Experiments on Parallel Composition of Timed Finite State Machines[1]

A.P. Sotnikov <sotnikhtc@gmail.com>
N.V. Shabaldina <nataliamailbox@mail.ru>
M.Л, Gromov <maxim.leo.gromov@gmail.com>
Tomsk State University,
36, Lenin ave., Tomsk, 634050, Russia

Abstract. In this paper, we continue our work that is devoted to the parallel composition of Timed Finite State Machines (TFSMs). We consider the composition of TFSMs with timeouts and output delays. We held experiments in order to estimate how often parallel composition of nondeterministic TFSMs (with and without timeouts) has infinite sets of output delays. To conduct these experiments we have created two tools: the first one for converting TFSMs into automata (this tool is integrated into BALM-II), the second one for converting the global automaton of the composition into TFSM. As it was suggested in earlier works, we describe the infinite sets of output delays by linear functions, and it is important to know how often these sets of linear functions appear to justify the importance of future investigations of the TFSM parallel compositions (especially for deriving cascade composition). Results of the experiments show significant amount (around 50 %) of TFSMs with infinite number of output delays. We also estimate the size of the global automaton and the composed TFSM. In the experiments, we do not consider global automata with the huge number of states (more then 10000).

Keywords: Timed finite state machine; parallel composition; BALM-II.

DOI: 10.15514/ISPRAS-2017-29(3)-13

For citation: Sotnikov A.P., Shabaldina N.V., Gromov M.L. Experiments on Parallel Composition of Timed Finite State Machines. *Trudy ISP RAN/Proc. ISP RAS*, vol. 29, issue 3, 2017, pp. 233-246. DOI: 10.15514/ISPRAS-2017-29(3)-13

1. Introduction

Different systems, for example web-services, telecommunication protocols digital networks etc. are targeted on interaction with each other. To analyze and synthesize such systems one needs an adequate formal model. *The Finite State Machine* (FSM) has proven to be a classical model for description of *input-output reactive discrete event systems* [1]. Here, by "*input-output reactive*" we mean, that every input action

[1] This work is supported by the grant for the basic research №16-49-03012 of Russian Scientific Foundation.

is always followed by output reaction, and by "*discrete event*" we mean, that domains for input and output actions are finite (discrete) sets. In this case, when talking about interacting systems, main concept is *a composition* of FSMs. If there are two communicating systems and the behavior of each system is described by an FSM, then their common work can be described by the composition of those FSMs. Under appropriate assumptions [2,3] this composition will also be an FSM. In this work we consider so-called parallel composition [2]. In the parallel composition the interacting systems work asynchronously in the assumption of a slow environment and this is enough to guarantee the composition to be an FSM again. To build an FSM composition BALM-II (Berkeley Automata and Language Manipulation) can be used [3].

For more precise description of a system one should consider time aspects of its behavior as well. For that reason we need some model which would be appropriate for description of an input-output reactive timed discrete event system. Probably the most general way to describe timed discrete event system (not necessarily input-output reactive) is a timed automaton [6]. In works [4, 5] authors describe web-services, using language BPEL. In [4] authors tell how they translate web services into timed automata in order to verify them. In [5] the authors use more complicated model – so-called *Timed Extended Finite State Machine*. Then they convert a TEFSM into timed automaton. For further analysis, both, [4] and [5], use UPPAAL [16] as an instrument.

Although timed automata are more than enough to describe any input-output reactive timed discrete event system, they are not convenient for us. The reason is that we would like to keep some room for analysis of the composition. Namely, we would like to use the composition as a specification for a test generation. For the best of our knowledge, methods of a test generation with guaranteed fault coverage for (timed) automata are not well developed (frankly speaking we know only one paper [7], which describes such a method). In contrary, test generation methods for FSMs are well-developed and are still developing [8, 9]. And since parallel composition of FSMs guarantees, that the result is FSM again, we would like to consider some kind of *Timed Finite State Machine* as a model, which would presume this property of the parallel composition. One possible timed augmentation of the FSM was mentioned in [5]. But this model is quite complex and it is not clear, how to build parallel composition for it. Another option to introduce timed FSM is Timed FSM with time guards [10]. The theory of this model is highly developed [11], but it lacks an efficient method to build parallel composition as well as the precious model.

