

Title	Modeling Urgency in Component-Based Real-Time Systems
Author(s)	Nguyen, Van Tang; Hung, Dang Van; Ogawa, Mizuhito
Citation	Lecture Notes in Computer Science, 4435: 248-255
Issue Date	2008
Type	Journal Article
Text version	author
URL	http://hdl.handle.net/10119/7882
Rights	This is the author-created version of Springer, Nguyen Van Tang, Dang Van Hung, Mizuhito Ogawa, Lecture Notes in Computer Science, 4435, 2008, 248-255. The original publication is available at www.springerlink.com , http://dx.doi.org/10.1007/978-3-540-77505-8_20
Description	

Modeling Urgency in Component-Based Real-time Systems

Nguyen Van Tang¹, Dang Van Hung², Mizuhito Ogawa¹

¹ Japan Advanced Institute of Science and Technology

E-mail: {tang-nguyen, mizuhito}@jaist.ac.jp

² United Nation University, International Institute for Software Technology

E-mail: dvh@iist.unu.edu

Abstract. A component-based realtime system is a simple model for the server-client relation with time constraints. This paper presents an efficient algorithm, called a *blackbox testing algorithm*, for detecting the emptiness of a component-based realtime system. This algorithm was originally proposed in [5], but with a certain flaw. We improve it and correct the flaw by using urgency [2] of transitions.

Keywords: Component Software, Duration Automata, Automatic Verification, Real-time Systems, Model Checking.

1 Introduction

The architectural design for embedded systems often relies on specification of the interface of components only, without accessing their internal behaviors. Based on this observation, a simple model for component-based real-time systems based on duration automata was proposed in [5]. A duration automaton does not have clock variables like a time automaton [1], but simply has an upper bound and a lower bound for each transition. A component-based real-time system is defined as a system consisting of a host, which is a general duration automaton, and several components which are duration automata with certain restrictions. A component-based real-time system can be regarded as a timed automaton, thus its emptiness is PSPACE-complete.

This paper presents an efficient algorithm for detecting the emptiness, called a *blackbox testing algorithm*. This algorithm was originally proposed in [5], but with certain flaws. We improve it and correct these flaws by using urgency of transitions, which was firstly introduced by Bornot et. al. [2] as a technique for choosing time deadline condition in complex system specifications.

2 Duration Automata

Duration automata was firstly introduced in [3] for modeling simple real-time systems. A duration automaton is a finite automaton in which each transition must occur in an associated time interval. Let \mathbb{R}^+ be the set of non-negative real numbers, and let $Intv = \{[l, u] \mid l \in \mathbb{R}^+, u \in \mathbb{R}^+ \cup \{\infty\}\}$.

Definition 1. A duration automaton is a tuple $M = \langle S, \tilde{\Sigma}, q, R, F \rangle$, where

1. S is a finite set of states,
2. $\tilde{\Sigma}$ is alphabet of actions,
3. $q \in S$ is the initial state,
4. $R \subseteq S \times \tilde{\Sigma} \times \text{Intv} \times S$ is timed transition relation, and
5. $F \subseteq S$ is the set of final states.

Each element of M is referred by $S(M)$, $\tilde{\Sigma}(M)$, $R(M)$, $q(M)$, and $F(M)$, respectively. An untimed automaton $\text{untimed}(M)$ is obtained by forgetting time constraints, i.e., replacing R with $\text{untimed}(R) = \{(s, a, s') \mid (s, a, [l, u], s') \in R\}$. As in standard terminology,

- A *configuration* of M is a pair $(s, d) \in S \times \mathbb{R}^+$.
- The *initial configuration* of M is $(q, 0)$.
- An *acceptance configuration* of M is a configuration (s, d) where $s \in F$.

A duration automaton is equivalent to a timed automata with a single clock such that each transition resets it. A configuration (s, d) is regarded as a state s with a clock d .

