VNU Journal of Science: Comp. Science & Com. Eng., Vol. 33, No. 1 (2017) 8-15

A Model for Real-time Concurrent Interaction Protocols in Component Interfaces

Van Hung Dang*, Trinh Dong Nguyen, Hoang Truong Anh

VNU University of Engineering and Technology, Hanoi, Vietnam

Abstract

Interaction Protocol specification is an important part for component interface specification. To use a component, the environment must conform to the interaction protocol specified in the interface of the component. We give a powerful technique to specify protocols which can capture the constraints on temporal order, concurrency, and timing. We also show that the problem of checking if a timed automaton conforms to a given real-time protocol is decidable and develop a decision procedure for solving the problem.

Received 16 January 2017; Accepted 27 February 2017

Keywords: Interaction Protocol, Timed Automata, Region Graph, Component Interface.

1. Introduction

Component-based system architectures have been an efficient divide-and-conquer design technique for the development of complex real-time embedded systems. A key role in this technique is component interface modeling and specification. There have been many significant progresses towards a comprehensive theory for interfaces, see for example [2, , 3, 5, 6, 7]. In those works different aspects of interfaces have been modeled and specified such as interaction protocols, contracts, concurrency, relations, synchnony and asynchrony. An approach that integrates all those aspects has been introduced in [4]. However, there has not been an intuitive and powerful model for real-time interaction protocols. This kind of model plays an crucial role in systems where a service from a component may take long time to finish.

An interaction protocol specified in the interface of a component is a precondition on the temporal order on the use of services from the component. Fail to satisfy this precondition may lead to a system deadlock [2]. In real-time systems, when a service from a component takes a considerable time to carry out, too frequently calling to this service may lead to the error state too. So, we need to specify the minimum duration between two consecutive calls to the services that takes time, and this also plays a role of precondition on the consecutive calls to those services in the interaction protocols. Another possibility that we need to consider when specifying this kind of time constraints is that a component may be able to provide services in parallel. In this case, time constraints do not apply to concurrent services.

Let us consider an example. Imagine that we have a software component that provide accesses to two files: one stores the information about products and the other stores the information about customers. To access to a

*Corresponding author. E-mail.: dvh@vnu.edu.vn
https://doi.org/10.25073/2588-1086/vnucsce.154

file, one needs to open it, and after use one needs to close it. Accesses to different files can be done in parallels, and access can be reads and writes such that all the reads should be before writes. Let us denote by O_p, R_p, W_p and C_p the accesses open, read, write and close for the file 1 (for products), and by O_c, R_c, W_c and C_c the accesses open, read, write and close for the file 2 (for customers). To use the component we need to activate it by action A, and we need to deactivate it by action F after use. The interaction protocol could be specified by two regular expressions to express the condition on the temporal order between actions on each file. These regular expressions could be $(A(O_pR_pW_pC_p)^*F)^*$ and $(A(O_cR_cW_cC_c)^*F)^*$. Does the execution $AO_pO_cR_pR_cW_cW_pC_pC_cF$ conform to this protocol? It does because it satisfies the restriction on the temporal order for each file. Now, assume that it takes 1 second for the read accesses, then the execution will satisfy the protocol if the delays between R_p and W_p (not R_p and R_c; these can be done in parallel), and R_c and W_c are more than 1 second.

In this work, we propose a technique to specify real-time concurrent interaction protocols for component interfaces that is an efficient formalization of the specification from the example mentioned above, and define formally what we mean by saying a real-time execution conforms to an interaction protocol in our model. Then we develop a technique to check if a real-time system modeled by a timed automaton satisfies a real-time concurrent interaction protocol specified in the interface of a component.

The paper is organized as follows. The next section presents our general model for real-time concurrent interaction protocols. Section 3 presents an algorithm to check if a timed automaton satisfies a protocol specification. The last section is the conclusion of our paper.

