Real-time Property Preservation in Concurrent Real-time Systems*

Jinfeng Huang, Jeroen Voeten and Marc Geilen

Eindhoven Univ. of Tech. Faculty of Electrical Engineering, The Netherlands
J.Huang@tue.nl

Abstract. A key step in concurrent real-time system development is to build a model from which the implementation is synthesized. It is thus important to understand the relation between the properties of a model and its corresponding implementation. In this paper, we first build two relations: 1) ϵ-weakening relations on $MITL_{\mathbb{R}}$ formulas, which are used to express real-time properties of the system, and 2) ϵ-neighboring relations on timed state sequences, which are used to describe the timing behavior of the system. Based on these relations, we formally prove the real-time property preservation in approximations of concurrent real-time systems. This result generalizes [11], which is restricted to sequential real-time systems. Finally, we demonstrate how the result can be applied to the real-time system development by a case study of a rail-road crossing system.

1 Introduction

A concurrent system often exhibits complex behaviors due to intricate interactions between its components, while real-time constraints add another dimension to the complexity of the system. To reduce the development risk and increase the likelihood of the development success, the development of a complex concurrent real-time system often involves a series of modelling activities. Developers can make design and implementation decisions based on the model of the system before the system is realized. As a result, expensive and time-consuming development iterations can be prevented.

Different from other systems, the correctness of a real-time system depends not only on the system outputs, but on their issued time as well. However, the model can only be an approximation of the implementation w.r.t. the timing behavior. It is difficult to guarantee that the issuing time of an event in the implementation is exactly the same as that in the model [9][11]. Since it is inevitable that the implementation can "deviate" from the model to a certain extent with respect to time, it is desirable to know the relation between properties in the implementation and properties in the model.

In [11], the implementation and the model of a real-time system are defined as two sets of timed state sequences. Properties of the implementation can be predicted from properties of the model, when the implementation is "ϵ-close" to the model. However,

* This research is supported by PROGRESS, the embedded systems research program of the Dutch organisation for Scientific Research NWO, the Dutch Ministry of Economic Affairs, the Technology Foundation STW and the Netherlands Organisation for Applied Scientific Research TNO.

concurrency is not addressed in [11]. In this paper, we generalize the result of [11] to concurrent systems (with interleaving semantics) and demonstrate how this result can be applied to the real-time control software development by a case study of a rail-road crossing system.

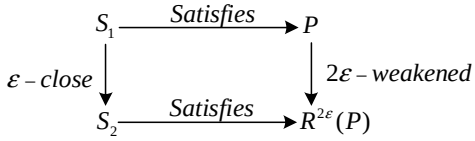


Fig. 1. Property-preservation between two timed systems

The remainder of the paper is organized as follows. Section 2 introduces the mathematical preliminaries. In Section 3, a temporal logic to formalize properties of real-time systems is introduced and a weakening relation is quantitatively defined on these real-time properties. In Section 4, we introduce a metric to measure the distance between timed state sequences. An equivalent representation of this metric in form of the ϵ-neighboring function is also defined to facilitate later proof. In Section 5, the real-time property preservation is explained and proven. An informal description of the result is illustrated in Fig. 1. If system S_1 satisfies property P, then system S_2 (which is ϵ close to S_1) satisfies a 2ϵ-weakened property of P. An application of property-preservation in a development approach for real-time systems is introduced in Section 6. Section 7 gives a case study of the rail-road crossing system using this approach. Conclusions and discussions are presented in Section 8.

2 Preliminaries

2.1 Basic concepts

In this paper, a real-time system is formalized as a set of timed state sequences. Each sequence represents a possible execution path of the real-time system. The related concepts are as follows.

Propositions: Proposition set $Prop$ is a (finite) set of atomic propositions. The observable state of a system can be interpreted by a subset of $Prop$, which includes all true-valued propositions.

State sequences: A state sequence $\overline{\sigma}$ over proposition set $Prop$ is an infinite or finite sequence $\sigma_0\sigma_1\sigma_2...$, where $i \in \mathbb{N}$ and $\sigma_i \in 2^{Prop}$. We use $\overline{\sigma}(i)$ to represent σ_i in the sequence. The untimed behavior of a system can be viewed as a set of state sequences, each of which represents a possible untimed execution trace.

Time intervals: A time interval I is a convex set of time points over a certain time domain $\mathbb{T}$, which is often defined on $\mathbb{R}^{\geq 0}$, $\mathbb{Z}^{\geq 0}$ or $\mathbb{N}$. It has one of the following forms: $[a, b]$, $[a, b)$ and $[a, \infty)$, where $a \leq b$ for $a, b \in \mathbb{T}$. The lower (upper) bound of the interval is represented by $l(I)(r(I))$. $|I|$ denotes the length of time interval I. Note that if the right bound of time interval I is infinite, $|I|$ is infinite too. A time interval is *singular* iff it takes the form of $[a, a]$. A time interval is *empty* iff it takes the form of $[a, a)$. Two *nonempty* time intervals I and I' are adjacent iff $r(I) = l(I')$. For example, the following interval pairs are adjacent, $[1.2, 3)$ and $[3, 4)$, [1.2, 1.2] and [1.2, 1.3). In the sequel, we assume that time intervals are nonempty if not explicitly stated otherwise.

Time interval sequences: A time interval sequence $\overline{I} = I_0 I_1 I_2 ...$ is a sequence of *nonempty* time intervals such that 1) Any two successive intervals of $\overline{I}$ are adjacent, that is, $r(I_{i-1}) = l(I_i)$ for every $i < \mathcal{N}(\overline{I})$ and $i \in \mathbb{N}$, where $\mathcal{N}(\overline{I})$ represents the number of time intervals in $\overline{I}$ and can be finite or countable infinite. 2) It is diverging, which indicates that for any $t \geq l(I_0)$, there exists some $i \in \mathbb{N} \cup \{0\}$, such that $t \in I_i$.