And at last, the model we use in this paper, is the *Timed Finite State Machine with output delays and timeouts* (TFSM) [12-15]. This model allows building parallel composition in the same manner as it is done for common FSM. Given two TFSMs we need to compose. First, the corresponding automata should be built [7], then we compose those automata, obtaining so called *the global automaton of the composition*. And then we need to transform the global automaton into TFSM. In [12] very interesting effect of the parallel composition is shown. It turned out that composition of two TFSMs (with constant delays) can have infinite number of output delays for a some transitions and those delays can be described by a finite set of linear functions $\{b + k \cdot t \mid b, k \in \{0\} \cup \mathbb{N}\}$. The main objective of this paper is to investigate how often this effect occurs. This would justify further development of the theory of TFSM with infinite (countable) number of delays.

In some works [4, 5] authors use UPPAAL [16] as an instrument for manipulation with models. Although UPPAAL is very powerful tool of timed systems analysis it does not suit us, because it does not allow to build the composition explicitly. In [14] we compare some tools that can be used for deriving the parallel composition of TFSMs, and explain why we have chosen BALM-II.

BALM-II was designed to build parallel composition of two FSMs. To be able to use this tool for TFSMs we use well-known transformation of TFSM into FSM, and in this work we create a tool for converting TFSM into automaton and integrate it into BALM-II. After deriving two automata for the given two TFSMs we construct a global automaton (using BALM-II). In work [14] we suggest two approaches for getting output delays from the composition of corresponding automata: first deals with BALM-II once again, and the second is based on analyzing of time loops in the automaton. In this work we create tool for converting global automaton into TFSM based on the second approach.

Moreover, in works [14, 15] we consider TFSMs with output delays (without timeouts). In this work we consider TFSMs with output delays and timeouts.

We use implemented tools to hold experiments. Since we describe the infinite sets of output delays by linear functions, it is important to know how often these sets of linear functions appear. The experimental results show significant amount (around 50 %) of TFSMs with infinite number of output delays. We also estimate the size of the global automaton and the composed TFSM. Unfortunately the upper bound of the number of states in the global automaton is exponential due to fact that the automaton determinization is needed for composition. In order to get the results of the experiments in reasonable time, we throw away all examples for which the number of states in the global automaton is too huge (more than 10000 states).

We see the contribution of the paper as three points. First, the algorithm for deriving TFSM from the given global automaton (for the case, when TFSMs have both output delays and timeouts). Second, new tools that allow to derive the binary parallel composition of TFSMs automatically (taking in the mind that composition of two *automata* can be derived using BALM-II). And probably the main point, the experiments have shown, that the theory of TFSMs with linearly-countable output delays is worth to be developed.

The outline of the paper is as follows. In Section II some preliminaries are given. In Section III we describe the structure of the composition that we consider in our work, and how the components communicate with each other. Section IV is devoted to one of the implemented tools which allow to derive TFSM based on the global automaton; we discuss extraction of output delay functions from the global automaton using an

example, and propose an algorithm that is lied in the basis of the tool. Section V describes the experiments and experimental results. Section VI concludes the paper.

2. Preliminaries

A finite automaton *S* is a 5-tuple $(S, X, s_0, F, \lambda_S)$, where *S* is a finite nonempty set of states with s_0 as the initial state and $F \subseteq S$ as a set of final (accepting) states; *X* is an alphabet of actions; and $\lambda_S \subseteq S \times X \times S$ is a transition relation. In this work we consider only finite automata, so we will write simply “automaton” (meaning finite automaton). The transition relation defines all possible transitions of the automaton. The language L_S of automaton *S* is the set of all sequences α in alphabet *X*, such that in automaton *S* there is a sequence of transitions (marked by α) from the initial state to some final state. An FSM *S* is a 5-tuple $(S, I, O, s_0, \lambda_S)$, where *S* is a finite nonempty set of states with s_0 as the initial state; *I* and *O* are input and output alphabets; and $\lambda_S \subseteq S \times I \times O \times S$ is a transition relation. In FSM all states are final.

