UDC 519.713

Using BALM-II for deriving parallel composition of timed finite state machines with outputs delays and timeouts: work-in-progress

Shabaldina N. (Tomsk State University),

Gromov M.(Tomsk State University)

In this paper we consider a procedure of parallel composition construction of Timed Finite State Machines (TFSMs) using BALM-II and suggest different ways of getting linear functions that describe a set of output delays. Our research consists of three steps: at first step we consider composition of TFSMs when an output delay may be a natural number or zero; at second – we add transitions under timeouts; at third we consider composition of TFSMs in general case (when output delays are described as sets of linear functions). This paper is devoted only to the first step of the research.

Keywords: *Timed finite state machines, parallel composition, BALM-II.*

1. Introduction

Most modern applications, such as web-services, telecommunication protocols, are oriented on interaction with each other. The classical model for a discrete system is Finite State Machine (FSM). If the behavior of each system is described by an FSM, then their common work can be described by their composition (that also will be an FSM under appropriate assumptions) [1,2]. In this work we are interested in so-called parallel composition [1], when the interacting systems work asynchronously in as-sumption of slow environment, and for deriving such FSM composition there is a tool named BALM-II (Berkeley Automata and Language Manipulation)[2].

Sometimes it is necessary to take into account time aspects of a discrete system. Probably the most general way to describe such a system is Timed Automaton [3]. However, in this work we are interested in input-output reactive systems, when every input action is necessary followed by output action, probably after some time. The class of such systems has been already mentioned, it includes telecommunication protocols, sequential circuits, web-services etc. In this case we can use Timed Finite State Machine (TFSM) as a model. There exist different ways to introduce Timed FSM, for example, with timed guards on transitions [4]. In this work we consider TFSM with output delays and timeouts [5,6]. We got inspiration for our research from the work [5], in which authors describe

how to build parallel binary composition of two timed FSMs with output delays and timeouts. In order to derive the composition of timed FSMs the corresponding automaton should be built [5]. First we transform both TFSMs into automaton, then we compose them, and then we need to go back to the TFSM model. In [5] it is shown that composition of two Timed FSMs can have infinite number of output delays for a given transition and those delays can be de-scribed by a finite set of linear functions $\{ b + k \cdot t \mid b, k \in \{0\} \cup \mathbb{N} \}$.

There are several tools dealing with timed automata, their composition and verification. One of the most popular is UPPAAL [7]. It allows to describe timed system using Extended Timed Automata, as well as composition of such systems. One of the key feature of UPPAAL is built-in verifier. Unfortunately, UPPAAL does not build composition explicitly and one of the objectives for this work is to get composition explicitly for further processing (for example, test generation). For that reason in this work we decided to use BALM-II since it was designed to build parallel binary com-position of two FSMs. To be able to use this tool for Timed FSMs we use well-known transformation of TFSM into FSM and then into common automaton by introducing new (tick) action. Also we suggest two approaches for extracting functions $f(t) = b + k \cdot t$ from the composition of corresponding automata in order to derive TFSM. First approach is based on using BALM-II once again. And the second one is to find corresponding loops in the transition graph of the automaton composition.

This work is partially supported by the basic part of the State Assignment of the Ministry of Education and Science of the Russian Federation (Project code No. 1975) and by the grant of Russian Fund for Basic Research No. 15-58-46013 CT_a.

2. Preliminaries

An 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. 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. TFSM [5] 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: $S \to S \times (\mathbb{N} \cup \{\infty\})$

is a timeout function that determine maximal time of waiting for input symbol, σ_S: $\lambda_S \to (\{0\} \cup \mathbb{N})$ is an output delay function that determine for each transition time delay for producing output (output timeout).

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

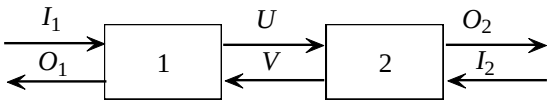


Fig. 1. Structure of binary parallel composition

We suppose that we have “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 synchronically, and they reset when the system gets an input or when the state is changed.