- A transition of M on configurations is either a *time transition* $(s, d) \xrightarrow{\delta} (s, d + \delta)$ or a *discrete transition* $(s, d) \xrightarrow{\delta, a} (s', 0)$ where $a \in \tilde{\Sigma}$, $\delta \geq 0$, $l \leq d + \delta \leq u$, and $(s, a, [l, u], s') \in R$.
- A (possibly empty) sequence $w = (a_1, t_1) \dots (a_k, t_k) \in (\tilde{\Sigma} \times \mathbb{R}^+)^*$ is a *timed word* of M if and only if there is a run $(s_0, 0) \xrightarrow{\delta_1, a_1} (s_1, 0) \xrightarrow{\delta_2, a_2} \dots \xrightarrow{\delta_k, a_k} (s_k, 0)$ such that $s_0 = q$, $s_k \in F$, $t_1 = \delta_1$ and $t_{i+1} - t_i = \delta_{i+1}$ for $1 \leq i \leq k-1$.

Theorem 1. Duration automata is closed under union, intersection and complementation. Decision problems for duration automata are decidable.

Proof. (Sketch) For a given duration automaton M , one can reduce M to a finite automaton M' . We first list the endpoints of intervals (lower and upper bounds of intervals) of transitions in M as an increasing sequence, say, $0 = p_0 < p_1 < p_2 \dots < p_n < \infty$. This is possible because the number of transitions of M is finite. Secondly, we define the set of basic intervals $BI = \{[p_0, p_1], \dots, [p_{n-1}, p_n], [p_n, \infty)\}$. Since each interval appeared in a transition of M is the union of certain basic intervals. So, each transition of M can be divided into several ones. For instance, $(s, a, [p_0, p_2], s')$ can be divided into $(s, a, [p_0, p_1], s')$ and $(s, a, [p_1, p_2], s')$. We now construct a finite automaton M' such that $S(M') = S(M)$, $F(M') = F(M)$, the input alphabet of M' is $\tilde{\Sigma}(M') = \tilde{\Sigma}(M) \times BI$. Let $(s, (a, [p_i, p_{i+1}]), s') \in R(M')$ if $(s, a, [p_i, p_{i+1}], s') \in R(M)$. Clearly, M' accepts a word $(a_1, [l_1, u_1]) \dots (a_n, [l_n, u_n])$ if and only if M accepts the timed word $(a_1, t_1) \dots (a_n, t_n)$, where $t_0 = 0$ and $(l_i \leq t_i - t_{i-1} \leq u_i)$ for $1 \leq i \leq n$. Thus, the emptiness and the closure properties of duration automata are reduced to that of finite automata, respectively. \square

3 Synchronized Composition Systems

Duration interface automata is duration automata in which the input alphabet $\tilde{\Sigma}$ is decomposed into pairwise disjoint alphabets Σ, Δ and ∇ , which correspond to internal, input and output actions, respectively.

Definition 2. A host is a duration interface automaton. A component is a duration interface automaton $X = \langle S, \Sigma \cup \Delta \cup \nabla, q, R, F \rangle$ that satisfies:

- $\Sigma = \emptyset$ (i.e., no “explicit” internal actions).
- $(s, a, [l, u], s') \in R \wedge a \in \Delta$ implies $l = 0 \wedge u = \infty$ (i.e., an input can occur anytime).
- $(s, a, [l, u], s') \in R \wedge a \in \nabla$ implies $u = \infty$ (i.e., when an output is ready, it can be sent at any time afterward).