Let $\mathbb{N}$ be the set of natural numbers. Let $\mathcal{F} = \{ b + k \cdot t \mid b, k \in \{0\} \cup \mathbb{N} \}$ – the set of all possible linear functions. TFSM [12] is an FSM with timeouts and output delays $S = (S, I, O, s_0, \lambda_S, \Delta_S, \sigma_S)$, where 5-tuple $(S, I, O, s_0, \lambda_S)$ is underlying FSM, Δ_S: $S \rightarrow S \times (\mathbb{N} \cup \{\infty\})$ is a timeout function that determine maximal time of waiting for input symbol, σ_S: $\lambda_S \rightarrow (2^{\mathcal{F}} \setminus \varnothing)$ is an output delay function that determine for each transition time delay for producing output symbol (output timeout).

The semantics of Timed FSM is as follows. We describe the behavior of a system that has time aspects: timeouts and output delays. Timeouts describe the situation when the system comes from one state to another not under the input symbol, but in the case when no inputs are applied during some period of time. In practice it’s the case of waiting for the password in internet-banking, etc. As for output delays, the meaning of them is that the output symbol is produced for the given input symbol not immediately but after some period of time. For example, a light can change not immediately after a button is pushed but after some time.

We suppose that there is a global clock (timed variable) and this clock is reset to zero when an input symbol (action) is applied, when an output symbol is produced and when the state of the system is changed (for example, in the case of transition under timeout).

3. Composition of Timed Finite State Machines

Parallel composition describes a dialog between two components. The structure of the composition is presented in Fig. 1.

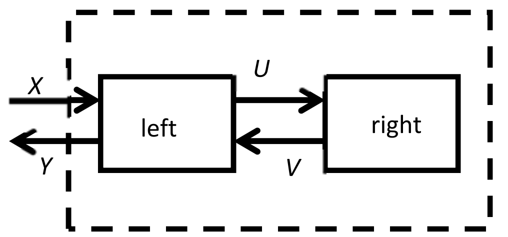

Рис. 1. Структура бинарной параллельной композиции
Fig. 1. Structure of binary parallel composition

We suppose that the system works in “slow environment” (it means that the next input can be applied to the composition only after it produces external output to the previous input), the alphabets of different channels don’t intersect and there are no infinite dialogs under internal inputs (it means no livelocks). We also suppose that each component and the whole composition have timed variables. The values of these variables are increasing synchronously, and they reset when the system gets an input or when the state is changed.

In order to compose two TFSMs using BALM-II, we need, first of all, to derive the corresponding automaton for each TFSM [12, 13]. In this work we implement a tool for this step and integrate it into BALM-II as a new command TFSM2AutV1. This implementation requires, that MV description of TFSM (BALM-II format) contains special variable called *Time*. Domain of the variable *Time* contains only non-negative integers which are used to describe timeouts and delays. For example, if a table of transitions has head as follows

```
.table I Time O CS -> NS
```

and we would like to represent timeout transition $s_1 \overset{t}{\rightarrow} s_2$, then it will appear as

```
^ t ^ s₁ s₂
```

where *I* – is the variable for input action, *O* – is the variable for output action, *CS* – is the variable for the current state, *NS* – is the variable for the next state, *t* – is some non-negative integer from the domain of the variable *Time* and the symbol ^ represents the fact, that there is no action in corresponding channel. The ordinary transition with delay, like $s_1 \xrightarrow{i/o(d)} s_2$ is described as

```
i d o s₁ s₂
```

where *i* – is from the domain of *I*, *o* – is from the domain of *O*, and *d* – is from the domain of *Time*.

Then we derive parallel composition of two automatons using BALM-II (we describe how to do this in works [14, 15]). The resulting automaton is so-called global automaton and it describes the common behavior of two automata that are working together in a dialogue mode.