3. Deriving an automata based on the given TFSMs

In order to derive the composition of timed FSMs we can use the corresponding automaton [5]. For deriving an automaton that corresponds to the classical FSM we need to do the following steps [2]:

1. Derive the set of states that contains all FSM states (final, or accepting states) and a number of intermediate (not-final) states (one new state for each transition in FSM). The initial state of the automaton is the same as the initial state of the FSM.
2. Derive the set of actions $X = I \cup O$.
3. Derive the set of transitions: for each FSM transition we add two transitions in automaton, i.e. $(s,\ i,\ o,\ s')$ generates $\{(s, i, s''), (s'', o, s')\}$, where s'' is one of the intermediate states we added at the first step which corresponds to the transition under consideration.

So, in order to construct an FSM from the given automaton, we need to split alphabet of actions into input alphabet and output alphabet, merge transitions and delete intermediate states.

In order to derive an automaton for the timed FSM with timeouts and output delays we first apply steps that are described above. Then we need to add into the set of actions a new special symbol $1 \notin I \cup O$ that corresponds to tick count and represents an action “to wait for one time unit”[5]. We add in each final state a loop under 1 (in order to describe the situation that the current component is

waiting for input and time variable of the other component is increasing). Then, we replace the transition under timeouts by a chain of transitions under 1 (in order to model the time delay), the length of the chain corresponds to the value of time delay. And we do almost the same by adding the chain of transitions under 1 between an input and output symbols (if there is an output delay for this transition in TFSM) [5].

In Figures 2 and 3 one can see TFSMs that describe the behavior of left and right components of the composition, correspondingly. We take this example from work [5]. In Figures 4 and 5 we show automata for these TFSMs. In this example the structure of the composition is simpler then in Figure 1 (left component has external input Request and external output Deliver; right component has no external inputs and external outputs) and also TFSMs are simpler: they have output delays, however, they have no transitions under timeouts.

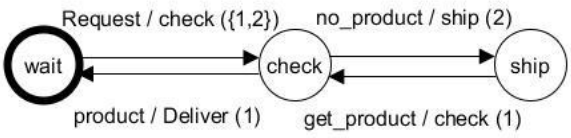


Fig. 2. Left-part component (TFSM)

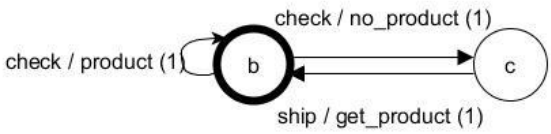


Fig. 3. Right-part component (TFSM)

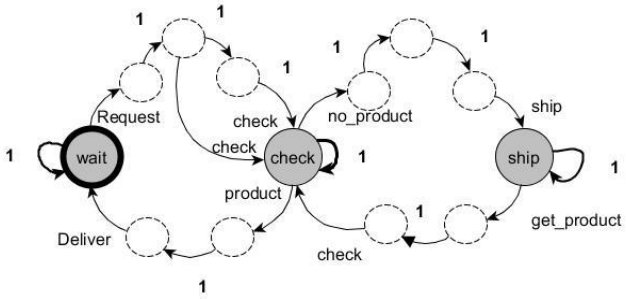


Fig. 4. Left-part component (automaton)

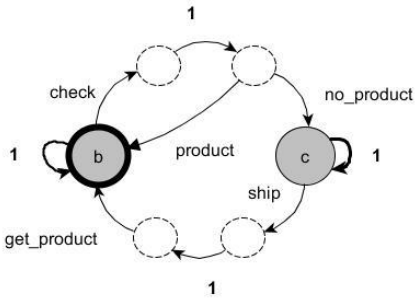


Fig. 5. Right-part component (automaton)

4. Deriving automata parallel composition using BALM-II

In this section we describe how to derive a binary parallel composition of two automata using BALM-II, and we illustrate this procedure using our example from previous Section.

