the behavior of TAs is difficult for the analysis since TAs were given too much freedom in handling their timers. Complex manipulations with timers are unavoidable when it is necessary to synchronize several events at once (e.g. to output the next control signal no more than 2 time units after receiving the input data, but no earlier than 1 time unit after the previous control signal is issued).

However, in many practically important cases, the behavior of control systems is not so complicated, and the response of a system is determined exclusively by the current control state and the parameters of the input data. Therefore, in order to avoid the difficulties arising when working with such a complex computational model as TAs, the authors of [7, 10, 11] extended a FSM with a single timer and distinguished three families of TFSMs depending on admissible *modus operandi* with timers: (i) a TFSM with timed guards which fire transitions only when a timestamp of an input satisfies the corresponding time guard; (ii) a TFSM with timeouts which force the machine to make a transition when a prescribed waiting time expires; (iii) a TFSM with timed guards and timeouts. Whenever a TFSM moves to a new state, it resets its timer. It was shown that TFSMs of this kind may be used as adequate formal models for many interesting applications where real-time effects are concerned, and they are also admit efficient algorithms for analysis and manipulations.

As a model of real time reactive system a TFSM converts timestamped input streams into timestamped output streams. A distinguishing feature of those classes of TFSMs that have been studied in [7, 10, 11] is that the order of the outputs is determined not by their timestamps, but by the order of the corresponding inputs. This principle alleviates substantially verification and optimization of TFSMs, but it also makes such a model inadequate for many applications. Consider, for example, a behavior of a controller in Software Defined Networks (SDN) [9]. It supplies flow tables of network switches with packet forwarding rules. Packets arrived to an input buffer of a switch are matched against a flow table to select an appropriate rule which either forwards the packet to some output buffer, or drops it. When a controller updates flow tables, the order in which new rules are set in a table may differ from the order in which they were sent by the controller due to delivery delays. When the controller, say, receives requests R_1, R_2 from the network it responds with a pair of commands A_1, A_2 which add new entries to a flow table. It may happen so (see [13]) that the controller at receiving a sequence of timestamped inputs (input timed word) $(R_1, t_1), (R_2, t_2)$, where $t_1 < t_2$, computes an output timed word $(A_2, \tau_1), (A_1, \tau_2)$, where $\tau_2 < \tau_1$, i.e. the second rule will be set before the first one. This incorrectness cannot be detected using the early variants of TFSMs.

To cope with this shortcoming of conventional TFSMs and to make their behaviour more "realistic" some improvements to TFSMs' semantics were introduced in [12]. A new model of TFSMs captures an important feature of reactive system computations: the order of responses in the output stream is determined by the time they were generated, not the order of the

corresponding requests in the input stream: if a request b arrives at the input of a machine after a signal a then it is possible that a response to a follows a response to b. An improved semantics of TFSMs allows one to represent explicitly the erroneous behaviour of SDN controller mentioned above. But the new semantics of TFSMs loses the incremental property: when an input α' extends an input α it is not necessary that the corresponding output β' extends an output β which results from α. The lack of this fine property significantly complicates the solution of the verification problem if we use only the transition relations in TFSM programs.

To overcome this fundamental difficulty we introduce an alternative semantics of TFSMs that better accommodate traditional verification techniques. This semantics is defined in terms of Labeled Transition Systems (LTSs) based on the concept of TFSM configuration. However, the statespace of such $LTS(\mathcal{T})$ corresponding to a TFSM $\mathcal{T}$ is uncountable. Our long term goal is to develop an algorithm which for every TFSM $\mathcal{T}$ constructs a finite $LTS_{fin}(\mathcal{T})$ which simulates $LTS(\mathcal{T})$ and, thus, can be used for verification of $\mathcal{T}$. In this paper we present some preliminary results of our research.