Definition 3. A synchronized composition system $Sys = \langle M, X_1, \dots, X_k \rangle$ consists of a single host M and components X_1, \dots, X_k such that $\tilde{\Sigma}(X_i) \cap \tilde{\Sigma}(X_j) = \emptyset$ for each $i \neq j$, $\Sigma(M) \cap \tilde{\Sigma}(X_i) = \emptyset$ for each i , $\Delta(M) = \bigcup_{i=1}^k \nabla(X_k)$, $\nabla(M) = \bigcup_{i=1}^k \Delta(X_k)$, and

- The set of configurations is $\{(s_0, d_0), (s_1, d_1), \dots, (s_k, d_k) \mid s_0 \in S(M), s_1 \in S(X_1), \dots, s_k \in S(X_k), d_i \in \mathbb{R}^+\}$.
- A transition is $((s_0, d_0), (s_1, d_1), \dots, (s_k, d_k)) \xrightarrow{\delta, a} ((s'_0, d'_0), (s'_1, d'_1), \dots, (s'_k, d'_k))$ for $\delta \geq 0$ and $a \in \bigcup_{i=1}^k \tilde{\Sigma}(X_i)$, if there exists i with $1 \leq i \leq k$ such that
 - $a \in \tilde{\Sigma}(X_i)$,
 - $l_0 \leq d_0 + \delta \leq u_0$ and $l_i \leq d_i + \delta \leq u_i$ (called synchronization condition) for $(s_i, a, [l_i, u_i], s'_i) \in R(X_i)$ and $(s_0, a, [l_0, u_0], s'_0) \in R(M)$,
 - $d'_0 = d'_i = 0$, and
 - $(s'_j, d'_j) = (s_j, d_j + \delta)$ for $j \neq 0, i$.
- A run is a sequence of transitions that starts from the initial configuration $((q(M), 0), (q(X_1), 0), \dots, (q(x_k), 0))$.
- A timed word $(a_1, t_1) \dots (a_k, t_k)$ with $t_1 = \delta_1$ and $t_{i+1} = t_i + \delta_{i+1}$ is accepted if there is a run $((q(M), 0), (q(X_1), 0), \dots, (q(x_k), 0)) \xrightarrow{\delta_1, a_1} \dots \xrightarrow{\delta_k, a_k} ((s_0, d_0), (s_1, d_1), \dots, (s_k, d_k))$ with $s_0 \in F(M), s_1 \in F(X_1), \dots, s_k \in F(X_k)$.

Theorem 2. A synchronized composition system $Sys = \langle M, X_1, \dots, X_k \rangle$ is a timed automaton with $k+1$ clocks such that each transition with a time constraint $l_i \leq d_i \leq u_i$ on a clock d_i will reset d_i to 0.

Proof. (Sketch) Let \mathcal{C} be the set of time constraints $[l_j, u_j]$ appearing in a host M and components X_i . Note that $l_j, u_j \in \mathbb{R}^+$. Assume that we can choose \mathcal{C}' (a digitization of \mathcal{C}) consisting of rational time constraints $[l'_j, u'_j]$ such that there is a run of Sys if and only if there is a run of Sys' , where Sys' is obtained replacing each $[l_j, u_j]$ with its digitization $[l'_j, u'_j]$. Then, the proof has done.

Let $rat(\mathcal{C})$ be the set of rational numbers appearing in \mathcal{C} and let m be a common multiplier of denominators of positive elements in $rat(\mathcal{C})$. Let $irr(\mathcal{C})$ be the set of irrational numbers appearing in \mathcal{C} and let $lin(\mathcal{C})$ be the set of all possible

linear combinations of $irr(\mathcal{C})$ with natural numbers (i.e., $lin(\mathcal{C}) = \{n_1\alpha_1 + \dots + n_l\alpha_l \mid n_j \in \mathbb{N}, \alpha_j \in irr(\mathcal{C})\}$). Assume that (α, β) is the pair such that $\alpha \in irr(\mathcal{C})$, $\beta \in lin(\mathcal{C})$, and $\epsilon_{\alpha, \beta} = \frac{\alpha}{\beta} - \lfloor \frac{\alpha}{\beta} \rfloor > 0$. Since a pair (α, β) with $\alpha \in irr(\mathcal{C})$, $\beta \in lin(\mathcal{C})$, and $\beta < \alpha$ is finitely many, (α, β) with $\epsilon_{\alpha, \beta}$ to be the least exists. We choose a sufficient large multiplier \bar{m} of m such that $\frac{1}{\bar{m}} < \min(\frac{\epsilon_{\alpha, \beta}}{2}, \frac{1 - \epsilon_{\alpha, \beta}}{2})$, and set $l'_j = \frac{\lfloor \bar{m}l_j \rfloor}{\bar{m}}$ and $u'_j = \frac{\lfloor \bar{m}u_j \rfloor}{\bar{m}}$ for each $l_j, u_j \in \mathbb{C}$. \square