BALM-II supports `AUT` file format for describing automata [2]. This format is a restricted form of `BLIF_MV` format. Due to the restriction of space we just mention the most important things. First of all, we need to determine our channels, in AUT format it will be like this for the left component:

```
.inputs x v u y t E
```

We underline that in addition to the channels of the left component that you can see in Figure 1, we need to mention the special time channel (channel *t*) that correspond to the timed variable (or to our special action 1). As for the channel *E*, this is also a special channel that determine which one from the channels *x*, *v*, *u*, *y* and *t* is active now (while the other channels are inactive).

For the time channel *t* we need to introduce in addition to the input 1 one more input (due to the fact that we need at least two values for the channel alphabet in BALM-II):

```
.mv t 2 1 none
```

When we have our automata in `AUT` format, the first thing we need to do is synchronizing channels of the composing automata:

```
chan_sync x|v|u|y|t|E u|v|t|E left_timed.aut right_timed.aut
left_t_sync.aut right_t_sync.aut
```

Then, according to the algorithm of deriving the composition of two automata [1,2], we need to extend the alphabet of the right-component automaton to the channels *X* and *Y*:

```
expansion E0,E3 right_t_sync.aut right_t_exp_aut
support x,v(3),u,y,t,E(5) right_t_exp.aut right_t_support.aut
```

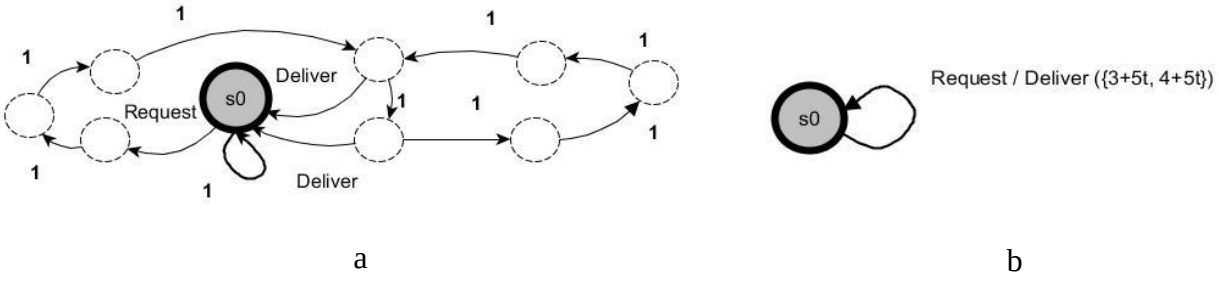
The next step is deriving an intersection of two automata:

```
product left_t_sync.aut right_t_support.aut product_timed.aut
```

Now we have an automaton that describes common behavior of left and right components, but its behavior does not always correspond to our “slow environment” restriction, and in this case we need to intersect derived automaton with the automaton that represent the language (*X*(*UT***V*)**T***Y*)*. In our example we don’t need to do this. So the next step is to restrict the automaton to external channels and special timed channel:

```
restriction E0,E3,E4 product_timed.aut restriction_timed.aut
support x,y,t,E(3) restriction_timed.aut comp_timed.aut
```

The result is shown in Figure 6 (a). 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. So in Figure 6 (b) you can see corresponding TFSM.



a b

Fig. 5. The composed automaton (a) and Timed FSM (b)

5. Deriving parallel composition of two TFSMs with output delays: two approaches for extracting output delays functions

In this section we propose two approaches for extracting a set of linear functions from the derived automaton.

5.1. Using BALM-II for extracting output delays functions

The idea of this approach is to intersect consequently the resulting automaton with the automata that correspond to the languages $X1^b(1^k)^*Y$, i.e., the languages with the following property: they contain sequences that start with any external input symbol, the end of the sequences is any external output symbol, and between these input and output there is subsequence that corresponds to the function $\{b + k\,t \mid b, k \in \{0,1,..,n\}$.