In Section 2 and 3 we introduce the concept of TFSMs with the improved automata-theoretic semantics, and demonstrate how it can be used to adequately describe the behavior of SDN controllers. In Section 4 we define LTS-based semantics for TFSMs under consideration, study its relationship with an automata-theoretic semantics, and present some results that contribute to reducing the size of $LTS(\mathcal{T})$ and building $LTS_{fin}(\mathcal{T})$. Section 5 concludes the paper and presents some avenues for future work.

This research is supported by RFBR project No. 18-01-00854.

2. Preliminaries

Consider two non-empty finite alphabets A and B; the alphabet A is called an *input* alphabet while B is an *output* alphabet. The symbols of A can be viewed as control signals (inputs) received by some real-time system, and the symbols of B may be regarded as responses (outputs) generated by the system. A finite sequence $\alpha = a_1, a_2, \ldots, a_n$ of inputs is called an *input word*, whereas a sequence $\beta = b_1, b_2, \ldots, b_n$ of outputs is called an *output word*.

For modeling time we use a special variable which takes values in the non-negative real domain $\mathbb{R}_0^+$. Temporal aspects of computations are defined with the help of such notions as timestamps, time sequences, time guards and delays. A *timestamp* is a positive real number from $\mathbb{R}^+$ which indicates a time instance when some event occurs. A *time sequence* is any

increasing sequence of timestamps. A *time guard* (and a *delay*) is an interval $g = \langle u, v\rangle$, where $\langle \in \{(, [\}, \rangle \in \{),]\}$, and u, v are such numbers from $\mathbb{R}^+$ that $u \leq v$. Time guards and delays are used to specify periods of time in which some events are possible. We call u the *left bound* of a timed guard (delay) g while v is the *right bound* of this timed guard (delay). If a delay is a singleton $[d, d]$ then we say that it is a *sharp delay* (and write d instead of $[d, d]$)

Let $w = x_1, x_2, \ldots x_n$ and $\tau = t_1, t_2, \ldots, t_n$ be an input (output) word and an increasing time sequence of the same length. Then any pair (x_i, t_i) is called an input (output) *timed symbol*, and a pair $\alpha = (w, \tau)$ is called a *timed word*. We denote by $t(\alpha)$ the last value τ_n in the sequence τ. Let A and B be an input and an output alphabets. Denote by G and D some sets of time guards and delays. Then a *Timed Finite State Machine* (TFSM) over A, B, G, and D is a triple $\mathcal{T} = (S, \rho, s_0)$ where S is a finite non-empty *set of states*, $\rho \subseteq (S \times A \times G \times B \times D \times S)$ is a finite *transduction relation*, s_0 is an *initial state*.

Every 6-tuple (s, a, g, b, d, s') in ρ is called a *transduction* of $\mathcal{T}$. Such a transduction is an atomic action that a TFSM $\mathcal{T}$ can perform: if after time t since the machine is into a state s, a signal a arrives at its input and the guard condition $t \in g$ for triggering the transduction is satisfied, then $\mathcal{T}$ immediately passes to the state s' and the output response b is produced after time $t' = t + \delta$ where $\delta \in d$. A TFSM applies these actions to all timed inputs (a_i, t_i) of a given input timed word (w, τ) and generates, as a result, an output timed word (z, τ'). This behavior of TFSMs is defined formally as follows.

