Separating Non-Determinisic Finite State Machines with Time-Outs

Rustam Galimullin, Natalia Shabaldina
Radiophysics department
Tomsk State University
Tomsk, Russia
nihilkhaos@gmail.com, NataliaMailBox@mail.ru

***Abstract*— In this paper we consider one of the classical finite state machine (FSM) model modifications - FSM with time-outs (or timed FSM). In this model in addition to the ordinary transitions under inputs there are transitions under time-outs when no input is applying. The behavior of many modern systems includes time-outs, for example, mobile phones, etc. In the past few years some work have been carried out on studying different relations between timed FSMs. Non-separability relation is very attractive for non-deterministic classical FSMs and FSMs with time-outs course for this relation we don't need «all weather conditions» while testing. In this paper we present and compare two approaches for building a separating sequence for two separable FSMs with time-outs. One of them is using a conversion to classical FSMs, while another one is dealing directly with timed FSMs.**

Keywords - finite state mashines with time-outs, non-deterministic finite state machines, non-separability relation, timed input sequence, separating sequence

I. INTRODUCTION

Most of the modern discrete systems, such as digital circuits, telecommunication protocols, logical games, etc., can be described as Finite State Machines (FSM). The entry of FSM receives one of the enabled inputs and returns an output. On condition it is necessary to take into account time aspects of discrete system, time function is interposed[1-4]. FSMs with introduced time function are called FSMs with time-outs or timed FSM (TFSM). Provided that input is being handled uniquely, TFSM is named **deterministic**, otherwise – **non-deterministic**. To distinguish correct and invalid TFSMs distinguishing sequences are generated. They claim exhaustive search of all TFSM's reactions to the input sequence, i.e. it is necessary to input every sequence from test suite enough times to observe all outputs of the system. Practical implementation of this assumption is almost impossible, and it's mostly used to check non-separability relation[5,6]. FSMs are separable[5], if there is an input sequence (called separating sequence), such that the sets of output sequences to this sequence doesn't intersect. In this paper two different approaches for building separating sequence for FSMs with time-outs are suggested.

II. PRELIMINARIES

Formally, **Finite State Machine** (FSM) is a quintuple $S = \langle S, I, O, s_0, \lambda_S \rangle$, where S is a finite nonempty set of states with initial state s_0, I and O – finite non-intersecting sets of inputs and outputs, $\lambda_S \subseteq S \times I \times S \times O$ – transition relation. If for each pair $(s, i) \in S \times I$ there is at least one pair $(o, s') \in O \times S$ such that $(s, i, o, s') \in \lambda_S$, FSM is called **comlete**. **FSM with time-outs** is a sextuple $S = \langle S, I, O, s_0, \lambda_S, \Delta_S \rangle$, where S is a finite nonempty set of states with initial state s_0, I and O – finite non-intersecting sets of inputs and outputs, $\lambda_S \subseteq S \times I \times S \times O$ – transition relation and $\Delta_S: S \to S \times (\mathbf{N} \cup \{\infty\})$ – time-outs function, that defines time-out for every state. The time reset operation resets the value of the TFSM's clock to zero at the execution of the transition. If TFSM, being in certain state s_1, doesn't receive input for a certain time t such that $(s_1, t, s_2) \in \Delta_S$, it transfers to the state s_2. FSM is called **observable**, if for each triple $(s, i, o) \in S \times I \times O$ there is not more, than one state s' such that $(s, i, o, s') \in \lambda_S$. FSM could be considered as FSM with time-outs where for each state $s \in S$ $\Delta_S(s) = (s, \infty)$. **Timed input** is a pair $<i, t> \in I \times Z$.