We need to mention that in this case we need to intersect not only the composition automaton, but also its modifications that can be derived by making each accepting state as an initial state (one by one). So we fix b and k and intersect automaton with the language $X1^{b+k}(1^k)^*Y$ with the composition, fixing in the composition automaton an initial state (we consider the automaton with the language $X1^{b+k}(1^k)^*Y$ instead of the language $X1^b(1^k)^*Y$ in order to avoid the case when in the composition automaton there is a chain that corresponds to 1^b and then no loop, i.e. the case when t can only be equal to zero). Then we test the intersection using `check_nb` BALM-II command. This command allows answering the question: whether we can extract output delays function for the fixed b and k or not. If in the composition automata between input and output there is a subsequence that corresponds to the function $b + k\cdot t$, then the corresponding intersection will be nonblocking, it means it has no deadlocks; otherwise, it will be blocking, so, the intersection will contain no external output after some sequence under 1. For our example the intersection (product) of composition automaton and the automaton with the language $X1^3(1^5)^*Y$ will be nonblocking, the intersection with the automaton with the language, for example, $X1^2(1^5)^*Y$ will be blocking.

5.2. Getting output delays procedure based on analyzing cycles in automaton

Let us notice some properties of automata, derived from TFSMs:

1. Every transition, marked with input symbol, starts at final state and ends at non-final state.
2. Every transition, marked with output symbol, starts at non-final state and ends at final state.
3. Every transition, marked with 1 (a tick count), starts at non-final state and ends at non-final state.
4. If there are several non-final states s_1, …, s_k, such that $(s_i, 1, s_{i+1}) \in \lambda_S$, $i = 1, \ldots, k-1$ (continuous non-final chain of transitions marked by 1), then $s_i \neq s_j$, for every i and j, $i \neq j$ (there are no time loops, Figure 7 (a)).

However, as it was shown with the example in previous Sections, when we have parallel composition of two TFSMs, the resulting automaton may have continuous non-final time chain with a loop (Figure 7 (b)). Nevertheless, there cannot be intermediate time loops, i.e. loops with outgoing edge that is marked by 1 (Figure 7(d)) or several (Figure 7(e)) time loops. We shall prove this by the following proposition.

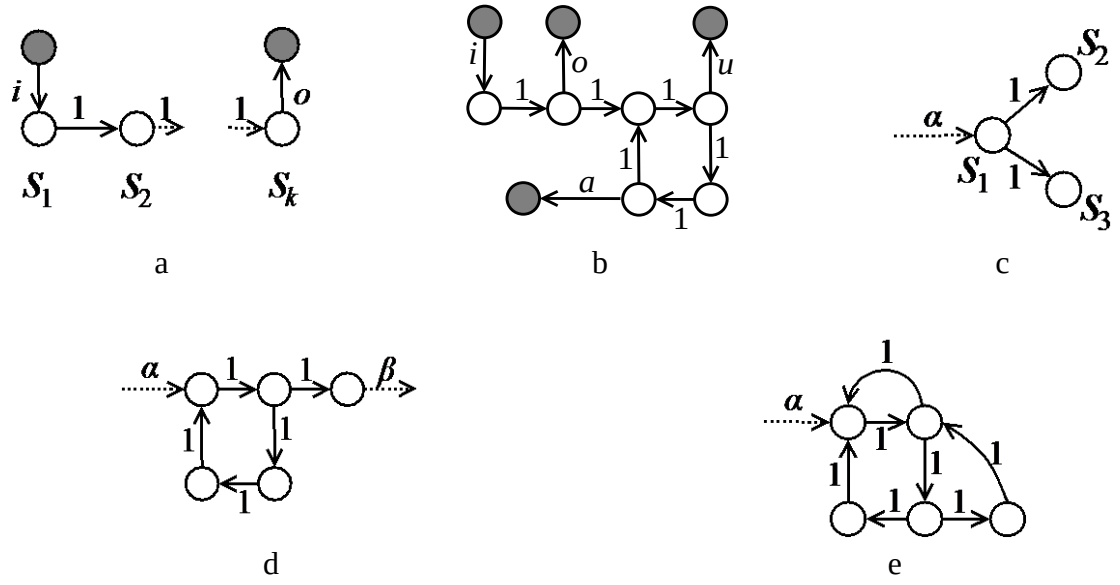


Fig. 7. Time chains. Here i – input symbol, o, u, a – output symbols, final states marked gray and non-final are blank

Proposition 1. Given automaton A, describing parallel compositions of two TFSMs P and Q. There are no states with more than one outgoing transition, marked by 1.