Timed state sequences: A timed execution of a system is formalized by a timed state sequence in which a time interval is attached to every state to represent its duration. An example of timed state sequence $\overline{\tau}$ is:

$$(\delta_0, I_0) \rightarrow (\delta_1, I_1) \rightarrow (\delta_2, I_2) \rightarrow ...,$$

where all δ_i from each pair form a state sequence $\overline{\delta}$ and all I_i form a time interval sequence $\overline{I}$. Therefore, the length of $\overline{\delta}$ and $\overline{I}$ is identical. A timed state sequence over proposition set $Prop$ is also represented by a pair of sequences $(\overline{\delta}, \overline{I})$, where $\overline{\delta}$ is a state sequence over $Prop$. We use $\overline{\tau}(t)$ to denote the finite state sequence[1] observed at time t.

2.2 Formalization of concurrent real-time systems

The well-established interleaving semantics has been adopted in many formal frameworks to model simultaneous actions (or events) by action sequences. For example, two parallel actions $a \parallel b$ are represented by two sequential actions $a.b$ and $b.a$. Such sequentialization of concurrent actions is greatly facilities calculations and it yields correctness proofs in a linear form [4].

Under the choice of interleaving semantics, the timing behavior of a concurrent system is often formalized by a two-phase execution model [14]. One phase is the advancing of time (the abstraction of the physical time) and no action takes place during this phase. The other phase is the performing of simultaneous actions sequentially (based on interleaving semantics). The two-phase model can be expressed by timed state sequences defined above, in which a singular time interval is attached to every instantaneous state. Simultaneous actions can be modelled as sequentialized instantaneous (state) transition sequences, where the duration of intermediate states of these transitions are zero. Assume for instance that a state transition system performs two concurrent events p and q at time point 3. One of the action sequences can be $p.q$, which can be represented as follows:

$$... \rightarrow (\delta_0, [a, 3)) \xrightarrow{p} (\delta_1, [3, 3]) \xrightarrow{q} (\delta_2, [3, b)) \rightarrow ..., \quad (1)$$

where a (b) represents the issuing time of the preceding action (the succeeding action) and where a singular time interval $[3, 3]$ is attached to instantaneous state δ_1.

Timed state sequences defined above can be considered as a function, in which every time point $t \in \mathbb{R}$ is mapped to a finite state sequence. Some timed state sequences with different state sequences and time interval sequences may be considered to be identical to each other. For example, timed state sequences $\overline{\tau}$ and $\overline{\tau}'$ defined as follows represent the same timed execution trace of a system.

$\overline{\tau}: \ (\delta_0, [0, 0]) \rightarrow ... \rightarrow (\delta_{2i}, [i, i]) \rightarrow (\delta_{2i+1}, [i, i + 1)) \rightarrow ...$

$\overline{\tau}': \ (\delta_0, [0, 0]) \rightarrow ... \rightarrow (\delta_{2i}, [i, i]) \rightarrow (\delta_{2i}, [i, i]) \rightarrow (\delta_{2i+1}, [i, i + 1)) \rightarrow ...$

[1] The diverging property of the timed interval sequence in a timed state sequence is also called the Non-Zeno property, which ensures that only a finite number of states can be observed at any time point.

Definition 1. *Two timed state sequences* $\overline{\tau} = (\overline{\sigma}, \overline{I})$ *and* $\overline{\tau}' = (\overline{\sigma}', \overline{I}')$ *are equivalent* ($\equiv$) *iff for all* $t \in \mathbb{R}^{\geq l(I_0)}$, $\overline{\tau}(t) = \overline{\tau}'(t)$.

Any timed state sequence $\overline{\tau}$ can be normalized to a timed state sequence $\overline{\tau}^*$, the normal form of $\overline{\tau}$, by performing the following operation. Along the state sequence of $\overline{\tau}$, successive identical states are replaced by one single state, and their corresponding time intervals are merged into one single time interval. In the above example, if no two sequential states δ_i and δ_{i+1} ($i \in \mathbb{N}$) are identical, then $\overline{\tau}$ is the normal form of $\overline{\tau}'$.

2.3 Labelling time interval sequences

As mentioned before, a timed state sequence $(\overline{I}, \overline{\delta})$ is viewed as a mapping from each time point $t \in \mathbb{R}^{\geq l(I_0)}$ to a finite state sequence. As a consequence, the order of interleaved simultaneous actions can not be discriminated only by their observed time points. For instance, in sequence (1), two states (δ_1 and δ_2) are both observed at time point 3, which is not sufficient enough to distinguish the order of them. To solve this problem, a labelling method on time points is applied to time interval sequences in timed state sequences to distinguish the order of interleaved simultaneous actions.

In timed state sequence $\overline{\tau}$, suppose the number of states at time instant t is $m_t^{\overline{\tau}}$, which is finite. The issuing time of an action is represented by a time pair $\langle t, i\rangle$, in which i is the label of t and represents the interleaving order of observed states at time t. In other word, a time pair $\langle t, i\rangle$ can be seen as the integration of a metric "macro-time" $t \in R^{\geq l(I_0)}$ and a linearly ordered discrete "micro-time" $1 \leq i \leq m_t^{\overline{\tau}}$ [2]. Correspondingly, a time interval sequence $\overline{I}$ in a timed state sequence is extended to a labelled time interval sequence $\overline{\mathcal{I}}$. All time points in a time interval sequence are labelled with 1 except in the following two cases.

- End points of a singular time interval. The right end point is always labelled with the same number as the left end point. For instance, $[7, 7]$ is labelled with $[\langle 7, i\rangle, \langle 7, i\rangle]$.
- Adjacent closed end points. Consider two adjacent time intervals, where the first time interval is right-closed and the second is left-closed. In this case, if the right end point of the first is labelled with i, the label of the left end point of the second interval is $i + 1$. For instance, time interval sequence $[3, 5)[5, \mathbf{5}][\mathbf{5}, \infty)$ is labelled as $\mathbf{[}\langle 3, 1\rangle, \langle 5, 1\rangle\mathbf{)[}\langle 5, 1\rangle, \langle \mathbf{5}, \mathbf{1}\rangle\mathbf{][}\langle \mathbf{5}, \mathbf{2}\rangle, \infty\mathbf{)}$, while the remaining time points in the time interval are labelled with 1, for instant time point 6 is labelled as $\langle 6, 1\rangle$.