Similar to [3], in order to extend transition relation to timed inputs we add a function $time_S: S \times \mathbf{Z}_0^+ \to S$ that allows to determine TFSM's state when the clock value is equal to t based on the current state. Let's consider the sequence of time-outs $\Delta_S(s) = (s_1, T_1)$, $\Delta_S(s_1) = (s_2, T_2)$, …, $\Delta_S(s_{p-1}) = (s_p, T_p)$ such that $T_1 + T_2 + \ldots + T_{p-1} \le t$, but $T_1 + T_2 + \ldots + T_p > t$. In this case $time_S(s, t) = s_{p-1}$. If $\Delta_S(s) = (s, \infty)$, then $time_S(s, t) = s$ for each t. For each timed input $\langle i, t \rangle$ we add a transition $(s, \langle i, t \rangle, s', o)$ to λ_S, if and only if $(time_S(s, t), i, s', o) \in \lambda_S$.

Sequence of timed inputs is called **timed input sequence**. Pair α/β, where $\alpha = <i_1, t_1>, \ldots, <i_k, t_k>$, $\beta = o_1, \ldots, o_k$ is called **timed input/output (I/O) sequence (or timed trace)**, if λ_s defines a sequence of transitions $(s_0, <i_1,t_1>, o_1, s_{r1})$, $(s_{r1}, <i_2,t_2>, o_2, s_{r2}) \ldots$, $(s_{r(k-1)}, <i_k,t_k> o_k, s_{rk})$.

As usual, the TFSM S is **connected** if for each state s there exists a timed trace that can take the machine from the initial state to state s.

State s' is called $\langle i, t\rangle$ -*successor* of state s, if there exists $o \in O$ such that $(s, \langle i, t\rangle, s', o) \in \lambda_S$. The set of all $\langle i, t\rangle$-successors of state s will be denoted by $suc_S(s, \langle i, t\rangle)$, in case of $t = 0$ we denote it as $suc_S(s, i)$.

An example of TFSM that describes mp3-player behavior is given below:

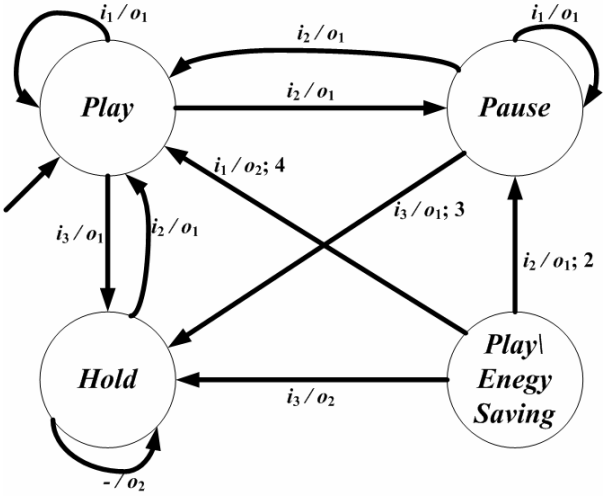

Figure 1. The TFSM that describes mp3-player behavior

The machine consists of the following states:

Play – the music is playing, player isn't in energy-saving mode (display's on);

Play\Energy Saving – the music is playing, but player is in energy-saving mode (display's off);

Pause – the music is stopped, display's on;

Hold – the music isn't playing, player's off (hold mode).

Inputs and outputs:

i_1 – player's controller is used;

i_2 – play/pause button;

i_3 – hold button;

o_1 – display's on;

o_2 – display's off.

Let us observe TFSM's behavior on timed input sequence $\alpha = \langle i_1,5\rangle\langle i_2,3\rangle\langle i_1,4\rangle$. In that case the output sequence is $\beta = o_2\, o_1\, o_2$.

A TFSM $S = \langle S, I, O, s_0, \lambda_S\rangle$ is a **submachine** of TFSM $P = \langle P, I, O, p_0, \lambda_P\rangle$ if $S \subseteq P$, $s_0 = p_0$ and each timed transition $(s, \langle i, t\rangle, o, s')$ of S is a timed transition of P.