A *run* of a TFSM $\mathcal{T}$ on an input timed word $\alpha = (a_1, t_1), (a_2, t_2), \ldots, (a_n, t_n)$ is a finite sequence $r = s_0 \xrightarrow{(a_1,t_1)/(b_1,\tau_1)} s_1 \xrightarrow{(a_2,t_2)/(b_2,\tau_2)} \ldots \xrightarrow{(a_n,t_n)/(b_n,\tau_n)} s_n$ of 6-tuples from $S \times A \times \mathbb{R}^+ \times B \times \mathbb{R}^+ \times S$ such that for each $i \in \{1, \ldots, n\}$ there exists a transduction $(s_{i-1}, a_i, g_i, b_i, d_i, s_i)$ of $\mathcal{T}$ which complies with the following requirements: 1) $t_i - t_{i-1} \in g_i$, and 2) $\tau_i = t_i + \delta$, where $\delta \in d_i$. In this case we say that a run r of a TFSM $\mathcal{T}$ *converts* the input timed word α to the output timed word $\beta = (b_{j_1}, \tau_{j_1}), (b_{j_2}, \tau_{j_2}), \ldots, (b_{j_n}, \tau_{j_n})$ which is the permutation of the sequence $(b_1, \tau_1), (b_2, \tau_2), \ldots, (b_n, \tau_n)$. For every TFSM $\mathcal{T}$ we denote by $TR(\mathcal{T})$ a *transduction relation* computed by $\mathcal{T}$ which is the set of all pairs (α, β), where α is an input timed word, and β is the result of conversion of α by $\mathcal{T}$. If all delays in the set D are sharp then we say that $\mathcal{T}$ is a *TFSM with sharp delays*. In this paper we consider only TFSMs with sharp delays.

Consider, for example, a TFSM $\mathcal{T}$ shown on Fig. 1 and a run r induced by the input timed word $\alpha = (a_1, 1.5), (a_2, 2.7), (a_3, 4)$. Then this run converts α to the output timed word $\beta = (b_2, 3.7), (b_1, 4.5), (b_3, 6)$.

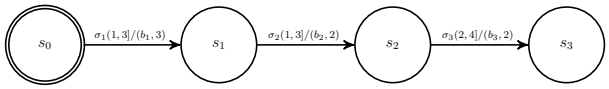

Figure 1. A TFSM $\mathcal{T}$

3. Safety property for real-time systems

In this section, we show that for the modified semantics of TFSMs as defined above, the analysis of safety properties turns out to be difficult. A subset $P_{safe} \subseteq A^\omega$ is called a *safety property* (see [5]) if every ω-word $\alpha \in A^\omega \setminus P_{safe}$ has such a finite prefix β that $\beta\alpha' \notin P_{safe}$ holds for any ω-word α', i.e. if a run α of a system is unsafe then a safety violation can be detected after a finite number of steps β, and no further system actions α' can eliminate this error. A straightforward extension of this concept to the case of real-time computations brings us to the following definition. A subset $P_{safe} \subseteq (A \times \mathcal{R}^+)^\omega$ is called a *real-time safety property* if for every timed ω-word $\alpha \in (A \times \mathcal{R}^+)^\omega \setminus P_{safe}$ there exists such a finite timed prefix β of α that $\alpha' \notin P_{safe}$ holds for any timed extension α' of β.

This is a reasonable and intuitive definition, but when it comes to TFSMs operating within the improved semantics, it brings also some unpleasant effects that hinder verification of these real time models of computation. Usually *safety* means that no errors happen in the course of a run of a model, and if such an error is detected after some finite number of steps, then it can be announced regardless of the subsequent steps of the computation. However, this common and useful feature of safety checking does not always hold in the case of TFSMs. We illustrate this phenomenon with an example. Consider an SDN controller $\mathcal{C}$ which receives requests to update a flow table of an SDN switch and generates in response the appropriate instructions. Since the formation and delivery of an instruction to a network switch takes some time, the responses start to take effect with some delay. A TFSM which is a real-time model of such a controller is depicted on Fig. 2. It operates with the set of inputs (requests) PF_i (put a rule r_i into a flow table), GF_i (get an info about a rule r_i from a flow table), DF_i (delete a rule r_i from a flow table), and the set of outputs (responses) FA_i (a rule r_i is inserted into a flow table), FR_i (info about a rule r_i is received from a flow table), FD_i (a rule r_i is erased in a flow table).