A timed state sequence can be labelled in the same way as above. For instance, timed state sequence (1) is labelled as [2]:

$$... \rightarrow (\delta_0, \mathbf{[}\langle a, 1\rangle, \langle 3, 1\rangle\mathbf{)}) \xrightarrow{p} (\delta_1, \mathbf{[}\langle 3, 1\rangle, \langle 3, 1\rangle\mathbf{]}) \xrightarrow{q} (\delta_2, \mathbf{[}\langle 3, 2\rangle, \langle b, 1\rangle\mathbf{)}) \rightarrow ...,$$

in which other internal time points are all labelled with 1. By applying this labelling to the time interval sequence of a timed state sequence, we can view a timed state sequence $\overline{\tau}$ as a function which maps every time pair in the time domain [3] $\mathbb{T}^{\overline{I}} = \{\langle t, i\rangle \mid t \in$

[2] Assume only one action is issued at time point a and $0 \leq a < 3$.

[3] In [14], a time domain is formally denoted as an additive group (T, +,0), where a total order is defined by the addition operations on T. The time domain here only requires a total order relation on its elements without an explicit definition of the addition operation on it.

$\mathbb{R}^{\geq l(I_0)} \wedge 1 \leq i \leq m_t^{\overline{\tau}}\}$ to a state in 2^{Prop}. A total order $<$ on time pairs is defined as the following. Let $\langle t, i\rangle$ and $\langle t', j\rangle$ be two time pairs in $\mathbb{T}^{\overline{I}}$. $\langle t, i\rangle < \langle t', j\rangle$ iff $t < t'$ or ($t = t'$ and $i < j$), $\langle t, i\rangle = \langle t', j\rangle$ iff $t = t'$ and $i = j$. The distance between them is defined as $|\langle t, i\rangle - \langle t', j\rangle| = |t - t'|$. Similar to time intervals, the lower (upper) bound of labelled time interval $\mathcal{I}$ is represented by $l(\mathcal{I})$ ($r(\mathcal{I})$). $|\mathcal{I}|$ denotes the length of $\mathcal{I}$.

Remark 1. Let $\overline{I}$ be a time interval sequence and $\overline{\mathcal{I}}$ be its labelled time interval sequence. For any $k < \mathcal{N}(\overline{I})$ and $k \in \mathbb{N}$, there is a bijection G_k between I_k and its labelled time interval $\mathcal{I}_k$, such that for any $\langle t, i\rangle \in \mathcal{I}_k$, $G_k(t, i) = t$, where $t \in I_k$.

In the sequel, we use $\overline{\mathcal{I}}$ to represent the labelled time interval sequence of $\overline{I}$, $\mathcal{I}_k$ to represent the corresponding labelled time interval of I_k and $\mathbb{T}^{\overline{\mathcal{I}}}$ to represent the labelled time domain $\mathbb{T}^{\overline{I}}$. On the other hand, the unlabelled time interval (i.e. that results from removing the label of every time pair) of $\mathcal{I}_k$ (or unlabelled time interval sequence of $\overline{\mathcal{I}}$) is represented by I_k(or $\overline{I}$). $\mathbb{N}(\overline{\mathcal{I}})$ represents the number of labelled time intervals in $\overline{\mathcal{I}}$.

3 Properties of concurrent real-time systems

Properties of real-time systems are often formalized by temporal logics. In [2], a comparison of a number of temporal logics is given based on a common semantics (timed state sequences). In this paper, we adopt an extension of temporal logic $MITL$ to formalize properties of real-time systems [1], which incorporates timing bounds into the temporal logic to express quantitative real-time properties.

3.1 $MITL_{\mathbb{R}}$ logic

$MITL_{\mathbb{R}}$ formulas are formed by the following structures.

$$\varphi ::= p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathsf{U}_I \varphi_2 \mid \varphi_1 \mathsf{V}_I \varphi_2 \quad ,$$

where I is a time-bound interval of nonnegative reals. It takes one of the following forms: $[a, b]$, $[a, b)$, $(a, b]$, (a, b), $[a, \infty)$ and (a, ∞), where $a \leq b$ for $a, b \in \mathbb{R}^{\geq 0}$. The standard $MITL$ puts certain constraints on the time bounds of a $MITL$ formula to make model-checking feasible, such as non-empty, non-singular and integer endpoints. Note that these constrains are not necessary in the case of $MITL_{\mathbb{R}}$, since $MITL_{\mathbb{R}}$ logic is used only for expressing real-time properties of a system instead of model-checking.

The interpretation of $MITL_{\mathbb{R}}$ formulas is similar to the standard one and given as follows. Let $\overline{\tau}$ be a timed state sequence and let φ be an $MITL_{\mathbb{R}}$ formula, for any $t \in \mathbb{R}^{\geq l(I_0)}$ and any $1 \leq i \leq m_t^{\overline{\tau}}$, the interpretation of φ over $(\overline{\tau}, \langle t, i\rangle)$ is given as follows:

- $(\overline{\tau}, \langle t, i\rangle) \models p$ iff $p \in \overline{\tau}(t, i)$;
- $(\overline{\tau}, \langle t, i\rangle) \models \neg p$ iff $p \notin \overline{\tau}(t, i)$;
- $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \vee \varphi_2$ iff $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1$ or $(\overline{\tau}, \langle t, i\rangle) \models \varphi_2$;
- $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \wedge \varphi_2$ iff $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1$ and $(\overline{\tau}, \langle t, i\rangle) \models \varphi_2$;
- $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \mathsf{U}_I \varphi_2$ iff there exists t_2 ($t_2 \in I$), j ($1 \leq j \leq m_{t+t_2}^{\overline{\tau}}$) and $\langle t, i\rangle \leq \langle t + t_2, j\rangle$, such that $(\overline{\tau}, \langle t + t_2, j\rangle) \models \varphi_2$ and for all t_1 and k ($1 \leq k \leq m_{t_1}^{\overline{\tau}}$) that satisfy $\langle t, i\rangle \leq \langle t_1, k\rangle < \langle t + t_2, j\rangle$, $(\overline{\tau}, \langle t_1, k\rangle) \models \varphi_1$;

– $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \mathsf{V}_I \varphi_2$ iff for all t_2 ($t_2 \in I$), j ($1 \leq j \leq m^{\overline{\tau}}_{t+t_2}$) and $\langle t, i\rangle \leq \langle t + t_2, j\rangle$, $(\overline{\tau}, \langle t + t_2, j\rangle) \models \varphi_2$ or there is some t_1 and k ($1 \leq k \leq m^{\overline{\tau}}_{t_1}$) $\langle t, i\rangle \leq \langle t_1, k\rangle < \langle t + t_2, j\rangle$, $(\overline{\tau}, \langle t_1, k\rangle) \models \varphi_1$.