III. Intersection of two timed FSMs

Intersection $S \cap P$ of two TFSMs $S = \langle S, I, O, s_0, \lambda_S, \Delta_S\rangle$ and $P = \langle P, I, O, p_0, \lambda_P, \Delta_P\rangle$ is the most connected sub-TFSM of $Q = (Q, I, O, q_0, \lambda_Q, \Delta_Q)$, where $Q = S \times K \times P \times K$, $K = \{0, ..., k\}$, $k = min(max\, \Delta_S(s)\downarrow_{\mathbf{N}}, max\, \Delta_P(p)\downarrow_{\mathbf{N}})$, initial state — quadruple $(s_0, 0, p_0, 0)$. Transition relation λ_Q and time-outs function Δ_Q are defined according to the following rules[3]:

1. Transition relation λ_Q contains quadruple $[(s, k_1, p, k_2), i, o, (s', 0, p', 0)]$, if and only if $(s, i, s', o) \in \lambda_S$ and $(p, i, p', o) \in \lambda_P$.
2. Time function is defined as $\Delta_Q((s, k_1, p, k_2)) = [(s', k'_1, p', k'_2), k]$, $k = \min(\Delta_S(s)\downarrow_N - k_1, \Delta_P(p)\downarrow_N - k_2)$. State $(s', k_1', p', k_2') = (\Delta_S(s)\downarrow_S, 0, \Delta_P(p)\downarrow_P, 0)$, if $\Delta_S(s)\downarrow_N = \infty$ or $\Delta_P(p)\downarrow_N = \infty$ or $(\Delta_S(s)\downarrow_N - k_1) = (\Delta_P(p)\downarrow_N - k_2)$. If $(\Delta_S(s)\downarrow_N - k_1)$, $(\Delta_P(p)\downarrow_N - k_2) \in Z_+$ and $(\Delta_S(s)\downarrow_N - k_1) < (\Delta_P(p)\downarrow_N - k_2)$, then state $(s', k_1', p', k_2') = (\Delta_S(s)\downarrow_S, 0, p, k_2 + k)$. If $(\Delta_S(s)\downarrow_N - k_1)$, $(\Delta_P(p)\downarrow_N - k_2) \in Z_+$ and $(\Delta_S(s)\downarrow_N - k_1) > (\Delta_P(p)\downarrow_N - k_2)$, then state $(s', k_1', p', k_2') = (s, k_1 + k, \Delta_P(p)\downarrow_P, 0)$.

Algorithm 1: Constructing an intersection of two TFSMs

Input: TFSMs $S = \langle S, I, O, s_0, \lambda_S, \Delta_S\rangle$ and $P = \langle P, I, O, p_0, \lambda_P, \Delta_P\rangle$

Output: TFSM $Q = (Q, I, O, q_0, \lambda_Q, \Delta_Q)$, $Q = S \cap P$

Step 1: add initial state $q_0 = (s_0, 0, p_0, 0)$ into Q.

Step 2: while set of states of TFSM Q has non-considered states, consider next in turn non-considered state q, step 3. Otherwise, **End**.

Step 3: for each input i find state $q'=(s', 0, p', 0)$ that is i-successor of state q. If the set Q doesn't include q' – add q' into the set Q, add transition $[(s, k_1, p, k_2), i, o, (s', 0, p', 0)]$ into λ_Q if $(s, i, s', o) \in \lambda_S$ and $(p, i, p', o) \in \lambda_P$.

Step 4: if there is a finite delay for state $q = (s, k_1, p, k_2)$, then:

$k := min(\Delta_S(s)\downarrow_{\mathbf{N}} - k_1, \quad \Delta_P(p)\downarrow_{\mathbf{N}} - k_2)$

If $(\Delta_S(s)\downarrow_{(\mathbf{N} \cup \{\infty\})} = \infty$ or $\Delta_P(p)\downarrow_{(\mathbf{N} \cup \{\infty\})} = \infty$ or $(\Delta_S(s)\downarrow_{\mathbf{N}} - k_1) = (\Delta_P(p)\downarrow_{\mathbf{N}} - k_2))$, then $q' = (\Delta_S(s), 0, \Delta_P(p), 0)$;

Else

if $(\Delta_S(s)\downarrow_{\mathbf{N}} - k_1) < (\Delta_P(p)\downarrow_{\mathbf{N}} - k_2)$,
then $q' := (\Delta_S(s)\downarrow_S, 0, p, k_2 + k)$;

if $(\Delta_S(s)\downarrow_{\mathbf{N}} - k_1) > (\Delta_P(p)\downarrow_{\mathbf{N}} - k_2)$,
then $q' := (s, k_1 + k, \Delta_P(p)\downarrow_P, 0)$;

Extend function Δ_Q: $\Delta_Q(q) = (q', k)$;

If the set Q doesn't include q', then add q' into Q. **Step 2**.