After deriving a global automaton that describes the common behavior of two given TFSMs we need to construct the corresponding TFSM. We also develop a tool for this step and describe the corresponding algorithm in the next section.

4. Deriving TFSM Based on the Global Automaton. Extracting Output Delays Functions

Let’s consider an example of a global automaton (Fig. 2) and describe how to derive the corresponding TFSM (Fig. 3). One can see that after Request there can be output Deliver after 3 + 5*t* or 4 + 5*t* tick counts, where *t* is arbitrary non-negative integer number.

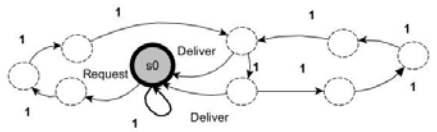

Рис. 2. Пример глобального полуавтомата

Fig. 2. An example of global automaton

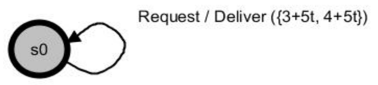

Рис. 3. Соответствующий временной автомат
Fig. 3. Corresponding TFSM

In work [14] we propose a procedure for deriving TFSM based on the global automaton for the case when the given TFSMs have only output delays (no timeouts). In this work we propose more common algorithm that works also for the case when the given TFSMs have both output delays and timeouts. The idea of this algorithm is very simple. According to the theory the final states of the global automaton correspond to the states of TFSM. Every sequence, which starts and finishes at some final states of the automaton and goes through non-final states, corresponds to the transition of the TFSM. The sequence can start only with input action or with special action 1. If this sequence starts with special action 1, then every action of the sequence is 1 and corresponding transition is timeout transition (timeout is the number of 1^s needed to reach final state). If the first action of the sequence is input action, then the last action is output action and the intermediate actions are 1^s. In this case the corresponding TFSM transition is ordinary input-output transition with delay. The delay – is the number of 1^s in-between the input and the output actions of the sequence. We just need to keep in mind, that sequence of 1^s may form a loop. So we do some precautions to detect loops when traversing the automaton transitions. The number of 1^s before the loop gives us *b* for linear function and the length of the loop gives *k* for the function.

Algorithm 1. Deriving TFSM based on the global automaton.

Input. Global automaton $A = \langle A, I \cup O \cup \{1\}, a_0, F, \lambda_A\rangle$

Output. TFSM $T = \langle T, I, O, t_0, \lambda_T, \Delta_T, \sigma_T\rangle$ with the same behavior.

```
$t_0 \equiv a_0$; $T := \{t_0\}$;
FOREACH non-visited state $t$ from $T$ DO
    IF $\exists \langle t, 1, t'\rangle \in \lambda_A$ THEN DO
        ADD $t'$ in $T$;
        IF $t == t'$ THEN ADD $\langle t, \infty, t'\rangle$ in $\Delta_T$;
        ELSE ADD $\langle t, 1, t'\rangle$ in $\Delta_T$;
    DONE
    FOREACH input action $i$ such, that $\exists \langle t, i, t'\rangle \in \lambda_A$
    DO
        $b := 0$; $V := \varnothing$;
        WHILE $t'$ != NULL AND $t'$ is NOT visited
        DO
            $t'.b := b$;
            FOREACH output action $o$ such, that $\exists \langle t', o, t''\rangle \in \lambda_A$
            DO
                IF $t'' \in F$ THEN DO
                    ADD $t''$ in $T$;
                    ADD $\langle t', o, t''\rangle$ in $V$;
                DONE
            DONE
            mark $t'$ as visited;
            $b$++;
            IF $\exists \langle t', 1, t''\rangle \in \lambda_A$ THEN $t' := t''$;
            ELSE $t'$ := NULL;
        DONE
        IF $t'$ == NULL THEN $k := 0$, $n_{\text{loop}} := \infty$;
        ELSE $k := b - t'.b$, $n_{\text{loop}} := t'.b$;
        FOREACH $\langle t', o, t''\rangle$ in $V$ DO
            ADD $\langle t, i, o, t''\rangle$ in $\lambda_T$;
            IF $t'.b < n_{\text{loop}}$ THEN
                ADD $\langle\langle t, i, o, t''\rangle, t'.b\rangle$ in $\sigma_T$;
            ELSE
                ADD $\langle\langle t, i, o, t''\rangle, t'.b + k*x\rangle$ in $\sigma_T$;
        DONE
```

```
        DONE
        mark $t$ as visited;
DONE
```

5. Experimental Results

We conduct the experiments according to the following steps:

Step 1. Generate two complete nondeterministic observable timed FSMs: `left.fsm` and `right.fsm`. At this step we use FSM generator from the tool [17].