The safety property that the designers of a controller wants to respect requires that flow table update instructions must be always activated the same order as flow table update requests are received. When TFSM $\mathcal{C}$ receives an input timed word $\alpha = (GF_1, 1.5), (PF_1, 3.7), (PF_2, 5.5)$ it converts it to an output timed word $\beta = (FR_1, 3.5), (FA_2, 8.5), (FA_1, 10.7)$. Clearly, such a

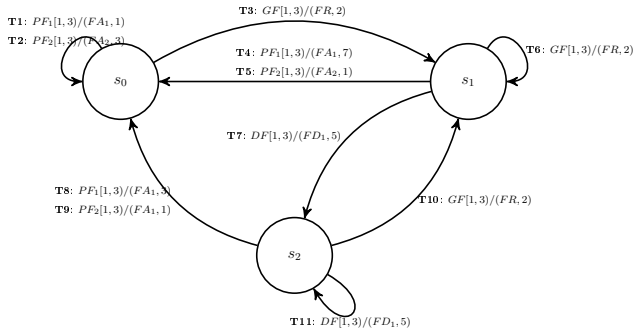

Figure 2. An SDN Controller

behavior does not satisfies the safety requirement. But if $\mathcal{C}$ continues to receive input signals $\alpha' = (GF_1, 1.5), (PF_1, 3.7), (PF_2, 5.5), (PF_1, 6.7)$ then the corresponding output timed word $\beta' = (FR_1, 3.5), (FA_1, 7.7), (FA_2, 8.5), (FA_1, 10.7)$ will indicate that the satisfiability of the safety property has been restored. To cope with such cases, TFSMs need to be provided with a more suitable way for presenting their behavior.

4. From TFSM configurations to a Labeled Transition System

Operational semantics of TFSMs defined so far suits well the modeling of real-time reactive information processing systems since it quite naturally captures many important effects of real-time computations. However, this semantics is unfavorable for the application of traditional methods and tools for analyzing the behavior of real-time systems, since it lacks the concept of a state of computation as a snapshot of a computing process. In this section to overcome this drawback we introduce a concept of a configuration which makes it possible to represent TFSMs' behaviors by means of Labeled Transition Systems on such configurations.

4.1. Configurations of Timed Finite State Machine

Intuitively, a configuration of a TFSM $\mathcal{T}$ is a snapshot of a TFSM's computation which includes 3 components: a control state of a TFSM $\mathcal{T}$, time elapsed since the last input (timer value), and a set of output timed symbols that have been generated but not yet issued due to delays. Formally, a configuration q is a triple $\langle s, t, TC \rangle$ such that:

- $s \in S$ is a state of $\mathcal{T}$; it will be referred as $q.s$;
- $t \in \mathcal{R}_0^+$ is time elapsed since the last input (referred as $q.t$);
- $TC = \{(b_1, \tau_1), (b_2, \tau_2), \ldots, (b_n, \tau_n)\}$, where $(b_i, \tau_i), 1 \leq i \leq n$ are timed output symbols, is an *output timed context* of q (referred as $q.c$).

A *capacity of a configuration* q is the cardinality $|q.c|$ of its output timed context. We denote by $q.T$ a value $min(\tau_1, \ldots, \tau_n)$ if $q.c \neq \emptyset$; in the case of $q.c = \emptyset$ we assume that $q.T = \infty$. We denote by $Q(\mathcal{T})$ the set of all possible configurations of a TFSM $\mathcal{T}$.

4.2. Labeled Transition Systems

For any given TFSM $\mathcal{T}$ over an input alphabet A and an output alphabet B we define a Labeled Transition System $\mathcal{LTS}(\mathcal{T})$ which has three types of transitions on configurations: (i) time advancement, (ii) input read, and (iii) output write.

Formally, a *Labeed Transition System* $\mathcal{LTS}(\mathcal{T})$ of a TFSM $\mathcal{T} = (S, \rho, s_0)$ over alphabets A and B is a triple $(Q(\mathcal{T}), q_0, \rho_{\mathcal{LTS}})$ where $q_0 = \langle s_0, 0, \emptyset \rangle$ is the *initial configuration*, and $\rho_{\mathcal{LTS}}$ is a transition relation of the type $(Q \times \mathcal{R}^+ \times Q) \cup (Q \times A \times Q) \cup (Q \times B \times Q)$ which is defined for every configuration $q = \langle s, t, \{(b_1, \tau_1), (b_2, \tau_2), \ldots, (b_m, \tau_m)\} \rangle$ as follows.