2. General protocol model

Let Σ_i, $i = 1, \dots, k$ be alphabets of service names for a component $\mathcal{C}$, and let $\Omega = \bigcup_{i=1}^{k} \Sigma_i$ be the alphabet of all service names that the component provides. Our intention is that services in each Σ_i need to be executed sequentially, and services in different Σ_i and Σ_j can be executed in parallel. Each Σ_i, $i = 1, \dots, k$ can overlap another, but they must not be included in each other, i.e. Σ_i is a maximal set of services that need to be executed in sequence. When $k = 1$ there is no concurrency for the component. Each service in Ω may take time to finish. We specify this fact by a function $\delta: \Omega \to \mathbb{R}^{\geq}$. So, a service $a \in \Omega$ takes $\delta(a)$ time units to finish. An interaction protocol specifies a constraint on the temporal order on the services in each separate Σ_i, and this is modeled efficiently by a regular expression on Σ_i. Therefore, we define:

Definition 1 (Real-time interaction protocol) A real-time interaction protocol π is a tuple $\langle(\Sigma_1, R_1), \dots, (\Sigma_k, R_k), \delta\rangle$, where $\delta: \bigcup_{i=1}^{k} \Sigma_i \to \mathbb{R}^{\geq}$, and R_i is a regular expression on Σ_i for $i = 1, \dots, k$.

Example. In the example introduced in the Introduction of this paper,

$(\Sigma_1, R_1) = (\{A, O_p, R_p, W_p, C_p, F\}, (A(O_pR_pW_pC_p)^*F)^*)$ *and*,

$(\Sigma_2, R_2) = (\{A, O_c, R_c, W_c, C_c, F\}, (A(O_cR_cW_cC_c)^*F)^*)$.

$\delta(R_p) = \delta(R_c) = 1$, and $\delta(X) = 0$ for all other services X.

Let, in the sequel, for the simplicity of the presentation, for a regular expression R we overload R to denote also the language generated by R, and when R is the language generated by R can be understood from the context. Note that a regular expression can always be represented by an automaton.

This definition gives a simple syntax representation for real-time protocols. To understand the meaning of this representation

we need to define what to mean by saying a real-time execution conforms to a protocol in our model. We will use a timed automaton as our system model, and therefore, use a timed language to represent the behavior of our system.

A timed word over an alphabet Ω is a sequence $w = (a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$, where $t_{i-1} \leq t_i$ for $0 < i \leq n$, $t_0 = 0$. The intuition of this representation for a behavior is that the action a_i takes place at time t_i. Given a protocol π as in Definition 1, how to mean that w conforms to π? Let us denote $untimed(w) = a_1 a_2 \dots a_n$. For a word $x \in \Omega^*$ we denote $x|_{\Sigma_i}$ the projection of x on Σ_i, i.e. the word obtained from x by removing all the characters that do not belong to Σ_i.

Definition 2 (Conformation) A timed word $w = (a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ conforms protocol π, denoted by $w \vDash \pi$, iff for all $i \leq k$

1. $untimed(w)|_{\Sigma_i} \in R_i$, and
2. let $untimed(w)|_{\Sigma_i} = a_{j_1} \dots a_{j_{m_i}}$, then $t_{j_{l+1}} - t_{j_l} \geq \delta(a_{j_l})$ for all $l < m_i$.

The first condition in the definition says that the temporal order between sequential services is allowed by the component and reach an acceptance state of the component, and the second condition says that the component has been given enough time for providing the services. According to this definition, the behavior

$(A, 0)(O_p, 0)(O_c, 0)(R_p, .5)(R_c, 1)(W_c, 2)$
$(W_p, 2)(C_p, 2)(C_c, 2)(F, 3)$

conforms to the protocol in Example 2. However,

$(A, 0)(O_p, 0)(O_c, 0)(R_p, .5)(R_c, 1)(W_c, 1.5)$
$(W_p, 2)(C_p, 2)(C_c, 2)(F, 3)$

does not as $1.5 - 1 < \delta(R_c)$.

From the semantics of a protocol π, when no services can be executed in parallel $k = 0$, and when there is no constraint for temporal order on Σ_i and acceptance state the regular expression $R_i = \Sigma_i^*$.

Given a component $\mathcal{C}$ with the protocol specification π in its interface, a design of a system, in order to use the services from $\mathcal{C}$, all the accepted behaviors of the system design need to conform to π. The best model of real-time systems is timed automata model [1] to the best of our knowledge. Now the question of the pluggability of a real-time environment to component $\mathcal{C}$ is to decide whether all the members of the timed language of a given timed automaton $\mathcal{A}$ conform to the protocol π. If it is the case, we write $\mathcal{A} \vDash \pi$ for short.