Step 2. Convert generated TFSMs into AUT-format (BALM-II format). After this step we have two files in AUT-format: `left.aut` and `right.aut`.

The number of all states in the automaton is $S + S*I*D + S*T$, where S – is the number of states in original TFSM, I – number of input symbols, D – maximal delay, T – maximal finite timeout. The number of stable states is $S*T$.

If we have no timeouts then $T = 0$ and the number of states of the automaton is $S + S*I*D$.

Step 3. Convert files `left.aut` and `right.aut` with TFSMs into files `left_aut.aut` and `right_aut.aut` with corresponding automata. In order to do this, we created the tool and integrated it into BALM-II. We described the algorithm for this transformation in the work [14]. In this work we only add in that algorithm the transformation for timeout transitions.

Step 4. Derive the global automaton. We derive the global automaton using the same sequence of BALM-II commands as we described in work [14].

The number of final states in the product automaton is $S1*T1*S2*T2$, where $S1$ is the number of final states in the left component, $S2$ is the number of final states in the right component, $T1$ is the maximal finite timeout in the left component, $T2$ is the maximal finite timeout in the right component.

After the restriction we will have the global automaton with at most $2^{S1*T1*S2*T2} - 1$ states (since the `restriction` command includes determinization of the automaton).

If we have no timeouts then the number of final states in the product automaton is at most $S1*S2$ and after restriction we have at most $2^{S1*S2} - 1$.

Step 5. Derive TFSM based on the global automaton. For this step we created the tool based on the algorithm that was proposed in the previous section.

We generated one hundred pairs of TFSMs for each set of parameters values (number of states, maximal time delays and timeouts). In order to get results of the experiments in reasonable time we fixed the number of inputs and outputs for each channel to 2. We also need to mention that in experiments we did not consider global automata with the huge number of states (more than 10000). It means that we have thrown away such examples. The reason is that the upper bound of the number of states in the global automaton is exponential because of determinization used during composition.

The experimental results are represented in Table 1. In the 3rd and 4th columns there are percentages of TFSMs with infinite number of output delays (we need to use linear functions for describing output delays). The difference is that for the 3rd column we calculated percentage of such TFSMs for the case when the components are TFSMs with timeouts and output delays, and for the 4th column – only output delays (no timeouts). First of all, we would like to comment on dashes (‘-’) in the 3rd and 4th columns. In those cases we could not conduct the experiments for the given parameters in the reasonable time and the reason is the exponential upper bound of the state’s number in the global automaton. For example, let’s consider the last row in the Table 1: we have 4 states in the left TFSM and 4 states in the right TFSM, the maximal finite timeout for the both is the same and it is equal to 7. According to our experiments’ procedure, we first derive the corresponding automata for the given TFSMs. The number of final states in the automata is $S*T$, where S is the number of states in original TFSM, T – maximal finite timeout. So for our case the number of states in the automaton for the left component (let’s denote it as $S1$) will be equal to the number of states in the automaton for the right component (let’s denote it as $S2$) and $S1 = S2 = = 4*7 = 28$. So, each automaton will have 28 stable states. Then, we estimate the number of states in the product automaton as $S1*T1*S2*T2$, where $T1$ is the maximal finite timeout in the left component, $T2$ is the maximal finite timeout in the right component, so, for our case the number of states in the product automaton will be 28*7*28*7 = 38416. After the `restriction` we will have the global automaton with at most $2^{S1*T1*S2*T2} – 1$ states since the command restriction does determinization of the automaton, and for the last row in our table in the worst case it can be $2^{38416} – 1$ states and of cause it’s too huge automaton to deal with.