Example 1. Fig. 3 shows a simple synchronized composition system $Sys = \langle X_1, X_2 \rangle$ and its corresponding timed automaton \mathcal{A} .

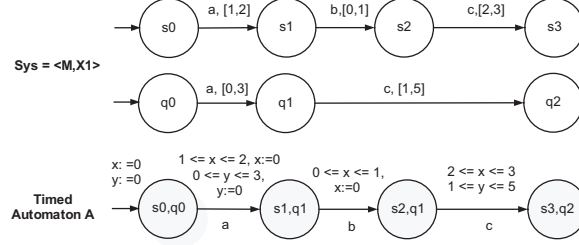


Fig. 1. Synchronized Composition System as a Timed Automaton

From Theorem 2, the emptiness problem of a component-based realtime system is decidable. However, its complexity is expensive, i.e., PSPACE-complete [1] after digitization of time constraints.

4 Component-based realtime systems

Definition 4. A component X is input/output deterministic if

- for $a \in \Delta(X)$, $(s, a, [0, \infty), s')$, $(s, a, [0, \infty), s'') \in R(X)$ implies $s'' = s'$ (input determinism), and
- for $b \in \nabla(X)$ and $b' \in \nabla(X) \cup \Delta(X)$, $(s, b, [l, \infty), s')$, $(s, b', [l', u'], s'') \in R(X)$ implies $s'' = s'$, $l' = l$, $u' = \infty$, and $b' = b$ (output determinism).

A synchronized composition system $Sys = \langle M, X_1, \dots, X_k \rangle$ is a component-based realtime system [5] if each component X_i is input/output deterministic.

Definition 5. We borrow notations from Definition 3. In a component-based system $Sys = \langle M, X_1, \dots, X_k \rangle$, a transition $((s_0, d_0), (s_1, d_1), \dots, (s_k, d_k)) \xrightarrow{\delta} \xrightarrow{a} ((s'_0, d'_0), (s'_1, d'_1), \dots, (s'_k, d'_k))$ is urgent if δ is the minimum among synchronization conditions of all possible transitions from $((s_0, d_0), (s_1, d_1), \dots, (s_k, d_k))$, and delayable otherwise. We also say a corresponding transition $(s_0, a, [l_0, u_0], s'_0) \in R(M)$ of a host is urgent, and delayable otherwise.

Definition 6. Let $w = (a_1, t_1) \cdots (a_k, t_k)$ and let $a_i \in A$. For $B \subseteq A$, the projection $w|_B$ is the subsequence of w obtained by filtering each element (a_j, t_j) with $a_j \in B$. For $a_j \in B$, (a_h, t_h) is a local predecessor of (a_j, t_j) wrt B , if $a_h \in B$, $h < j$, and $a_i \notin B$ for each i with $h < i < j$.

Definition 7. Let $Sys = \langle M, X_1, \dots, X_k \rangle$ be a component-based real-time system. For a timed word $w = (a_1, t_1) \dots (a_n, t_n)$, let $a_j \in \nabla(X_i)$ and let (a_h, t_h) be the local predecessor of (a_j, t_j) wrt $\tilde{\Sigma}(X_i)$. For $(s', a_j, [d_j, \infty), s'') \in R(X_i)$ with $q(X_i) \xrightarrow{\text{untime}(w|_{\tilde{\Sigma}(X_i)})} s'$ in $\text{untimed}(X_i)$, d_j is the minimum delay at (a_j, t_j) .