3. Checking the pluggability

In this section we present a technique to solve the problem mentioned in the last section. Namely, we will prove that it is decidable if all the accepted behaviors of a timed automaton $\mathcal{A}$ conform to a real-time concurrent interaction protocol π. Then we develop an algorithm to check if $\mathcal{A} \vDash \pi$. The algorithm serves for answering the question if the component $\mathcal{C}$ can fit to our design. For simplicity, we now restrict ourselves to the case that the value of function δ in π is integers.

Since the concept of timed automata may not be familiar to some readers, we recall this concept from [1]. A timed automaton is a finite state machine with an additional set of clock variables X and an additional set of clock constraints. A clock constraint ϕ over X is defined by the following grammar:

$\phi \mathrel{\widehat{=}} x \leq n \mid x \geq n \mid \neg\phi \mid \phi_1 \wedge \phi_2,$

where $x \in X$ and n stands for a natural number. Let $\Phi(X)$ denote the set of all clock constraints over X.

Definition 3 (Timed automata) A timed automaton M is a tuple $\langle L, s_I, \Sigma, X, E, \mathcal{F} \rangle$, where

- L is a finite set of locations,
- $s_I \in L$ is an initial location,
- Σ is a finite set of labels,
- X is a finite set of clocks,
- $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$ is a finite set of transitions. An $e = \langle s, a, \phi, \lambda, s' \rangle \in E$ represents a transition from location s to

locationss', labeled with a; s and s' are called source and target locations of e, and denoted by $\overleftarrow{e}$and$\overrightarrow{e}$respectively; ϕ is a clock constraintover X that must be satisfied when the transition e is enabled, and $\lambda \subseteq X$ is the set of clocks to be reset by e when it takes place. In the sequel, we will use the subscript e with ϕ and λ to indicate that ϕ and λ are associated to e.

• $\mathcal{F} \subseteq L$is the set of acceptance locations.

In this paper, for simplicity, we only consider the deterministic timed automata, i.e. those timed automata which do not have more than one a-labeled edge starting from a location s for any label $a \in \Sigma$.

A clock interpretation ν for a set of clock X is a mapping $\nu: X \to Reals$, i.e. ν assigns to each clock $x \in X$ the value $\nu(x)$. A clock interpretation represents the values of all clocks in X at a time point. We adopt the following denotations. $\nu_0$always denotes the clock interpretation which maps from X to $\{0\}$. For a clock interpretation ν and for $t \in R$, $\nu + t$ denotes the clock interpretation which maps each clock $x \in X$ to the value $\nu(x) + t$. For $\lambda \subseteq X$, $[\lambda \mapsto 0]\nu$ is the clock interpretation which assigns 0 to each $x \in \lambda$ and agrees with ν over the rest of the clocks.

A state of a timed automaton M is a pair $\langle s, \nu \rangle$, where $s \in L$ and ν is a clock interpretation for X. The fact that M is in a state $\langle s, \nu \rangle$ at a time instant means that M stays in location s with all clock values agreeing with ν at that instant.

The behavior of timed automata can be represented by timed words (or timed-stamped transition sequences). A behavior σ is a timed word

$\sigma = (e_1, \tau_1)(e_2, \tau_2) \dots (e_m, \tau_m)$, where $m \geq 1$ and $e_i \in E$, $\overrightarrow{e_{i-1}} = \overleftarrow{e_i}$ for $1 \leq i \leq m$ (with the convention $\overrightarrow{e_0} = s_I$), and where $0 = \tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_m$, such that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies ϕ_{e_i} for all $1 \leq i \leq m$, where $\nu_i = [\lambda_{e_i} \mapsto 0](\nu_{i-1} + \tau_i - \tau_{i-1})$ for $1 \leq i \leq m$.