The intersection of two TFSMs S and P (Figures 2,3) is presented in Figure 4.

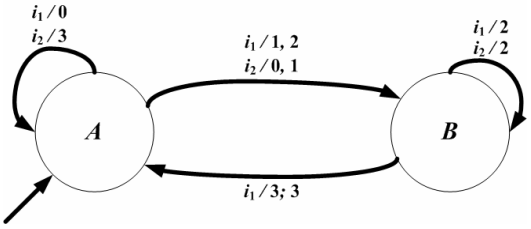

Figure 2. TFSM S

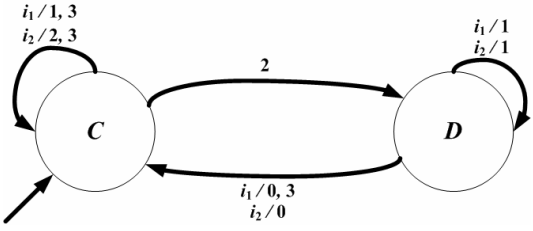

Figure 3. TFSM P

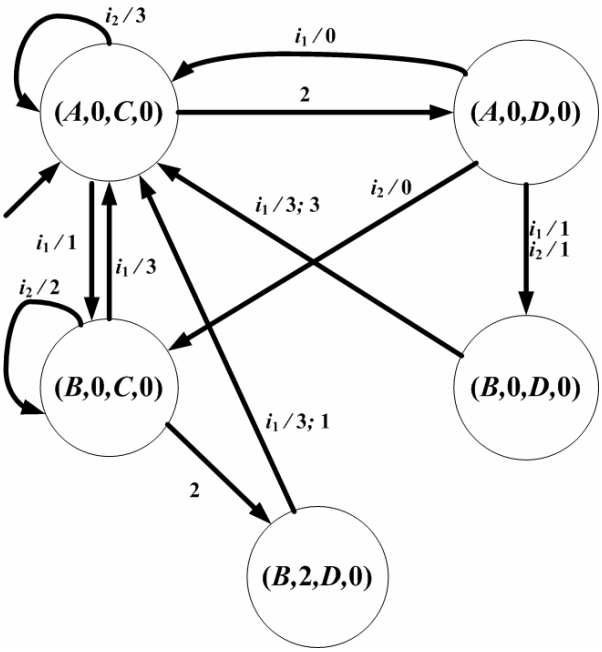

Figure 4. TFSM $S \cap P$

IV. SEPARATING SEQUENCE FOR TWO TFSMS

We suggest algorithm for constructing a separating sequence for two TFSMs.

Algorithm 2: Constructing a separating sequence of two TFSMs

Input: Complete observable TFSMs $S = \langle S, I, O, s_0, \lambda_S, \Delta_S \rangle$ and $P = \langle P, I, O, p_0, \lambda_P, \Delta_P \rangle$

Output: separating sequence for S and P (if exists)

Step 1: construct the intersection of S and P with the help of Algorithm 1. If TFSM $S \cap P$ is complete, S and P couldn't be separated. **End**.

Step 2: Derive a truncated successor tree of $S \cap P$. The root of the tree is the pair $\langle q_0, 0 \rangle$, other nodes – sets of the pairs $\langle q, t \rangle$, where q is the state of $S \cap P$.

$k := 0$;

Edge: = ∅;

$Q_{k0} := \{\langle q_0, 0 \rangle\}$;

$Q_k := \{Q_{k0}\}$.