1. For every $\delta \in \mathcal{R}^+$ such that $\delta \leq q.T$ there exists a time advancement transition $q \xrightarrow{\delta} q'$ in $\rho_{\mathcal{LTS}}$, where $q' = \langle s, t + \delta, \{(b_1, \tau_1 - \delta), (b_2, \tau_2 - \delta), \ldots, (b_m, \tau_m - \delta)\} \rangle$.
2. For every transduction $(s, a, \langle u, v \rangle, c, d, s')$ of $\mathcal{T}$ such that $t \in \langle u, v \rangle$ there exists an input transition $q \xrightarrow{a} q'$ in $\rho_{\mathcal{LTS}}$, where $q' = \langle s', 0, q.c \cup \{(c, d)\} \rangle$; in this case, we say that this input transition is *induced* by this transduction.
3. For every pair $(b, 0) \in q.c$ there exists an output transition $q \xrightarrow{b} q'$ in $\rho_{\mathcal{LTS}}$, where $q' = \langle s, t, q.c \setminus \{(b, 0)\} \rangle$.

A *path* in $\mathcal{LTS}(\mathcal{T})$ is any sequence of transitions $\pi = q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_1} \cdots q_{k-1} \xrightarrow{x_k} q_k$. A path π is called *complete* if $q_k.c = \emptyset$. With every path π in $\mathcal{LTS}(\mathcal{T})$ we associate a pair $TR(\pi) = (\alpha, \beta)$ of an input and an output timed words which is defined by induction on the length of π according to the following rules.

1. If π is an empty path then $TR(\pi) = (\varepsilon, \varepsilon)$;
2. Suppose that π' is a path in $\mathcal{LTS}(\mathcal{T})$ from q_0 to q' such that $TR(\pi') = (\alpha', \beta')$, and a path π is an extension of π' with a transition $E = q' \xrightarrow{x} q$. If E is
   - a time advancement transition then $TR(\pi) = TR(\pi')$;
   - an input transition then $TR(\pi) = (\alpha, \beta')$, where $\alpha = \alpha', (x, t(\alpha') + q'.t)$;
   - an output transition then $TR(\pi) = (\alpha', \beta)$, where $\beta = \beta', (x, t(\alpha') + q'.t)$.

A transduction relation $TR(\mathcal{LTS}(\mathcal{T}))$ specified by a $\mathcal{LTS}(\mathcal{T})$ is the set of all pairs $TR(\pi)$ associated with all complete paths π in $\mathcal{LTS}(\mathcal{T})$. As it may be seen, $\mathcal{LTS}(\mathcal{T})$ thus defined completely characterizes the behavior of a TFSM $\mathcal{T}$.

Proposition 1. $TR(\mathcal{T}) = TR(\mathcal{LTS}(\mathcal{T}))$ holds for every TFSM $\mathcal{T}$.

Proof (draft). Clearly, every finite path in $\mathcal{LTS}(\mathcal{T})$ can be extended to a complete path; so, $\mathcal{LTS}(\mathcal{T})$ has no useless paths. The inclusion $TR(\mathcal{T}) \subseteq TR(\mathcal{LTS}(\mathcal{T}))$ is proved straightforwardly by constructing an appropriate complete path in $\mathcal{LTS}(\mathcal{T})$ for every finite run of $\mathcal{T}$. The inclusion $TR(\mathcal{LTS}(\mathcal{T})) \subseteq TR(\mathcal{T})$ can be proved by induction on the length of an input timed word α.

The main advantage of $\mathcal{LTS}(\mathcal{T})$ is that this a model of the same type as that used for verification of timed automata [1]. This makes it possible to apply such model checking tools as Uppaal [6] for the analysis of behavior of TFSMs. However, such an analysis is fraught with certain difficulties: the statespace of $\mathcal{LTS}(\mathcal{T})$ may be infinite. Partially, this is due to the fact that the capacity of configurations in $\mathcal{LTS}(\mathcal{T})$ is unbounded in general case. Nevertheless, under certain conditions some bounds on the capacity of configurations of TFSMs can be established.