Definition 8. A consecutive sequence of transitions $(s_{i-1}, a_i, [l_i, u_i], s_i) \in R(M)$ ($i = 1, \dots, n$) is called an accepted sequence of transitions of the host M if $s_0 = q(M)$ and $s_n \in F(M)$.

Note that such a minimum delay is well-defined, since each component in Sys is input/output deterministic. Let r be the number of states of M , and let m is the maximal number of states of components $X_j, j \leq k$. Let P be the length of the longest path (number of transitions) from the initial state to a final state of M in which any cycle is not repeated more than $r * m^k$ times. The next theorem reduces the emptiness of a whole component-base realtime system to that of its host under certain conditions.

Theorem 3. Let $Sys = \langle M, X_1, \dots, X_k \rangle$ be a component-based realtime system. There is an accepted timed word of Sys if and only if there are an accepted sequence of transitions of the host M $\sigma = (s_0, a_1, [l_1, u_1], s_1)(s_1, a_2, [l_2, u_2], s_3) \dots (s_{n-1}, a_n, [l_n, u_n], s_n)$ with the length $n \leq P$, and a real number sequence $0 = t_0 \leq t_1 \leq \dots \leq t_n$ satisfying following conditions:

- $w_i = a_1 a_2 \dots a_n |_{\tilde{\Sigma}(X_i)}$ is accepted by $\text{untimed}(X_i)$ for each i with $1 \leq i \leq k$,
- $l_i \leq t_i - t_{i-1} \leq u_i$ for all i with $1 \leq i \leq n$,
- When $a_j \in \nabla(X_i)$, let (a_h, t_h) be the local predecessor of (a_j, t_j) wrt $\tilde{\Sigma}(X_i)$ and let d_j be the minimum delay at (a_j, t_j) . Then,
 - $t_j - t_h \geq d_j$, and
 - if a transition $(s_j, 0) \xrightarrow{\delta} \xrightarrow{a_j} (s_{j+1}, 0)$ is urgent, $t_j = \min \{t \mid t - t_h \geq d_j \wedge l_j \leq t - t_{j-1} \leq u_j\}$.

Proof. (Sketch) We only have to prove the bound P in “only if” part. Assume that a timed word $w = (a_1, t_1) \cdots (a_n, t_n)$ is accepted by Sys . Timed word w is inductively computed by constructing an accepted sequence of transitions $\phi = (s_0, a_1, [l_1, u_1], s_1)(s_1, a_2, [l_2, u_2], s_3) \cdots (s_{n-1}, a_n, [l_n, u_n], s_n)$ of the host M . If $n \leq P$, the proof is done. If $n > P$, then ϕ must include at least a cycle c with more than $r * m^k$ repetitions. By the pumping lemma like argument, we can find a shorter accepted sequence of transitions of M that satisfies all the conditions in the Theorem. \square

In the next section, the blackbox testing algorithm will be presented by searching an accepted sequence of the host M satisfying the conditions in Theorem 3 up to the length P .

5 Checking Emptiness of Component-based Realtime Systems

The emptiness problem for a system plays a key role in checking the safety. An algorithm for checking the emptiness of a component-based system using black box testing was originally proposed in [5]. However, there is a flaw such that a component-based realtime system is empty, whereas the algorithm in [5] reports that the system is not empty. For instance, consider the following simple example.

Example 2. Let $Sys = \langle M, X \rangle$ where M is a host and X is a component.

- $M = \langle \{s_0, s_1, s'_1\}, \{a\}, s_0, \{(s_0, a, [2, 4], s_1), (s_0, a, [5, 10], s'_1)\}, \{s'_1\} \rangle$.
- $X = \langle \{q_0, q_1\}, \{a\}, q_0, \{(q_0, a, [3, \infty), q_1)\}, \{q_1\} \rangle$.

In [5], the state (s'_1, q_1) is regarded as a successor of (s_0, q_0) . But, (s'_1, q_1) is not reachable from (s_0, q_0) . This is due to the fact that Sys has already changed from (s_0, q_0) to (s_1, q_1) at some point in the time interval $[3, 4]$.