Until

(Rule 1: for the set $Q_{kj} \in Q_k$, $j \geq 0$, there is an input *i-sep* such that each state q, $\langle q, t \rangle \in Q_{kj}$, has no *i-sep*-successors in TFSM $S \cap P$

or

Rule 2: for each set $Q_{kj} \in Q_k$ there exists $Q_{am} \in Q_a$, $a < k$, such that $Q_{kj} \supseteq Q_{am}$)

Do:

For each input i construct the set of successors M: = $\bigcup_{q \in Q_{kj}} suc_Q(q_{\downarrow Q}, i) \times \{0\}$, add M to Q_{k+1} and add triple (Q_{kj}, i, M) to the set *Edge*.

If there exists q, $\langle q, t \rangle \in Q_{kj}$, such that $(\Delta_Q(q))_{\downarrow(\mathbf{N} \cup \{\infty\})} = \infty$, define minimum time-out T and set of successors R for the set $Q_{kj} = \{\langle q_1, t_1 \rangle, \langle q_2, t_2 \rangle, \ldots, \langle q_r, t_r \rangle\}$ as follows:

$T := \min_{1 \leq u \leq r} \{T_u - t_u\}$, $T_u = (\Delta_Q(q_u))_{\downarrow(\mathbf{N} \cup \{\infty\})}$;

$R := \{\langle q_1', t_1' \rangle, \langle q_2', t_2' \rangle, \ldots, \langle q_r', t_r' \rangle\}$, $q_u' = time_Q(q_u, t_u + T)$ and either $t_u' = 0$ if ($T_u = \infty$ or $T_u = t_u + T$), or $t_u' = t_u + T$ if $T_u > t_u + T$.

Add R to Q_{k+1} and add triple (Q_{kj}, T, R) to the set *Edge*.

Step 3: If the tree was terminated according to the Rule 1, then construct the sequence of edges (Q_{00}, g_1, Q_{1j_1}), $(Q_{1j_1}, g_2, Q_{2j_2}), \ldots, (Q_{(k-1)j_{k-1}}, g_k, Q_{kj_k})$ such that $(Q_{(l-1)j_{l-1}}, g_l, Q_{lj_l}) \in Edge$ for each $l \in \{1, \ldots, k\}$ and $g_l \in \{I \cup \mathbf{N}\}$.

Collect the separating sequence $\alpha = \langle i_1, t_1 \rangle \ldots \langle i_m, t_m \rangle$:

$j := 0$;

$T_j := 0$;

$r := 0$;

While ($j \leq k$) execute:

If $g_j \in I$,

Then $i_r := g_j$, $t_r := T_j$, $r := r+1$, $T_j := 0$;

Else $T_j := T_j + g_j$;

$j := j+1$;

$m := r$;

$i_r := i\text{-}sep$, $t_r := T_j$.

If all branches of the tree were terminated according to the Rule 2, then TFSMs S and P are unseparable.

End.

The truncated tree for TFSMs S and P is presented in Figure 5:

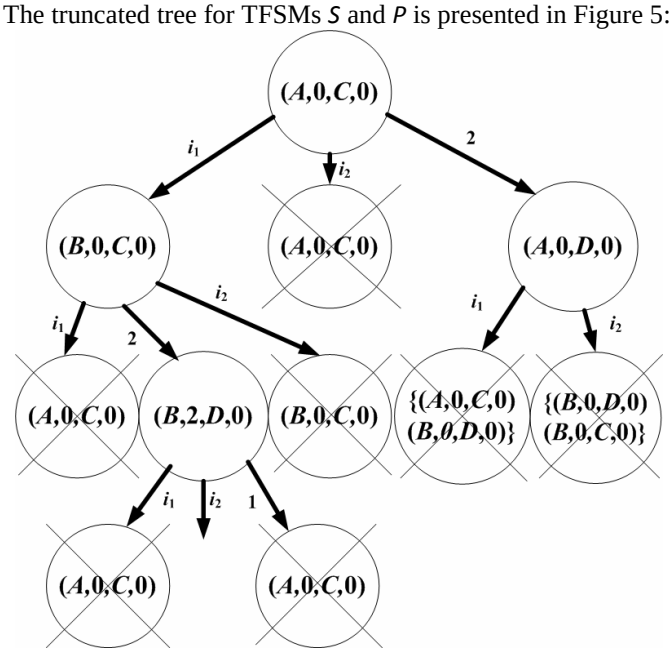

Figure 5. Truncated successor tree of $S \cap P$

So, the separating sequence is $\alpha = \langle i_1, 0 \rangle \langle i_2, 2 \rangle$.