Табл. 1. Экспериментальные результаты
Table 1. Experimental results

Number of states	***Maximal delay / timeout***	***Percent of TFSMs with infinite number of output delays (with timeouts)***	***Percent of TFSMs with infinite number of output delays (without timeouts)***
2	2	38	23
3	2	39	37
4	2	43	28
5	2	34	-
2	3	47	38
3	3	56	42
4	3	66	55
2	4	46	42
3	4	61	47
4	4	63	53
2	5	67	36
3	5	-	51
4	5	34	69
2	6	-	52
3	6	-	54
4	6	-	68
2	7	-	51
3	7	-	50
4	7	-	75

According to our experimental results, around 50 % of TFSMs, that describe the behavior of the composition, has the infinite number of output delays, so, further investigations of such compositions are needed. It is an actual task especially for the case of cascade composition [15], when each component is a TFSM with timeouts and final sets of output delays, and we first compose two internal components and then we need to compose the resulting TFSM with the remaining part of the system. However, according to our experimental results, this resulting TFSM has infinite number of output delays with high probability. So, more investigations of such compositions are needed.

6. Conclusions

This paper is devoted to parallel composition of Timed Finite State Machines (TFSMs). We consider the composition of TFSMs with transitions under timeouts and output delays. It is known that even for the case when output delays are the finite sets of nonnegative integers, the result of such composition can be a TFSM with infinite set of output delays, and we describe such infinite sets by linear functions. It is important to know how often these sets of linear functions appear in order to estimate the importance of future investigations such compositions (especially for deriving cascade composition). In order to conduct the experiments we created two tools: the first one for converting TFSM into automaton (we integrated it into BALM-II), the second one for converting the global automaton into TFSM. The experimental results show significant amount (around 50 %) of TFSMs with infinite number of output delays, so, further investigations of such compositions are needed. We also estimate the size of global automaton and the composed TFSM. In experiments we do not consider global automata with the huge number of states (more then 10000).

The reason is that the upper bound of the number of states in the global automaton is exponential because of determinization used during composition. We plan to propose another approach for deriving the composition of Timed Finite State Machines. It will be the part of our future work.

References

[1]. Gill A. Introduction to the theory of finite state machines, New-York, McGraw-Hill, 1962.
[2]. N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli. Solution of parallel language equations for logic synthesis. In The Proceedings of the International Conference on Computer-Aided Design. 2001. pp. 103–110.
[3]. G. Castagnetti, M. Piccolo, T. Villa, N. Yevtushenko, A. Mishchenko, Robert K. Brayton. Solving Parallel Equations with BALM-II. Technical Report No. UCB/EECS-2012-181, Electrical Engineering and Computer Sciences University of California at Berkeley. 2012. [Electronic resource] http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.pdf (date of access: 21.04.2016).
[4]. Gregorio Diaz, Juan-Jos e Pardo, Mar a-Emilia Cambronero, Valent n Valero, and Fernando Cuartero. Automatic Translation of WS-CDL Choreographies to Timed Automata, volume 3670 of Lecture Notes in Computer Science, book section 17, pages 230{242. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28701-8. doi: 10.1007/11549970 17
[5]. M. Lallali, F. Zaidi, and A. Cavalli. Timed modeling of web services composition for automatic testing. In Signal-Image. Technologies and Internet-Based System, 2007. Bibliography 102 SITIS '07. Third International IEEE Conference on, pages 417- 426, Dec 2007. DOI: 10.1109/SITIS.2007.110.
[6]. R. Alur and D. L. Dill. A theory of timed automata. Theoretical computer science. 1994. Vol.126, Iss. 2. pp. 183–235.
[7]. Springintveld J., Vaandrager F. and D'Argenio P. Testing timed automata. Theoretical Computer Science, 254 (1-2). pp. 225-257, 2001.
[8]. Kushik N., Lopez J., Cavalli A., Yevtushenko N. Improving Protocol Passive Testing through 'Gedanken' Experiments with Finite State Machines. Proceedings 2016 IEEE International Conference on Software Quality, Reliability and Security, pp. 315-322.
[9]. Hierons R., Turker U. Parallel Algorithms for Testing Finite State Machines: Generating UIO Sequences. IEEE Transactions on Software Engineering, 42(11),7429774. pp. 1077-1091, 2016.
[10]. K. El-Fakih, M. Gromov, N. Shabaldina, N. Yevtushenko. Distinguishing Experiments for Timed Non-Deterministic Finite State Machines. Acta Cybernetica. 2013. Vol. 21, № 2. pp. 205–222.
[11]. Tvardovskii A., Yevtushenko N. Minimizing timed Finite State Machines. Vestnik TGU [The Bulletin of TSU], 2014. Vol. 4 (29). pp. 77-82 (in Russian).
[12]. O. Kondratyeva, N. Yevtushenko, and A. Cavalli. Parallel composition of nondeterministic finite state machines with timeouts. Journal of Control and Computer Science. Tomsk State University, Russia. 2014. Vol. 2(27). pp. 73–81 (in Russian).
[13]. O. Kondratyeva, N. Yevtushenko, A. Cavalli. Solving parallel equations for Finite State Machines with Timeouts. Trudy ISP RAN / Proc. ISP RAS, vol. 26, issue 6, pp. 85–98 (in Russian). DOI: 10.15514/ISPRAS-2014-26(6)-8.