In case that I is empty, $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \mathsf{U}_I \varphi_2$ is always false and $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1 \mathsf{V}_I \varphi_2$ always holds. We use $(\overline{\tau}, \langle l(I_0), 1\rangle) \models \varphi$ ($\overline{\tau} \models \varphi$, in short) to denote that a timed state sequence $\overline{\tau}$ satisfies $MITL_{\mathbb{R}}$ formula φ. Next, we extend the interpretation of $MITL_{\mathbb{R}}$ to sets of timed state sequences.

Definition 2. *Let φ be an $MITL_{\mathbb{R}}$ formula, and let T be a set of timed state sequences. $T \models \varphi$ iff for each timed state sequence $\overline{\tau} \in T$, $\overline{\tau} \models \varphi$.*

Here we introduce two additional operators: $\Diamond_I \varphi$ (time-bounded eventually) and $\Box_I \varphi$ (time-bounded always) to abbreviate $true \mathsf{U}_I \varphi$ and $false \mathsf{V}_I \varphi$ respectively.

3.2 Weakening formulas

After giving the interpretation for $MITL_{\mathbb{R}}$ formulas, we can now investigate the weakening relations between formulas. Formally speaking, a $MITL_{\mathbb{R}}$ formula φ' is a weakened formula of φ, iff for any timed state sequence $\overline{\tau}$ and $\langle t, i\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$, $(\overline{\tau}, \langle t, i\rangle) \models \varphi$ implies $(\overline{\tau}, \langle t, i\rangle) \models \varphi'$.

Theorem 1. *For any $MITL_{\mathbb{R}}$ formula φ, let ϕ be a sub-formula of φ. If ϕ' is a weakened formula of ϕ and φ' is the formula obtained by replacing one occurrence of ϕ with ϕ' in φ, then φ' is a weakened formula of φ.*

Proof. If $\phi = \varphi$, it is easy to see $\varphi' = \phi'$. Hence φ' is a weakened formula of φ.
If $\phi \neq \varphi$, we can prove the theorem by induction on the structure of φ, which can be decomposed into two sub-formulas φ_1 and φ_2 and ϕ is a sub-formula of one of them.
Case 1: $\varphi = \varphi_1 \vee \varphi_2$. Without loss of generality, assume that ϕ belongs to φ_1. Let φ'_1 be the formula obtained by replacing ϕ with ϕ'. By induction we have that φ'_1 is a weakened formula of φ_1. Then, by the interpretation of $MITL_{\mathbb{R}}$ formula, it is easy to prove that φ' ($\varphi'_1 \vee \varphi_2$) is a weakened formula of φ.
The proof of the rest cases are similar to Case 1.
Hence, φ' is a weakened formula of φ. This completes our proof of Theorem 1.

The following Lemmas show that the inclusive relation between the quantitative timing bounds of $MITL_{\mathbb{R}}$ formulas can also imply the weakening relation among them.

Lemma 1. *Let I and I' be two time bounds. Further let φ_1, φ_2 be two $MITL_{\mathbb{R}}$ formulas. If $I \subseteq I'$, then $\varphi_1 \mathsf{U}_{I'} \varphi_2$ is a weakened formula of $\varphi_1 \mathsf{U}_I \varphi_2$.*

Proof. By the interpretation of $MITL_{\mathbb{R}}$ formulas, the proof is straightforward.

Lemma 2. *Let I and I' be two time bounds. Further let φ_1, φ_2 be two $MITL_{\mathbb{R}}$ formulas. If $I' \subseteq I$, then $\varphi_1 \mathsf{V}_{I'} \varphi_2$ is a weakened formula of $\varphi_1 \mathsf{V}_I \varphi_2$.*

Proof. By the interpretation of $MITL_{\mathbb{R}}$ formulas, the proof is straightforward.

3.3 ϵ-weakening function

After a general introduction of the weakening and strengthening relations between $MITL_{\mathbb{R}}$ formulas, in this subsection, we define a specific weakening function (ϵ-weakening function) on these formulas. Before giving the definition of the ϵ-weakening function, we first define two useful operators ($\oplus$ and $\ominus$) on time-bound intervals.

Informally speaking, $I \oplus \epsilon$ represents the time interval which elongates the end points of I to $l(I) - \epsilon$ and $r(I) + \epsilon$ respectively in nonnegative reals. Similarly, $I \ominus \epsilon$ represents the time interval which shrinks the end points of I to $l(I) + \epsilon$ and $r(I) - \epsilon$ respectively. Several examples are: $[2\pi, 4\pi] \oplus \pi = [\pi, 5\pi]$, $(3, \infty) \oplus 5 = [0, \infty)$ and $[1.1, 2) \ominus 3 = \emptyset$. The formal definition of $\oplus$ and $\ominus$ is given as:

Definition 3. *Let $\epsilon \in \mathbb{R}^{\geq 0}$. $\oplus$ and $\ominus$ are defined as:*

$$I \oplus \epsilon = \{t \in \mathbb{R}^{\geq 0} \mid \exists t' \in I : |t - t'| \leq \epsilon\}$$
$$I \ominus \epsilon = \{t \in \mathbb{R}^{\geq 0} \mid \forall t' \in \mathbb{R}^{\geq 0} : |t - t'| \leq \epsilon \rightarrow t' \in I\}$$

Now we can give the definition of the ϵ-weakening function. For every $\epsilon \in \mathbb{R}^{\geq 0}$, R^{ϵ} defines a function over $MITL_{\mathbb{R}}$. $R^{\epsilon}(\varphi)$ relaxes the quantitative timing constraints in formula φ and is called the ϵ-weakened formula of φ.