So, a behavior σ expresses that M starts from the initial location s_I, transits to $\overrightarrow{e_1}$by taking e_1 at time τ_1, then transits to $\overrightarrow{e_2}$ by taking e_1 at time τ_2, and so on, and at last transits to $\overrightarrow{e_m}$ at time τ_m. Note that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ is the value of the clock variables just before e_i's taking place, and ν_i is the value of the clock variables just after e_i's taking place. The behavior σ expresses also that the system M stays in the location$\overleftarrow{e_i}$for $\tau_i - \tau_{i-1}$ time units, and then transits to by $\overleftarrow{e_{i+1}}$for $(1 \leq i \leq m)$. If $\sigma = (e_1, \tau_1)(e_2, \tau_2) \dots (e_m, \tau_m)$ is a behavior of timed automaton M, we call $\overrightarrow{e_m}$ a reachable location of M and $\langle \overrightarrow{e_m}, \nu_m \rangle$ a (discrete) reachable state of M. A behavior of timed automaton M is accepted iff $\overrightarrow{e_m} \in \mathcal{F}$. Let $s_i = \overrightarrow{e_i}$, for $1 \leq i \leq m$, and $s_0 = s_I$. Then the run corresponding to σ is the sequence:

$\langle s_0, \nu_0 \rangle \to_{\tau_1}^{e_1} \langle s_1, \nu_1 \rangle \to_{\tau_2}^{e_2} \dots$

$\to_{\tau_m}^{e_m} \langle s_m, \nu_m \rangle$.

The finite language of M is the set of all accepted behaviors of M.

In order to solve the emptiness problem for a timed automaton, Alur and Dill [1] have introduced a finite index equivalence relation over the state space of the automaton. The idea is to partition the set of the clock interpretations into a number of regions so that two clock interpretations in the same region will satisfy the same set of clock constraints.

For each $x \in X$, let K_x be the largest integer constant occurring in a clock constraint for the clock variable x of the timed automaton M, i.e.

$K_x =$ max$\{a | either x \leq a$ or $x \geq a$occursinaclockconstraint ofϕofatransitione $\}$.

Let $K_X = \max_{x \in X} K_x$.

For a real number r, let $frac(r) = r - \lfloor r \rfloor$ ($\lfloor r \rfloor$ is the maximal integer number which is not greater than r) be the fractional part of x. The equivalence relation $\cong$ over the set of clock interpretations is defined as follows: for two clock interpretations ν and ν', $\nu \cong \nu'$ iff the following three conditions are satisfied:

1. For all x ∈ X either $\nu(x) > K_x \wedge \nu'(x) > K_x$ or $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$.
2. For all x, y ∈ X such that $\nu(x) \leq K_x$ and $\nu(y) \leq K_y$, $frac(\nu(x)) \leq frac(\nu(y))$ iff $frac(\nu'(x)) \leq frac(\nu'(y))$.
3. For all x ∈ X such that $\nu(x) \leq K_x$, $frac(\nu(x)) = 0$ iff $frac(\nu'(x)) = 0$.

When $\nu \cong \nu'$, it is not difficult to see that for any clock constraint ϕ occurring in a transition $e = \langle s, a, \phi, \lambda, s' \rangle \in E$, ν satisfies ϕ iff ν' satisfies ϕ.

A clock region for M is an equivalence class of the clock interpretations induced by $\cong$. We denote by $[\nu]$ the clock region to which a clock interpretation ν belongs. From the definition of $\cong$, a region is characterized by the integer part of the value of each clock x when it is not greater than K_x, by the order between the fraction part of the clocks when they are different from 0. Therefore, the number of clock regions is bounded by $|X|! \cdot 2^{|X|} \cdot \prod_{x \in X} (2K_x + 2)$. A configuration is defined as a pair $\langle s, \alpha \rangle$ where $s \in L$ and α is a clock region. Based on the clock regions, the region automaton of M, whose states are configurations of M, and whose transitions are the combination of a time transition and a action transition from M. There is a time transition from $\langle s, \alpha \rangle$ to $\langle s, \beta \rangle$ iff $\beta = \alpha + t$ for some t (here for $\alpha = [\nu]$ we define $\alpha + t = [\nu + t]$).