Proof. *Indeed, suppose there is such a state (Figure 7(c)), reachable by sequence α. It means, that by construction in automaton A_P there is state p reachable by α and automaton A_Q there is state q reachable by α as well, such that either p has two different outgoing transitions marked by*

1, or q has two different outgoing transitions marked by 1, or both of them have such transitions. Neither of listed is possible. □

Corollary 1.1. There cannot be intermediate time loop in any continuous time chain of an automaton, describing TFSM parallel composition.

Corollary 1.2. There cannot be more than one time loop in in any continuous time chain of an automaton, describing TFSM parallel composition.

Corollary 1.3. There cannot be more than one state in continuous time chain with more than one ingoing transitions, and the number of ingoing transitions is not more than two (Figure 7(b)).

Now we describe a procedure for counting output delays. In this procedure we shall use sets Q_{siop}. Each set Q_{siop} contains functions (constant or linear) of output delays for transition (*s*, *i*, *o*, *p*). We notice that the estimation of the procedure is ~*N*, where *N* is the number of states in automaton *A*, describing parallel composition of two TFSMs.

Procedure 1. Getting output delays.

Input. Automaton *A*, describing parallel composition of two TFSMs.

Output. Set of sets of output delays for every input-output pair possible in composition.

1. Get next final state *s* of automaton *A*. **IF** they are over, **THEN END.**
2. Get next outgoing transitions of *s*. **IF** they are over, **THEN GOTO Step 1.** Let outgoing transition be marked with input symbol *i*, and the next state of the transition be *s*1.
3. *s*curr := *s*1; *b* := 0; *k* := 0.
4. **IF** *s*curr has more than one ingoing transition **THEN** *k* := countLoopLength(*A*, *s*curr)**(Procedure 2).**
5. **FOR** every transition (*s*curr, *o*, *p*) ∈ λ_S, where λ_S is transition relation of *A*, *o* is output symbol, and *p* is final state of *A*, **DO** put function *b* + *k***t* in *Qsiop*.
6. **IF** there is transition (*s*curr, 1, *s′*) ∈ λ_S, **THEN** *s*cur := *s′*.
7. **GOTO Step 2.**

Procedure 2. countLoopLength

Input. Automaton *A* and non-final state *s* of *A*.

Output. Length of a loop, containing *s*, or *N* + 1, if there is no such loop, where *N* – number of all states in *A*.

1. s_{curr} := s; k := 0.
2. **IF** there exists transition (s_{curr}, 1, s') ∈ λ_s, **THEN** s_{curr} := s', k := k + 1.
   ELSE RETURN N + 1. **END.**
3. **IF** s_{curr} == s, **THEN RETURN** k, **END.**

6. Conclusion and Future Research Work

In this paper, we consider the procedure of parallel composition construction of TFSMs using BALM-II and investigate different ways of extraction the set of linear functions (that describe an infinite set of output delays) from the composition of corresponding automata. This is work in progress, so we represent here just the first step of our investigation, considering only the case of deriving the composition of TFSMs with output delays that are natural number or zero. 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 time loops in automaton. In our future work we'll consider the composition of TFSMs with transitions under timeouts and the composition of TFSMs when the output delays are infinite and represented by the set of linear functions; this can happen in cascade composition.

References

1. 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. P. 103–110.
2. 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).
3. R. Alur and D. L. Dill. A theory of timed automata // Theoretical computer science. 1994. Vol.126, Iss. 2. P. 183–235.
4. K. El-Fakih, M. Gromov, N. Shabaldina, N. Yevtushenko. Distinguishing Experiments for Timed Non-Deterministic Finite State Machines // Acta Cybernetica. 2013. Vol. 21, № 2. P. 205–222.
5. 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). P. 73–81.
6. O. Kondratyeva, N. Yevtushenko, A. Cavalli. Solving parallel equations for Finite State Machines with Timeouts // Trudy ISP RAN [The Proceedings of ISP RAS]. 2014. Vol. 26, Iss. 6. P. 85–98.

7. http://www.uppaal.com/