Definition 4. *Weakening function R^{ϵ} ($\epsilon \in \mathbb{R}^{\geq 0}$): $MITL_{\mathbb{R}} \rightarrow MITL_{\mathbb{R}}$ is defined as:*

$$R^{\epsilon}(p) = p;$$
$$R^{\epsilon}(\neg p) = \neg p;$$
$$R^{\epsilon}(\varphi_1 \vee \varphi_2) = R^{\epsilon}(\varphi_1) \vee R^{\epsilon}(\varphi_2);$$
$$R^{\epsilon}(\varphi_1 \wedge \varphi_2) = R^{\epsilon}(\varphi_1) \wedge R^{\epsilon}(\varphi_2);$$
$$R^{\epsilon}(\varphi_1 \mathsf{U}_I \varphi_2) = R^{\epsilon}(\varphi_1) \mathsf{U}_{I \oplus \epsilon} R^{\epsilon}(\varphi_2);$$
$$R^{\epsilon}(\varphi_1 \mathsf{V}_I \varphi_2) = R^{\epsilon}(\varphi_1) \mathsf{V}_{I \ominus \epsilon} R^{\epsilon}(\varphi_2)$$

For example, the 0.1-weakened formula of $(p \mathsf{V}_{[2,4]} q) \mathsf{U}_{(1,\infty)} r$ is $(p \mathsf{V}_{[2.1,3.9]} q) \mathsf{U}_{(0.9,\infty)} r$. Next, we will prove that formula $R^{\epsilon}(\varphi)$ is indeed weaker than formula φ.

Lemma 3. *For any $\epsilon \in \mathbb{R}^{\geq 0}$, $\varphi \in MITL_{\mathbb{R}}$, timed state sequence $\overline{\tau}$, $t \in \mathbb{R}^{\geq l(I_0)}$ and $1 \leq i \leq m_t^{\overline{\tau}}$, if $(\overline{\tau}, \langle t, i \rangle) \models \varphi$, then $(\overline{\tau}, \langle t, i \rangle) \models R^{\epsilon}(\varphi)$.*

Proof. It is not hard to prove by Lemma 1, Lemma 2, Theorem 1 and the induction on the structure of formula $R^{\epsilon}(\varphi)$.

Theorem 2. (Weakening property of R^{ϵ}) *For any $\epsilon \in \mathbb{R}^{\geq 0}$, $\varphi \in MITL_{\mathbb{R}}$ and timed state sequence $\overline{\tau}$, if $\overline{\tau} \models \varphi$, then $\overline{\tau} \models R^{\epsilon}(\varphi)$.*

Proof. Follows directly from Lemma 3.

In general, the larger the value of ϵ is, the weaker is formula $R^{\epsilon}(\varphi)$.

4 Behaviors of concurrent real-time systems

The model and the implementation of a concurrent real-time system can be viewed as timed transition systems consisting of a set of timed state sequences. To investigate the property-preservation between them, we need to measure the distance between two timed transition systems. To this end, a metric over timed state sequences is first defined.

4.1 Measuring the distance between timed state sequences

Let set S_{Prop} consist of all timed state sequences in normal form over a proposition set $Prop$. We define an equivalence relation ($\equiv_s$) that partitions S_{Prop} into groups sharing the same state sequence $\overline{\sigma}$.

Definition 5. *Equivalence relation ($\equiv_s$) on set S_{Prop}: for any two timed state sequences $\overline{\tau}, \overline{\tau}' \in S_{Prop}$, if $\overline{\tau} = (\overline{\sigma}, \overline{\mathcal{I}})$ and $\overline{\tau}' = (\overline{\sigma}', \overline{\mathcal{I}}')$, then $\overline{\tau} \equiv_s \overline{\tau}'$ iff $\overline{\sigma} = \overline{\sigma}'$.*

In order to evaluate the distance between two timed state sequences in S_{Prop}, we adopt a metric d_{sup}. Intuitively, for two timed sequences with the same state sequence, d_{sup} is calculated from the absolute difference between the time-stamps of corresponding state transitions in two state sequences. An example of such a case is shown in Fig. 2. There are four state transitions in each timed state sequence. $d_{sup}(\overline{\tau_1}, \overline{\tau_2})$ is computed as follows: $sup\{|\langle 0,1\rangle - \langle 0,1\rangle|, |\langle 2.7,1\rangle - \langle 3,2\rangle|, |\langle 3.2,1\rangle - \langle 3,3\rangle|, |\langle 6,1\rangle - \langle 5.5,1\rangle|\} = sup\{|0-0|, |2.7-3|, |3.2-3|, |6-5.5|\} = 0.5$.

In other words, the distance between two timed state sequences which share the same state sequence is determined by their labelled timed interval sequences. Therefore, we first formally define a metric d_{int} on labelled timed interval sequences. Then the definition of metric d_{sup} on timed state sequences is straightforward.

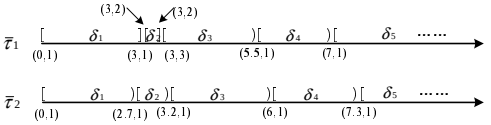


Fig. 2. Two finite timed state sequences

Definition 6. *Let $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}}'$ be two labelled time interval sequences* [4].

$$d_{int}(\overline{\mathcal{I}}, \overline{\mathcal{I}}') = \begin{cases} sup\{|l(\mathcal{I}_k) - l(\mathcal{I}'_k)| \mid i < N(\overline{\mathcal{I}}) \wedge i \in \mathbb{N}\} & if\ N(\overline{\mathcal{I}}) = N(\overline{\mathcal{I}}'); \\ \infty & otherwise. \end{cases}$$

Lemma 4. *Let $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}}'$ be two labelled time interval sequences. $d_{int}(\overline{\mathcal{I}}, \overline{\mathcal{I}}') < \epsilon$ implies that for any $k < N(\overline{\mathcal{I}})$, $|r(\mathcal{I}_k) - r(\mathcal{I}'_k)| < \epsilon$* [5].