Definition 4 (Region automata) Given a timed automaton M as in Definition 3, the region automaton of M is the automaton $\mathcal{R}(M) = \langle L', s'_I, \Sigma, E', \mathcal{F}' \rangle$, where

- The set of states L' consists of all configurations of M,
- $s'_I = \langle s_I, [\nu_\theta] \rangle$ where ν_θ is the clock valuation that assigns 0 to all clock variables in X,
- E' is the set of transitions of $\mathcal{R}(M)$ such that a transition $((s, \alpha), a, (s', \beta)) \in E'$ iff there is a timed transition from $\langle s, \alpha \rangle$ to $\langle s, \alpha' \rangle$ and a transition in $M\langle s, a, \phi, \lambda, s' \rangle$ such that α' satisfies ϕ and $\beta = [\lambda \mapsto 0]\alpha'$,
- $\mathcal{F}' \subseteq L'$ such that $s' \in \mathcal{F}'$ iff $s' = \langle s, \alpha \rangle$ where $s \in \mathcal{F}$ and α is a clock region.

Note that $\mathcal{R}(M)$ is a 'untimed' automaton, and we also denote its (untimed) language by $\mathcal{L}(\mathcal{R}(M))$.

We can simplify the automata M and $\mathcal{R}(M)$ such that all states (locations) are reachable and all states can lead to an acceptance state.

We recall some results from the timed automata theory [1] that will be used in our checking procedure later. Let $\mathcal{L}(M)$ denote the ω-timed language (language of infinite timed words) generated by M (by adding ε-transitions from a final state to itself we can extend the finite language of M to the ω language).

Theorem 1

1. For the timed automaton M, $untimed(\mathcal{L}(M)) = \mathcal{L}(\mathcal{R}(M))$. Therefore, the emptiness problem for M is decidable.
2. If $\langle s_0, \nu_0 \rangle \rightarrow^{e_1}_{\tau_1} \langle s_1, \nu_1 \rangle \rightarrow^{e_2}_{\tau_2} \ldots \rightarrow^{e_m}_{\tau_m} \langle s_m, \nu_m \rangle$ is a run from the initial state of M then $\langle s_0, [\nu_0] \rangle \rightarrow^{e_1} \langle s_1, [\nu_1] \rangle \rightarrow^{e_2} \ldots \rightarrow^{e_m} \langle s_m, [\nu_m] \rangle$ is a run of $\mathcal{R}(M)$, and reversely, if $\langle s_0, [\nu_0] \rangle \rightarrow^{e_1} \langle s_1, [\nu_1] \rangle \rightarrow^{e_2} \ldots \rightarrow^{e_m} \langle s_m, [\nu_m] \rangle$ is a run in $\mathcal{R}(M)$ then there are $\tau_1, \ldots, \tau_m$ such that $\langle s_0, \nu_0 \rangle \rightarrow^{e_1}_{\tau_1} \langle s_1, \nu_1 \rangle \rightarrow^{e_2}_{\tau_2} \ldots \rightarrow^{e_m}_{\tau_m} \langle s_m, \nu_m \rangle$ is a run from the initial state of M.

Let in the sequel, for an automaton M the size of M (the number of transitions and locations) be denoted by $|M|$.

Now, we return to the problem to decide if $untimed(\mathcal{L}(\mathcal{A}))|_{\Sigma_i} \subseteq R_i$ for a given timed automaton $\mathcal{A}$. It turns out that this problem is solvable, and just a corollary of Theorem 1.

Theorem 2 Given a regular expression R_i and a timed automaton $\mathcal{A}$ the problem $untimed(\mathcal{L}(\mathcal{A}))|_{\Sigma_i} \subseteq R_i$ is decidable in $\mathcal{O}(|\mathcal{R}(\mathcal{A})|.|R_i|)$ time.

Proof. Let $\mathcal{B}$ be an automaton that recognizes all the strings on Σ_i that do not belong to R_i, i.e. an automaton that recognizes