Let u, v be a pair of real numbers such that $0 < u \leq v$. A TFSM $\mathcal{T} = (S, \rho, s_0)$ is called (u, v)-*progressive* if for every transduction $(s, a, g, b, d, s') \in \rho$ a guard $g = (u', v')$ is such that $u < u'$ and $v' < v$, i.e. the guards of all transductions of $\mathcal{T}$ are within the interval (u, v).

Proposition 2. If $\mathcal{T}$ is a (u, v)-progressive TFSM with sharp delays then there exists such an integer ℓ that $|q.c| < \ell$ holds for all configurations $q \in Q(\mathcal{T})$ reachable in $\mathcal{LTS}(\mathcal{T})$ from the initial configuration q_0.

Proof (draft). Denote by d_{max} the maximal sharp delay in the transductions of $\mathcal{T}$, and let $\ell = \lceil \frac{d_{max}}{u} \rceil$. Then by *redictio ad absurdum* it can be shown that for every path $\pi = q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_1} \cdots q_{k-1} \xrightarrow{x_k} q_k$ in $\mathcal{LTS}(\mathcal{T})$ the inequality $|q_i.c| \leq \ell$ holds for every $1 \leq i \leq k$.

Another difficulty in analyzing the models $\mathcal{LTS}(\mathcal{T})$ of TFSMs is a kind of livelock effect (see [5]) when any extension of a path in $\mathcal{LTS}(\mathcal{T})$ is achieved by time advancement transitions only. To detect and exclude from further consideration such redundant paths, we will use the notion of timelock configuration (see [5]). For every state s of a TFSM $\mathcal{T}$ denote by $up(s)$ the maximal upper bound v in the guards $g = (u, v)$ of all transductions (s, a, g, b, d, s') that are enable in s. A configuration q is called a *timelock configuration* if $q.c = \emptyset$ and $q.t > up(q.s)$. All other configurations in $Q(\mathcal{T})$ are called *progressive*. As it may be seen from this definition, only progressive configurations matter: if a path in $\mathcal{LTS}(\mathcal{T})$ reaches a timelock configuration then there are no outputs pending and no inputs will be able to trigger any further transduction

of $\mathcal{LTS}(\mathcal{T})$. Therefore, no further analysis of paths outgoing from timelock configuration is necessary. Detection of timelock configurations is easy for (u, v)-progressive TFSMs.

Proposition 3. For every state s of a (u, v)-progressive TFSM $\mathcal{T}$ there exists $c_s \geq 0$ such that every configuration $q = (s, t, TC)$ is a timelock configuration iff $q.t > c_s$

Proof. Let s be a state of $\mathcal{T}$ and let $v = ub(s)$. Consider a set of configurations $Q_s = \{q \in Q \mid (q.s = s) \wedge (q.t \geq v) \wedge (q.c = \emptyset)\}$ and the corresponding set of timestamps $TS_s = \{t \mid (q \in Q_s) \wedge (q.t = t)\}$. It easy to see that TS_s has the infimum c_s which satisfies the assertion of the proposition.

Our next step will be to find out or introduce effectively a finitely indexed equivalence relation $\sim$ on the set of progressive configurations of $\mathcal{LTS}(\mathcal{T})$ which makes it possible to construct such a finite model $\mathcal{LTS}_{fin}(\mathcal{T}) = \mathcal{LTS}(\mathcal{T})/\sim$ that $\mathcal{LTS}_{fin}(\mathcal{T})$ simulates $LTS(\mathcal{T})$ w.r.t. some suitable simulation relation. Such a finite model $\mathcal{LTS}_{fin}(\mathcal{T})$ opens a way for applying conventional model checking tools for verification of reactive systems represented by TFSMs of a new type.