Proof. For any $k < N(\overline{\mathcal{I}})$, let I_k and I'_k be unlabelled time intervals of $\mathcal{I}_k$ and $\mathcal{I}'_k$ respectively. Notice that for any $k < N(\overline{\mathcal{I}})$, $r(I_{k-1}) = l(I_k)$, $r(I'_{k-1}) = l(I'_k)$ and $|r(\mathcal{I}_k) - r(\mathcal{I}'_k)| = |r(I_k) - r(I'_k)|$. The rest is straightforward by Definition 6.

Definition 7. *Let $\overline{\tau}, \overline{\tau}' \in S_{Prop}$, and $\mathcal{I}$ and $\mathcal{I}'$ be their labelled timed interval sequences respectively. Then $d_{sup}(\overline{\tau}, \overline{\tau}')$ is defined as*

$$d_{sup}(\overline{\tau}, \overline{\tau}') = \begin{cases} d_{int}(\overline{\mathcal{I}}, \overline{\mathcal{I}}')\ if\ \overline{\tau} \equiv_s \overline{\tau}'; \\ \infty \qquad otherwise \end{cases}$$

[4] In $\mathbb{R}$, the supremum (sup) only exists on bounded sets. In this definition, we define $supS = \infty$ when $S \subseteq \mathbb{R}^{\geq 0}$ and is unbounded.

[5] In the case that both $r(\mathcal{I}_k)$ and $r(\mathcal{I}'_k)$ are ∞, we define $|r(\mathcal{I}_k) - r(\mathcal{I}'_k)| = 0$

4.2 ϵ-neighboring function $\mathcal{F}$ over two labelled time intervals

The metric d_{int} in Definition 6 establishes a relation between time pairs from two time interval sequences. In this section, the relation is characterized by an ϵ-neighboring function, which is first defined on two time intervals.

Definition 8. *Let $\epsilon \in \mathbb{R}^{\geq 0}$. Further let $\mathcal{I}$ and $\mathcal{I}'$ be two labelled time intervals. $\mathcal{F}$ is a multi-valued function [6] from $\mathcal{I}$ to $\mathcal{I}'$ and for any $\langle t, i\rangle \in \mathcal{I}$, $\mathcal{F}(t,i)$ represents the set of images of time pair $\langle t, i\rangle$. $\mathcal{F}$ is called an ϵ-neighboring function from $\mathcal{I}$ to $\mathcal{I}'$, iff the following properties are satisfied.*

- ***Surjection:*** *For any $\langle t', i'\rangle \in \mathcal{I}'$, there exists some $\langle t, i\rangle \in \mathcal{I}$, such that $\langle t', i'\rangle \in \mathcal{F}(t,i)$. That is, the image set of $\mathcal{I}$ under $\mathcal{F}$ is $\mathcal{I}'$.*
- ***Increasing mapping:*** *For any $\langle t_1, i_1\rangle, \langle t_2, i_2\rangle \in \mathcal{I}$, $\langle t_1, i_1\rangle < \langle t_2, i_2\rangle$ implies that either $\mathcal{F}(t_1,i_1) = \mathcal{F}(t_2,i_2)$ and $\mathcal{F}(t_1,i_1)$ is singular or $\mathcal{F}(t_1,i_1) < \mathcal{F}(t_2,i_2)$ (i.e. for any $\langle t, i\rangle \in \mathcal{F}(t_1,i_1)$ and $\langle t', i'\rangle \in \mathcal{F}(t_2,i_2)$, $\langle t, i\rangle < \langle t', i'\rangle$).*
- ***ϵ-boundedness:*** *For every $\langle t, i\rangle \in \mathcal{I}$ and every $\langle t', i'\rangle \in \mathcal{F}(t,i)$, $|\langle t', i'\rangle - \langle t, i\rangle| < \epsilon$.*

Lemma 5. *If $\mathcal{F}$ is an ϵ-neighboring function from $\mathcal{I}$ to $\mathcal{I}'$, then $\mathcal{F}^{-1}$ is an ϵ-neighboring function from $\mathcal{I}'$ to $\mathcal{I}$.*

Proof. It is not hard to prove by the definition of ϵ-neighboring function.

Theorem 3. *Let $\mathcal{I}$ and $\mathcal{I}'$ be two labelled time intervals in $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}}'$ respectively. If $max\{\,|l(\mathcal{I}) - l(\mathcal{I}')|,\ |r(\mathcal{I}) - r(\mathcal{I}')|\,\} < \epsilon$, then there is an ϵ-neighboring function from $\mathcal{I}$ to $\mathcal{I}'$.*

Proof. There are three different cases according to the form of $\mathcal{I}$ and $\mathcal{I}'$.
Case 1: One of the intervals of $\mathcal{I}$ and $\mathcal{I}'$ is a singular interval. An ϵ-function can be established by letting the other interval be the source (or destination) of the element of the singular interval. It is not hard to check that such a multi-valued function is an ϵ-function.
Case 2: $|\mathcal{I}| > 0$, $|\mathcal{I}'| > 0$ and $\mathcal{I}$ and $\mathcal{I}'$ are of the same form of labelled time interval (i.e. both of them are either right-open or right-closed). Let I and I' be the unlabelled intervals of $\mathcal{I}$ and $\mathcal{I}'$ respectively. We can construct a function $F : I \to I'$ as follows:

$$F(t) = (t - l(I))\frac{|I'|}{|I|} + l(I') \quad t \in I$$

In case that both $|I|$ and $|I'|$ are infinite, we define $\frac{|I|}{|I'|} = 1$.

By Remark 1, we know there are bijections G_I (from $\mathcal{I}$ to I) and $G_{I'}$ (from $\mathcal{I}'$ to I'). Construct a function F as follows:

$$\mathcal{F}(G_I^{-1}(t)) = G_{I'}^{-1}(F(t)) \quad t \in I$$

It is easy to prove that $\mathcal{F}$ is a function from $\mathcal{I}$ to $\mathcal{I}'$, and furthermore, it is an ϵ-function from $\mathcal{I}$ to $\mathcal{I}'$.