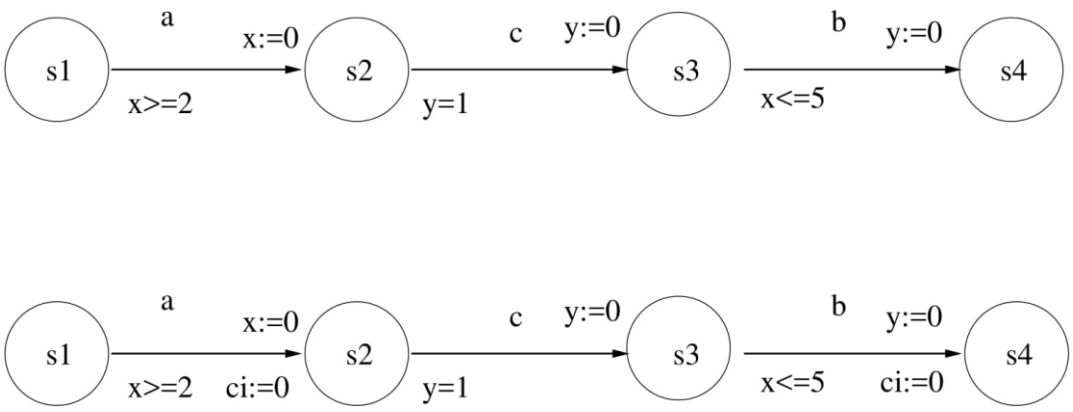

Fig. 1. Transitions in $\mathcal{A}$ and $\mathcal{A}'$: $a, b \in \Sigma_i, c \notin \Sigma_i$.

the complement $\bar{R}_i$ of R_i. The synchronized product $\mathcal{B} \times_{\Sigma_i} \mathcal{R}(\mathcal{A})$ recognizes the language $\bar{R}_i || \mathcal{L}(\mathcal{R}(\mathcal{A}))$ $(\{w \mid w|_{\Sigma_i} \in \bar{R}_i \wedge w|_{\Sigma}, \in \mathcal{L}(\mathcal{R}(\mathcal{A}))\})$. It follows Theorem 1 that $\bar{R}_i || \mathcal{L}(\mathcal{R}(\mathcal{A})) = \bar{R}_i || untimed(\mathcal{L}(\mathcal{A}))$. The emptiness of the language generated by $\mathcal{B} \times \mathcal{R}(\mathcal{A})$ is decidable in $\mathcal{O}(|\mathcal{R} \times \mathcal{R}(\mathcal{A})|)$ time. But $\bar{R}_i || untimed(\mathcal{L}(\mathcal{A}))$ is empty if and only if $untimed(\mathcal{L}(\mathcal{A}))|_{\Sigma_i} \subseteq R_i$. Hence, the theorem is proved.

Now we consider the problem to decide if all the strings generated by $\mathcal{A}$ satisfy the second item of Definition 2. Let $\mathcal{A} = \langle L, s_I, \Sigma, X, E, \mathcal{F} \rangle$. Let $\Sigma_i \subseteq \Sigma$. Let c_i be a new clock variable, $c_i \notin X$. Define $\mathcal{A}'$ to be the automaton that is the same as $\mathcal{A}$ except that transitions with label in Σ_i will have to reset the clock c_i as well, i.e. $\mathcal{A}' = \langle L, s_I, \Sigma, X \cup \{c_i\}, E', \mathcal{F} \rangle$, and $E' = \{e' = (s, a, \phi, C \cup \{c_i\}, s') \mid e = (s, a, \phi, C, s') \in E \wedge a \in \Sigma_i\} \cup \{e' = (s, a, \phi, C, s') \mid e = (s, a, \phi, C, s') \in E \wedge a \notin \Sigma_i\}$We illustrate the difference of transitions in $\mathcal{A}$ and $\mathcal{A}'$ in Fig. 1.

Since clock variable c_i does not appear in any guard ϕ of $\mathcal{A}$, the automaton $\mathcal{A}'$ generates the same timed language as $\mathcal{A}$ does. Adding the clock variable c_i is just for the purpose of counting time between two (consecutive) transitions in Σ_i. A clock valuation for $\mathcal{A}'$ now is of the form $\nu \cup \{c_i \mapsto v\}$ for some $v \in Reals$. Now we construct the region graph $\mathcal{R}(\mathcal{A}')$ for $\mathcal{A}'$, and analyze this graph to see if the second condition of Definition 2 is violated by a timed word from $\mathcal{L}(\mathcal{A})$. If $\delta(a) = 0$ for all $a \in \Sigma_i$, then the second condition for i is satisfied trivially. Otherwise, Theorem 1 gives that this condition is violated if and only if there is a run