5. Conclusion and future work

In this paper we showed that timed finite state machines which generate responses to the input signals in the order that corresponds to the output delays is a quite adequate model of real-time reactive computing systems and it could be used in some applications. However, it turned out that some safety properties that arise in such applications are difficult to analyze based on the conventional operational semantics for TFSMs. To overcome this trouble we adapt the operational semantics of TFSMs to the concept of Labeled Transition Systems to take advantage of well-known verification techniques for models of real-time systems.

Since $LTS(\mathcal{T})$ which represents all possible runs of a TFSM $\mathcal{T}$ may have infinitely many internal states (configurations), our next step we are going to made in the future work is to develop a technique for converting infinite $LTS(\mathcal{T})$ to finite $\mathcal{LTS}_{fin}(\mathcal{T})$ in such a way that $\mathcal{LTS}_{fin}(\mathcal{T})$ simulates $LTS(\mathcal{T})$ w.r.t. some suitable simulation relation. We expect to find out an appropriate simulation relation which (i) preserves the most important properties of TFSM behaviours, and (ii) admits an efficient translation of a TFSM $\mathcal{T}$ into its finite model $\mathcal{LTS}_{fin}(\mathcal{T})$. We think that such a translation could be developed with the help of configuration patterns – a new concept which would play the same role for model checking of TFSMs as timed

regions for the analysis of timed automata (see [1]).

The authors of the article are grateful to the anonymous reviewers for useful comments that helped to improve the article.

References

1. Alur R., Dill D. A theory of timed automata // Theoretical Computer Science. 1994. Vol. 126, № 2. P. 183–235.
2. Alur R., Madhusudan P. Decision Problems for Timed Automata // Proceedings of the 4-th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM'04). 2004. P. 1–24.
3. Asarin E., Caspi P., Oded M. Timed Regular Expressions // Journal of the ACM. 2001. Vol. 49, № 2. P. 1–35.
4. Asarin E., Caspi P., Oded M. A Kleene theorem for timed automata // Proceedings of the 12-th Annual IEEE Symposium on Logic in Computer Science (LICS'97). 1997. P. 160–171.
5. Baier C., Katoen J. Principles of Model Checking. Cambridge: MIT Press Cambridge. 2008. 994 p.
6. Behrmann G., David A., Larsen K.G. A tutorial on Uppaal // Proceedings of the International School on Formal Methods for the Design of Computer, Communication, and Software Systems. 2004. P. 200–236.
7. Bresolin D, El-Fakih K., Villa T., Yevtushenko N. Deterministic Timed Finite State Machines: Equivalence Checking and Expressive Power // Proceedings of the 5-th International Symposium on Games, Automata, Logics and Formal Verification. 2014. P. 203–216.
8. Gill A. Introduction to the Theory of Finite-state Machines. New York: McGraw-Hill, 1962, 207 p.
9. McKeown N., Anderson T., Balakrishnan H., et. al. OpenFlow: enabling innovation in campus networks // ACM SIGCOMM Computer Communication Review, 2008. Vol. 38, № 2. P. 69–74.
10. Tvardovskii A., Yevtushenko N. Minimizing finite state machines with time guards and timeouts // Proceedings of ISP RAS. 2017. V. 29, № 4. P. 139–154
11. Tvardovskii A., Yevtushenko N. Minimizing Timed Finite State Machines // Tomsk State University Journal of Control and Computer Science. 2014. Vol. 29, № 4. P. 77–83.
12. Vinarskii E., Zakharov V. On the verification of strictly deterministic behaviour of Timed Finite State Machines// Proceedings of ISP RAS. 2018. Vol. 30, № 3. P. 325–340.
13. Vinarskii E., López J., Kushik N., Yevtushenko N., Zeghlache M. A Model Checking Based Approach for Detecting SDN Races // Proceedings of the 31-st IFIP WG 6.1 International Conference on Testing Software and Systems – ICTSS. 2019. P. 194–211.