[14]. Shabaldina N., Gromov M. Using BALM-II for deriving parallel composition of timed finite state machines with outputs delays and timeouts: work-in-progress. System Informatics [Sistemnaya informatika], № 8, 2016, pp. 33-42.
[15]. Gromov M..L, Shabaldina N.V. Using balm-ii for deriving cascade parallel composition of timed finite state machines. Modeling and Analysis of Information Systems [Modelirovanie i analis inforamzionnych system], 23:3 (2016). pp. 699-712 (in Russian).
[16]. http://www.uppaal.com/ (date of access: 21.04.2016)
[17]. N. Shabaldina , M. Gromov. FSMTest-1.0: a manual for researches. Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2015). Ukraine, Kharkov: SCITEPRESS, 2015. pp. 216-219.

Эксперименты по построению параллельной композиции временных автоматов

А.П. Сотников <sotnikhtc@gmail.com>
Н.В. Шабалдина <nataliamailbox@mail.ru>
М.Л. Громов <maxim.leo.gromov@gmail.com>
Томский государственный университет,
634050, Россия, г. Томск, пр. Ленина, д. 36

Аннотация. В данной работе мы продолжаем наши исследования параллельной композиции временных конечных автоматов. Мы рассматриваем композицию временных автоматов с таймаутами и задержками выходных символов. Для того чтобы оценить, насколько часто в параллельной композиции недетерминированных временных автоматов (с таймаутами и без таймаутов) возникают бесконечные множества задержек выходных символов, мы провели компьютерные эксперименты. Для проведения таких экспериментов мы реализовали два инструмента: первый позволяет преобразовать временной конечный автомат в полуавтомат (данный инструмент встроен в BALM-II), второй позволяет преобразовать глобальный полуавтомат композиции во временной автомат. Ориентируясь на известные работы по данной тематике, мы описываем бесконечные множества задержек выходных символов конечным образом, а именно, при помощи линейных функций, и нужно знать, как часто такое множество линейных функций возникает, чтобы оценить важность дальнейших исследований параллельной композиции временных автоматов (особенно случая каскадной композиции). Результаты экспериментов показали, что в значительном количестве случаев (около 50 %) временной автомат композиции содержит бесконечное множество задержек выходных символов. Кроме того, мы оценили размер глобального полуавтомата и автомата композиции. При проведении экспериментов мы не рассматривали глобальные полуавтоматы с большим числом состояний (более 10000).

Ключевые слова: временные конечные автоматы; параллельная композиция; BALM-II.