$\langle s_0, [\nu_0] \rangle \rightarrow^{e_1} \langle s_1, [\nu_1] \rangle \rightarrow^{e_2} \ldots \rightarrow^{e_m} \langle s_m, [\nu_m] \rangle$

in $\mathcal{R}(\mathcal{A}')$ in which there are two transitions e_l and e_{l+h} corresponding to resetting clocks c_i in $\mathcal{A}'$: $e_l = (\langle s_l, [\nu_l] \rangle, a, \langle s_{l+1}, [\nu_{l+1}] \rangle$ where $a \in \Sigma_i$, $\nu_{l+1}(c_i) = 0$, and $e_{l+h} = (\langle s_{l+h}, [\nu_{l+h}] \rangle, b, \langle s_{l+h+1}, [\nu_{l+h+1}] \rangle$ where $b \in \Sigma_i$, $\nu_{l+h+1}(c_i) = 0$, and transitions $e_{l+1}, \ldots, e_{l+h-1}$ do not have label in Σ_i (not corresponding to transitions in $\mathcal{A}'$ resetting clock c_i) that makes the following condition satisfied: Let the run in $\mathcal{A}'$ according to Theorem 1 corresponding to that path be

$\langle s_l, \nu_l \rangle \rightarrow_{\tau_l}^{e_l} \ldots \rightarrow_{\tau_{l+h-1}}^{e_{l+h-1}}$

$\langle s_{l+h}, \nu_{l+h} \rangle \rightarrow_{\tau_{l+h}}^{e_{l+h}} \langle s_{l+h+1}, \nu_{l+h+1} \rangle$

Then, $\nu_{l+h}(c_i) + \tau_{l+h} < \delta(a)$. This implies the following: After having removed all non-reachable states from $\mathcal{R}(\mathcal{A}')$, and adding time transitions (labeled with "time") to $\mathcal{R}(\mathcal{A}')$, we have that there is also a path in $\mathcal{R}(\mathcal{A}')$

$\langle s_l, [\nu_l] \rangle \rightarrow^{e_l} \ldots \rightarrow^{e_{l+h-1}}$

$\langle s_{l+h}, [\nu_{l+h}] \rangle \rightarrow^{time}$

$\langle s_{l+h}, [\nu_{l+h} + \tau_{l+h}] \rangle \rightarrow^{e_{l+h}} \langle s_{l+h+1}, [\nu_{l+h+1}] \rangle$

in which $\nu_{l+h}(c_i) + \tau_{l+h} < \delta(a)$ where a is the label of e_l, and e_{l+h} has label in Σ_i. A path in $\mathcal{R}(\mathcal{A}')$ satisfying this condition is called "violation" path. Now, checking for the

violation of the second condition of Definition 2 from $\mathcal{A}$ is done by searching in the graph of $\mathcal{R}(\mathcal{A}')$ for a single path (not containing a loop) from e_l to e_{l+h} with the violation property as mentioned above (we call it violation path). If no such a path found, then the timed language $\mathcal{L}(\mathcal{A})$ satisfies the condition. This can be done in $\mathcal{O}(|\mathcal{R}(\mathcal{A}')|^2)$ time. Therefore, we have:

Theorem 3 The problem "if a given timed automaton $\mathcal{A}$ conforms to a real-time concurrent interaction protocol π" is decidable in time $\mathcal{O}(|\mathcal{R}(\mathcal{A}')|^2)$.

We sumarizes our results in the following deciding procedure:

Algorithm (Deciding if a timed automaton satisfies a real-time interaction protocol)

Input: A real-time protocol $\pi = \langle(\Sigma_1, R_1), \dots, (\Sigma_k, R_k), \delta\rangle$,
where $\delta: \bigcup_{i=1}^{k} \Sigma_i \to \mathbb{N}^{\geq}$, and R_i is a regular expression on Σ_i for $i = 1, \dots, k$.
A timed automaton $\mathcal{A} = \langle L, s_I, \Sigma, X, E, \mathcal{F}\rangle$ that satisfies $\Sigma_i \subseteq \Sigma$ for all $i \leq k$.

Output: "Yes" if $\mathcal{L}(\mathcal{A}) \vDash \pi$, "no" otherwise.