To deal with this problem, we introduce urgency for transitions to specify time deadline condition of configurations. For the emptiness problem, we first use the **BlackboxTest** algorithm proposed in [5] for solving membership for a component. Secondly, we construct Algorithm 1 to compute time deadline condition of a given configuration. Lastly, with the aid of Algorithm 1 and Theorem 3, we construct Algorithm 2 to check the emptiness of a component-based system using black box testing.

For a sequence of transitions ϕ , let $label(\phi)$ denote the sequence of the labels corresponding to ϕ . For a given prefix of a generated sequence of transitions $\sigma = e_1 e_2 \dots e_n$, where $e_i = (s_{i-1}, a_i, [l_i, u_i], s_i) \in R(M)$ ($i = 1..n$). Suppose that t_0, t_1, \dots, t_n are inductively computed in advance. Time deadline of s_n along σ is denoted by $deadline_\sigma(s_n)$. It can be computed by the following algorithm:

ALGORITHM 1. **Deadline** $_\sigma(s_n)$: (Check the conditions (2) and (3) of Theorem 3)

Input: A prefix-generated sequence $\sigma = e_1 e_2 \dots e_n$

Output: $deadline_\sigma(s_n)$.

Method:

1. Compute the set $R(s_n) := \{e \mid e = (s_n, a, [l, u], s) \in R(M)\}$.
2. $deadline := \infty$. For $j \leq k$ let m_j be the largest index of σ such that $a_{m_j} \in \tilde{\Sigma}(X_j)$ if it exists, otherwise, set $m_j = 0$. For each $e = (s_n, a, [l, u], s) \in R(s_n)$.
 - (a) If $a \in \Delta(X_j) \cup \Sigma(M)$, if **BlackboxTest** $(X_j, label(\sigma)|_{\tilde{\Sigma}(X_j)}) = \text{“yes”}$ and $u < deadline$ then $deadline := u$.
 - (b) If $a \in \nabla(X_j)$. If **BlackboxTest** $(X_j, label(\sigma)|_{\tilde{\Sigma}(X_j)}) = \text{“yes”}$, let d be the value of d_{X_j} .

- **Case 1: e is delayable.** If $t_n - t_{m_j} + u \geq d$ and $u < deadline$ then $deadline := u$.
 - **Case 2: e is urgent.**
 - If $t_n - t_{m_j} + l \leq d \leq t_n - t_{m_j} + u$ and $d - (t_n - t_{m_j}) < deadline$ then $deadline := d - (t_n - t_{m_j})$.
 - If $t_n - t_{m_j} + l \geq d$ and $l < deadline$ then $deadline := l$.
3. **return** Delainey;

With the aid of the Algorithm 1, the emptiness of a component-based real-time system can be solved by the following testing procedure.

ALGORITHM 2. Non-Emptiness(Sys): (Check all conditions of Theorem 3)

Input: Component-based real-time system $Sys = \langle M, X_1, \dots, X_k \rangle$

Output: “Yes” if the set of timed words of Sys is not empty, “No” otherwise.

Method:

1. Compute P . Generate all accepted sequences of transitions of M with length less than P .
2. Check on-the-fly whether any prefix of a generated sequence satisfies the conditions of Theorem 3. This can be done by:

For each prefix of a generated sequence of transitions $\sigma = e_1 \dots e_{n-1}$, where $e_i = (s_{i-1}, a_i, [l_i, u_i], s_i)$ for each i with $1 \leq i \leq n-1$. Suppose that t_0, t_1, \dots, t_n are inductively computed in advance. For $j \leq k$ let m_j be the largest index of σ such that $a_{m_j} \in \tilde{\Sigma}(X_j)$ if it exists, otherwise, let $m_j = 0$. For each transition $e_n = (s_{n-1}, a_n, [l, u], s)$ of the host M starting from s_{n-1} . Compute $deadline_\sigma(s_{n-1})$ using Algorithm 1. If $l \leq deadline_\sigma(s_{n-1})$ then:

- (a) If $a_n \in \Delta(X_j)$, then if: **BlackboxTest** $(X_j, label(\sigma)|_{\tilde{\Sigma}(X_j)}) = \text{“no”}$, σe_n does not satisfy the conditions of Theorem 3. Otherwise, $\sigma := \sigma e_n$, $m_j := n$. If e_n is delayable then $t_n := t_{n-1} + u$. If e_n is urgent then $t_n := t_{n-1} + l$.
- (b) If $a_n \in \nabla(X_j)$.

If **BlackboxTest** $(X_j, label(\sigma)|_{\tilde{\Sigma}(X_j)}) = \text{“yes”}$, let d be the value of d_{X_j} .

Case 1: If e_n is delayable

- i. If $t_{n-1} - t_{m_j} + u < d$: then σe_n does not satisfy the conditions of Theorem 3.
- ii. If $t_{n-1} - t_{m_j} + u \geq d$: then $\sigma := \sigma e_n$, $m_j := n$, $t_n := t_{n-1} + u$.

Case 2: If e_n is urgent

- i. If $t_{n-1} - t_{m_j} + u < d$ then σe_n does not satisfy the conditions of Theorem 3.
- ii. If $(t_{n-1} - t_{m_j} + l) < d \leq (t_{n-1} - t_{m_j} + u)$ then the conditions of Theorem 3 are satisfied; update $\sigma := \sigma e_n$, $t_n := t_{m_j} + d$, $m_j := n$.
- iii. If $t_{n-1} - t_{m_j} + l \geq d$ then update $\sigma := \sigma e_n$, $t_n := t_{n-1} + l$, $m_j := n$.

If **BlackboxTest** $(X_j, label(\sigma)|_{\tilde{\Sigma}(X_j)}) = \text{“no”}$, the conditions of Theorem 3 are not satisfied.

- (c) If $a_n \in \Sigma(M)$ then $\sigma := \sigma e_n$, $t_n := t_{n-1} + u$.

3. If a generated sequence satisfying the conditions of Theorem 3 is found, return “Yes”. Otherwise, return “No”.

The complexity for the worst cases of this algorithm is $O(P^2 * K^{P+1})$, where $K = |\tilde{\Sigma}(M)|$ is the size of the alphabet of the system Sys . Unlike the complexity of checking the emptiness for timed automata, this complexity does not depend on the size of the constants occurring in the time intervals for the transitions.

6 Conclusion

This paper presented an efficient algorithm for detecting the emptiness, called a *blackbox testing algorithm*. This algorithm was originally proposed in [5], but with a certain flaw. We improved and corrected it by using urgency of transitions, which was firstly introduced by Bornot et. al. [2] as a technique for choosing time deadline condition in complex system specifications. The urgency enables us to compute the deadline of an accepted behavior of a system using Algorithm 2.

Currently, the algorithm covers checking emptiness only. With the urgency, we can describe a property in Timed Computation Tree Logic (TCTL), such as $\phi \implies F_{\leq t}\psi$. The next step is to give an efficient checking algorithm for such TLCL properties of a component-based realtime system.

Acknowledgments

This research is supported by the 21st Century COE “Verifiable and Evolvable e-Society” funded by Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

1. R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126: 183-235, 1994.
2. S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In COMPOS 1997, 103-129, Springer LNCS 1536, 1997.
3. Zhou Chaochen. Linear Duration Invariants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pp. 86-109, Springer LNCS 963, 1994.
4. Zhe Dang and Gaoyan Xie. CTL model-checking for systems with unspecified finite state components. In SAVCBS'04, ACM SIGSOFT 2004/FSE-12, pp. 32-38, 2004.
5. Dang Van Hung and Bui Vu Anh. Model Checking Real-time Component Based Systems with Blackbox Testing. In the IEEE proceeding of RTCSA05, page 76-79, Hong Kong, August 2005.