DOI: 10.15514/ISPRAS-2017-29(3)-13

Для цитирования: Сотников А.П., Шабалдина Н.В.,, Громов М.Л. Эксперименты по построению параллельной композиции временных автоматов. Труды ИСП РАН, том 29, вып. 3, 2017 г., стр. 233-246. DOI: 10.15514/ISPRAS-2017-29(3)-13

Список литературы

[1]. Gill A. Introduction to the theory of finite state machines, New-York, McGraw-Hill, 1962.

[2]. N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli. Solution of parallel language equations for logic synthesis. In The Proceedings of the International Conference on Computer-Aided Design. 2001. pp. 103–110.

[3]. G. Castagnetti, M. Piccolo, T. Villa, N. Yevtushenko, A. Mishchenko, Robert K. Brayton. Solving Parallel Equations with BALM-II. Technical Report No. UCB/EECS-2012-181, Electrical Engineering and Computer Sciences University of California at Berkeley. 2012. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.pdf (дата доступа 21.04.2016).

[4]. Gregorio Diaz, Juan-Jos e Pardo, Mar a-Emilia Cambronero, Valent n Valero, and Fernando Cuartero. Automatic Translation of WS-CDL Choreographies to Timed Automata, volume 3670 of Lecture Notes in Computer Science, book section 17, pages 230{242. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28701-8. doi: 10.1007/11549970 17

[5]. M. Lallali, F. Zaidi, and A. Cavalli. Timed modeling of web services composition for automatic testing. In Signal-Image. Technologies and Internet-Based System, 2007. Bibliography 102 SITIS '07. Third International IEEE Conference on, pages 417- 426, Dec 2007. DOI: 10.1109/SITIS.2007.110.

[6]. R. Alur and D. L. Dill. A theory of timed automata. Theoretical computer science. 1994. Vol.126, Iss. 2. pp. 183–235.

[7]. Springintveld J., Vaandrager F. and D’Argenio P. Testing timed automata. Theoretical Computer Science, 254 (1-2). pp. 225-257, 2001.

[8]. Kushik N., Lopez J., Cavalli A., Yevtushenko N. Improving Protocol Passive Testing through 'Gedanken' Experiments with Finite State Machines. Proceedings 2016 IEEE International Conference on Software Quality, Reliability and Security, pp. 315-322.

[9]. Hierons R., Turker U. Parallel Algorithms for Testing Finite State Machines: Generating UIO Sequences. IEEE Transactions on Software Engineering, 42(11),7429774. pp. 1077-1091, 2016.

[10]. K. El-Fakih, M. Gromov, N. Shabaldina, N. Yevtushenko. Distinguishing Experiments for Timed Non-Deterministic Finite State Machines. Acta Cybernetica. 2013. Vol. 21, № 2. pp. 205–222.

[11]. А.С. Твардовский, Н.В. Евтушенко. К минимизации автоматов с временными ограничениями. Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика, 2014. № 4 (29), стр. 77-82

[12]. О.В. Кондратьева, Н.В. Евтушенко, А.Р. Кавалли. Параллельная композиция конечных автоматов с таймаутами. Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика, 2014. № 2 (27), стр. 73–81

[13]. О.В. Кондратьева, Н.В. Евтушенко, А.Р. Кавалли. Решение автоматных уравнений для временных автоматов относительно параллельной композиции. Труды ИСП РАН, том 26, вып. 6, стр. 85–98

[14]. Shabaldina N., Gromov M. Using BALM-II for deriving parallel composition of timed finite state machines with outputs delays and timeouts: work-in-progress. Системная информатика, № 8, 2016, pp. 33-42.

[15]. Громов М. Л., Шабалдина Н. В. Построение каскадной параллельной композиции временных автоматов в balm-ii. Моделирование и анализ информационных систем, Т. 23, No 6 (2016), стр. 699-712.

[16]. http://www.uppaal.com/ (дата доступа 21.04.2016)

[17]. N. Shabaldina, M. Gromov. FSMTest-1.0: a manual for researches. Proceedings of IEEE East-West Design & Test Symposium (EWDTS’2015). Ukraine, Kharkov: SCITEPRESS, 2015, pp. 216-219.