Methods:

1. Construct the region automaton of $\mathcal{A}$, namely the automaton $\mathcal{R}(\mathcal{A})$.
2. For each $i = 1, \dots, k$ construct automata $\mathcal{B}_i$ that recognizes regular language $\bar{R}_i$. Then, construct the synchronized product $\mathcal{R}(\mathcal{A}) \times_{\Sigma_i} \mathcal{B}_i$ and check if $\mathcal{L}(\mathcal{R}(\mathcal{A}) \times_{\Sigma_i} \mathcal{B}_i)$ is empty. If $\mathcal{L}(\mathcal{R}(\mathcal{A}) \times_{\Sigma_i} B_i)$ is not empty for some i, stop with output "no".
3. If there is no time constraint in π, i.e. δ is 0 mapping on Σ, stop with output "yes".
4. For each $i = 1, \dots, k$, where δ is not a 0-mapping on Σ_i, construct the timed automaton $\mathcal{A}' = \langle \mathrm{L}, \mathrm{s_I}, \Sigma, \mathrm{X} \cup \{\mathrm{c_i}\}, \mathrm{E}', \mathcal{F}\rangle$, where $\mathrm{E}' = \{\mathrm{e}' = (\mathrm{s}, \mathrm{a}, \phi, \mathrm{C} \cup \{\mathrm{c_i}\}, \mathrm{s}') \mid \mathrm{e} = (\mathrm{s}, \mathrm{a}, \phi, \mathrm{C}, \mathrm{s}') \in \mathrm{E} \wedge \mathrm{a} \in \Sigma_\mathrm{i}\} \cup \{\mathrm{e}' = (\mathrm{s}, \mathrm{a}, \phi, \mathrm{C}, \mathrm{s}') \mid \mathrm{e} = (\mathrm{s}, \mathrm{a}, \phi, \mathrm{C}, \mathrm{s}') \in \mathrm{E} \wedge \mathrm{a} \notin \Sigma_\mathrm{i}\}$, and then construct the region graph $\mathcal{R}(\mathcal{A}')$. Add all "time" transitions to $\mathcal{R}(\mathcal{A}')$ and simplify it by removing all nonreachable states. Search in $\mathcal{R}(\mathcal{A}')$ for a single violation path. If such a path is found for some i, stop with the output "no".
5. Stop with the output "yes".

Note that a concurrent real-time system can be modeled as a timed automata network which is a synchronized product of a number of timed automata, where the concurrency can be expressed explicitly. A synchronized product of a number of timed automata is also a timed automaton, and hence, our algorithm works also on timed automata networks.

4. Conclusion

We have proposed a simple but powerful technique to specify interaction protocols for the interface of components. Our model can specify many aspects for interaction: the temporal order between services, concurrency for services, and timing constraints. We also have shown that the problem of checking if a timed automaton conforms to a given real-time protocol is decidable, and developed a decision procedure for solving the problem. The complexity of the procedure is proportional to the size of the region graph of the input timed automaton which is acceptable for many cases (like the way that the tool UPAAL handles systems). We will incorporate this technique to our model for real-time component-based systems in our future work. We believe that our results can be extended to the cases in which systems are modeled by timed automata with parameters, i.e. timed automata where a parameter can appear in guards and can be reset by a transition.

Acknowledgments

This research was funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.03-2014.23.

References

[1] R. Alur and D.L. Dill. A Theory of Timed Automata. Theoretical Computer Science, pages 183-235, 1994.

[2] Luca de Alfaro and Thomas A. Henzinger. Interface Automata. In ACM Symposium on Foundation of Software Engineering (FSE), 2001.

[3] Jifeng He, Zhiming Liu, and Xiaoshan Li. rCOS: A refinement calculus of object systems. Theor. Comput. Sci., 365(1-2):109-142, 2006. UNU-IIST TR 322.

[4] Dang Van Hung and Hoang Truong. Modeling and specification of real-time interfaces with UTP. In Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday, pages 136-150, 2013.

[5] Hung Ledang and Dang Van Hung. Timing and concurrency specification in component-based real-time embedded systems development. In TASE, pages 293-304. IEEE Computer Society, 2007.

[6] Stavros Tripakis, Ben Lickly, Thomas A. Henzinger, and Edward A. Lee. On relational interfaces. In EMSOFT'09, pages 67-76. ACM, 2009.

[7] Dang Van Hung. Toward a formal model for component interfaces for real-time systems. In Proceedings of the 10th international workshop on Formal methods for industrial critical systems, FMICS '05, pages 106-114, New York, NY, USA, 2005. ACM.