Algorithm 2 is the modification of the algorithm from [6], of deriving a separating sequence for two untimed FSMs. The modifications are associated with time-outs, because the only way to reach some states is to wait for a while. Thus in Algorithm 2 each node of the tree is not the set of states of the intersection, but the set of pairs <state, time>. For the set in the node we determine the minimal delay and the set of successors under this delay is derived in the same way as when deriving the intersection. We need the edges labeled by delay because for the timed FSMs the separating sequence is timed input sequence, so we need to wait some time before applying another input.

Rule 2 is inherited from [6] and in this case we can't separate given timed FSMs.

Since Rule 1 is also inherited from algorithm [6], and transitions under time-outs are derived according to the rules that specify the common behavior of timed systems, the sequence $\alpha = \langle i_1, t_1\rangle \ldots \langle i_m, t_m\rangle$ derived according to Algorithm 2 will be a separating sequence for two timed separable FSMs.

It is known [6], that for given two complete separable untimed FSMs S and P, $|S| = n$ and $|P| = m$, the length of a shortest separating sequence of S and P is at most 2^{mn-1}, and this estimation is reachable. Since untimed FSM is a particular case of timed FSM the estimation will be the same.

V. Correlation between FSMs and TFSMs

To transform TFSM $S = \langle S, I, O, s_0, \lambda_S, \Delta_S\rangle$ into FSM A_S with the same behavior [3] we add to TFSM a special input $1 \notin I$ and a special output $N \notin O$. FSM $A_S = \langle S \cup S_t, I \cup \{1\}, O \cup \{N\}, s_0, \lambda_S^{\Delta}\rangle$ is constructed by adding $T - 1$ copies of each state $s \in S$ with finite time delay T, $T > 1$. Formally, A_S is constructed in the following way:

1) For each $s \in S$, $\Delta_S(s) = (s', T)$, $1 < T < \infty$, the set S_t contains each state $\langle s, t\rangle$, $t = 1, ..., T - 1$.
2) For each $s \in S$, $\langle s, t\rangle \in S_t$ and for each pair i/o, $i \in I$, $o \in O$, $(\langle s, t\rangle, i, s', o) \in \lambda_S^{\Delta}$, if and only if $(s, i, s', o) \in \lambda_S$.
3) For each $s \in S$ such that $\Delta_S(s) = (s, \infty)$ there is a transition $(s, 1, s, N)$ in A_S.
4) For each $s \in S$ such that $\Delta_S(s) = (s', T)$, $T = 1$ there is a transition $(s, 1, s', N)$ in A_S.
5) For each $s \in S$ such that $\Delta_S(s) = (s', T)$, $1 < T < \infty$, there are transitions $(s, 1, \langle s, 1\rangle, N)$, $(\langle s, j\rangle, 1, \langle s, j + 1\rangle, N)$, $j = 1, \ldots, T - 2$, and transition $(\langle s, T - 1\rangle, 1, s', N)$ in A_S.

Hence, on condition, that timed FSM S includes n states and maximal delay is T_{max}, A_S can include up to $n \cdot T_{max}$ states. Be more precise, number of states in A_S is $\sum_{s \in S} n(s)$, $n(s) = 1$, if $\Delta_S(s) = (s, \infty)$, and $n(s) = T$, if $\Delta_S(s) = (s', T)$.