[6] A multi-valued function is a total relation; i.e. every input is associated with one or more outputs.

Case 3: $|\mathcal{I}| > 0$, $|\mathcal{I}'| > 0$ and $\mathcal{I}$ and $\mathcal{I}'$ are of different forms of time interval (i.e. one is right-open and the other is right-closed). Again, let I and I' be the unlabelled intervals of $\mathcal{I}$ and $\mathcal{I}'$ respectively.

First, suppose I is right-closed and I' is right-open (see Fig. 3-a). Let $|r(\mathcal{I}') - r(\mathcal{I})| = |r(I') - r(I)| = \epsilon'$, where $\epsilon' < \epsilon$. Now we can divide I and I' into two time intervals respectively as shown in Fig. 3-b. $I = I_1 \cup I_2$, where $I_1 = [l(I), r(I))$ and $I_2 = [r(I), r(I)]$. $I' = I'_1 \cup I'_2$, where $I'_1 = [l(I'), d)$, $I'_2 = [d, r(I'))$ and $d = max\{r(I') - \frac{\epsilon - \epsilon'}{3}, r(I') - \frac{|I'|}{3}\}$. It is not hard to prove that $|r(I_1) - r(I'_1)| < \epsilon$ and $|l(I_2) - l(I'_2)| < \epsilon$.

Correspondingly, we can divide $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$ ($\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$) and $\mathcal{I}' = \mathcal{I}'_1 \cup \mathcal{I}'_2$ ($\mathcal{I}'_1 \cap \mathcal{I}'_2 = \emptyset$), where I_1, I_2, I'_1 and I'_2 are unlabelled intervals of $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{I}'_1$ and $\mathcal{I}'_2$ respectively. It is easy to see that $max\{\,|l(\mathcal{I})_1 - l(\mathcal{I}'_1)|,\, |r(\mathcal{I})_1 - r(\mathcal{I}'_1)|\,\} < \epsilon$ and $|l(\mathcal{I})_2 - l(\mathcal{I}'_2)|,\, |r(\mathcal{I})_2 - r(\mathcal{I}'_2)|\,\} < \epsilon$.

According to the results of Case 1 and Case 2, there exists an ϵ-function $\mathcal{F}_1$ from $\mathcal{I}_1$ to $\mathcal{I}'_1$ and an ϵ-function $\mathcal{F}_2$ from $\mathcal{I}_2$ to $\mathcal{I}'_2$. Construct a function $\mathcal{F} : \mathcal{I} \to \mathcal{I}'$ as follows:

$$\mathcal{F}(t,i) = \begin{cases} \mathcal{F}_1(t,i), & \langle t,i\rangle \in \mathcal{I}_1 \\ \mathcal{F}_2(t,i), & \langle t,i\rangle \in \mathcal{I}_2 \end{cases} \tag{2}$$

It is not hard to check that $\mathcal{F}$ satisfies three properties of ϵ-neighboring functions.

In the case that I' is right-closed and I is right-open, we know there is an ϵ-function $\mathcal{F}'$ from $\mathcal{I}'$ to $\mathcal{I}$. By Lemma 5, $\mathcal{F}'^{-1}$ is an ϵ-function from $\mathcal{I}$ to $\mathcal{I}'$.

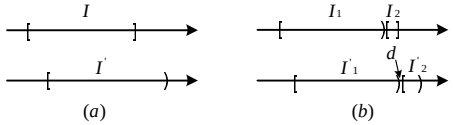


Fig. 3. Mappings over two time intervals

Corollary 1. *Let $\mathcal{I}$ and $\mathcal{I}'$ be labelled time intervals in $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}'}$ respectively. $max\{\,|l(\mathcal{I}) - l(\mathcal{I}')|,\, |r(\mathcal{I}) - r(\mathcal{I}')|\,\} < \epsilon$ implies that there is an ϵ-neighboring function $\mathcal{F}$ from $\mathcal{I}$ to $\mathcal{I}'$, such that for any $\langle t,i\rangle \in \mathcal{I}$, $min\{\mathcal{F}(t,i)\}$ exists. Furthermore, for any $\langle t',i'\rangle \in \mathcal{I}'$, $min\{\mathcal{F}^{-1}(t',i')\}$ exists.*

Proof. According to the form of $\mathcal{I}$ and $\mathcal{I}'$, it is easy to prove the Corollary by constructing an ϵ-neighboring function $\mathcal{F}$ as in the proof of Theorem 3.

4.3 ϵ-neighboring function $\mathcal{F}$ over two labelled time interval sequences

Now we define the ϵ-neighboring function $\mathcal{F}$ on two labelled time interval sequences.

Definition 9. *Let $\epsilon \in \mathbb{R}^{\geq 0}$. Further let $\overline{\mathcal{I}}$ and $\overline{\mathcal{I}}'$ be two labelled time interval sequences. $\mathcal{F}$ is a multi-valued function from $\overline{\mathcal{I}}$ to $\overline{\mathcal{I}}'$ and $\mathcal{F}(t,i)$ represents the set of images of time $\langle t,i\rangle$ ($\langle t,i\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$). $\mathcal{F}$ is called an ϵ-neighboring function from $\overline{\mathcal{I}}$ to $\overline{\mathcal{I}}'$, iff the following properties are satisfied.*