Thus, in order to derive a separating sequence for two TFSMs, we transformed TFSMs into FSMs. Intersection of two FSMs could be constructed with the help of Algorithm 1 without taking into account time-outs. To construct truncated successor tree we use Algorithm 2 without time-outs [6], and then collect the sequence (if exists) with the help of step 3 (Algorithm 2). One can assure that the separating sequence for A_S and A_P (Figures 6 and 7) will be the same, i.e., $\alpha = \langle i_1, 0\rangle\langle i_2, 2\rangle$.

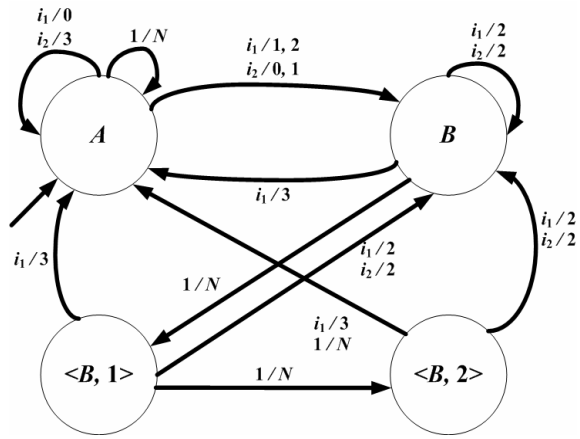

Figure 6. FSM A_S

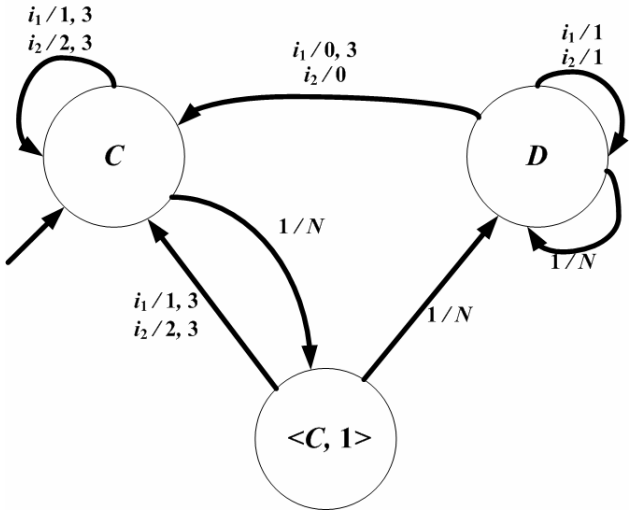

Figure 7. FSM A_P

Conclusions

In this paper we suggested two approaches to separate TFSMs. The idea of the first approach is that we construct an intersection of two TFSMs and then find separating sequence. The main advantage of this approach is comparative simplicity due to small amount of states in intersection. Disadvantage – weak theoretical basis of complete test suites derivation for TFSMs. The second approach is based on "TFSM to FSM" transformation. As a result of this transformation we have enormous increasing of states in intersection. Thus this way is hardly applicable to TFSMs with great time delays. But theoretical basis for complete test suites derivation is much more stronger for classical FSMs. In the future we're planning

to compare program implementations of these two approaches in order to find out the range of applicability of each one.

REFERENCES

[1] R. Alur, C. Courcoubetis, M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines // STOC'95, NewYork: ACM, 1995. P.363-372.

[2] M. G. Merayo. Formal Testing from Timed Finite State Machines / // Computer Networks. – 2008. – Vol. 52 №2. – P. 432-460.

[3] M. Zhigulin, S.Maag, A.Cavalli, N.Yevtushenko. FSM-based test derivation strategies for systems with time-outs // Presented to QSIC'2011.

[4] M. Gromov, D. Popov, N. Yevtushenko. Deriving test suites for timed Finite State Machines // Proceedings of IEEE East-West Design & Test Symposium 08, Kharkov: SPD FL Stepanov V.V., 2008. P.339-343.

[5] Starke, P.: Abstract automata, American Elsevier, 3–419 (1972).

[6] N. Spitsyna, K. El-Fakih, N. Yevtushenko Studying the Separability Relation between Finite State Machines // Software Testing, Verification and Reliability. –2007. – Vol. 17(4). – P. 227-241.