– ***Interval consistency:*** *For every* $\langle t,i\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$, $\langle t,i\rangle \in \mathcal{I}_k$ *implies that* $F(t,i) \subseteq I'_k$ *(where* $k < N(\overline{\mathcal{I}})$*).*
– ***Surjection:*** *For any* $(t',i') \in \mathbb{T}^{\overline{\mathcal{I}}'}$*, there exists some* $(t,i) \in \mathbb{T}^{\overline{\mathcal{I}}}$*, such that* $(t',i') \in \mathcal{F}(t,i)$*. That is, the image set of* $\mathbb{T}^{\overline{\mathcal{I}}}$ *under* F *is* $\mathbb{T}^{\overline{\mathcal{I}}'}$
– ***Increasing mapping:*** *For any* $(t_1,i_1),(t_2,i_2) \in \mathbb{T}^{\overline{\mathcal{I}}}$,$(t_1,i_1) < (t_2,i_2)$ *implies that either* $F_{\mathcal{I}}(t_1,i_1) = F_{\mathcal{I}}(t_2,i_2)$ *and* $F_{\mathcal{I}}(t_1,i_1)$ *is singular or* $F_{\mathcal{I}}(t_1,i_1) < F_{\mathcal{I}}(t_2,i_2)$ *(i.e. for any* $(t,i) \in F_{\mathcal{I}}(t_1,i_1)$ *and* $(t',i') \in F(t_2,i_2), (t,i) < (t',i')$*).*
– ϵ***-boundedness:*** *For every* $t \in \mathbb{T}^{\overline{\mathcal{I}}}$ *and every* $(t',i') \in \mathcal{F}(t',i')$, $|(t',i')-(t,i)| < \epsilon$.

Lemma 6. *If* $\mathcal{F}$ *is an* ϵ*-neighboring function from* $\overline{\mathcal{I}}$ *to* $\overline{\mathcal{I}}'$*, then* $l(\mathcal{I}'_i) \in \mathcal{F}(l(\mathcal{I}_i))$*, for all* $i \in \mathbb{N}$ *(*$i < N(\overline{\mathcal{I}})$*).*

Proof. Since any form of labelled time intervals is left-closed, it is easy to see that $l(\mathcal{I}'_i) \in \mathcal{I}'_i$ and $l(\mathcal{I}_i) \in \mathcal{I}_i$. Suppose $l(\mathcal{I}'_i) \notin \mathcal{F}(l(\mathcal{I}_i))$. By the surjection property and the interval consistency property of $\mathcal{F}$, there is some $\langle t,k\rangle \in \mathcal{I}$ and $l(\mathcal{I}'_i) \in \mathcal{F}(t,k)$. It is easy to see that $\langle t,k\rangle > l(\mathcal{I}'_i)$. Furthermore, by interval consistency property of $\mathcal{F}$, for any $\langle t',k'\rangle \in \mathcal{F}(l(\mathcal{I}_i))$, $\langle t',k'\rangle \in \mathcal{I}'_i$. It is easy to see that $\langle t',k'\rangle > l(\mathcal{I}'_i)$, which contradicts the increasing mapping property of $\mathcal{F}$.

Theorem 4. *Let* $\overline{\mathcal{I}}$ *and* $\overline{\mathcal{I}}'$ *be two labelled time interval sequences.* $d_{int}(\overline{\mathcal{I}},\overline{\mathcal{I}}') < \epsilon$ *implies that there is an* ϵ*-neighboring function from* $\overline{\mathcal{I}}$ *to* $\overline{\mathcal{I}}'$*.*

Proof. For any $i \in \mathbb{N}$ ($i < N(\overline{\mathcal{I}})$), by Lemma 4 and Theorem 3, there is an ϵ-function $F_{\mathcal{I}_k}$ from $\mathcal{I}_k$ to $\mathcal{I}'_k$. Now construct function $\mathcal{F}$ as follows:

$$\mathcal{F}(i,t) = F_{\mathcal{I}_k}(i,t) \;\; if \;\; \langle i,t\rangle \in \mathcal{I}_k, \;\; for\ all\ k < N(\overline{\mathcal{I}})$$

It is easy to prove that $\mathcal{F}$ is a multi-valued function from $\overline{\mathcal{I}}$ to $\overline{\mathcal{I}}'$ and it satisfies the properties of the ϵ-neighboring function in Definition 9.

Corollary 2. *Let* $\overline{\mathcal{I}}$ *and* $\overline{\mathcal{I}'}$ *be two labelled time interval sequences.* $d_{int}(\overline{\mathcal{I}},\overline{\mathcal{I}}') < \epsilon$ *implies that there is an* ϵ*-neighboring function* $\mathcal{F}$ *from* $\overline{\mathcal{I}}$ *to* $\overline{\mathcal{I}'}$*, such that for any* $\langle t,i\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$, $min\{\mathcal{F}(t,i)\}$ *exists. Furthermore, for any* $\langle t',i'\rangle \in \mathbb{T}^{\overline{\mathcal{I}}'}$, $min\{F^{-1}_{\overline{\mathcal{I}}}(t',i')\}$ *exists.*

Proof. It follows from Corollary 1, the interval consistency property of $\mathcal{F}$ and Lemma 5.

This corollary is important for the later proof of the preservation of "Until" properties between two timed state sequences. The well-known "Until" problem in [6] states that given two adjacent intervals where φ_1 and φ_2 are hold respectively, the semantics of $\varphi_1 \mathsf{U} \varphi_2$ in general is sensitive to whether the second interval is left-closed or not. Therefore the time intervals in a timed state sequence only accept left-closed intervals in this paper. Furthermore, the corollary states that there exists an ϵ-neighboring function, which can ensure the image set of each point is left-closed during the mapping.

5 Real-time property preservation

In common practice, properties of a system are checked by verification techniques such as model-checking, theorem proving and simulation. In this section, we are going to present a way to predict the real-time properties of a real-time system based on the real-time properties of another real-time system, which can be used to predict the properties of the implementation based on the properties of the model (see Section 6 and 7).

5.1 Real-time property preservation between timed state sequences

In the previous sections, we used $MITL_{\mathbb{R}}$ formulas to formally express real-time properties of a real-time system and defined ϵ-weakening functions over them. At the same time, the timing behavior of the system is formally expressed by a set of timed state sequences, on which ϵ-neighboring functions are defined [7]. Now we are going to show that ϵ-neighboring timed state sequences satisfy "almost the same" properties, which is formally proven in Lemma 7 and Theorem 5.

Lemma 7. *Let $\epsilon \in \mathbb{R}^{\geq 0}$, $\langle t, i\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$ and $\varphi \in MITL_{\mathbb{R}}$. Further let $\overline{\tau}$ and $\overline{\tau}'$ be two ϵ-neighboring timed state sequences and let $\mathcal{F}$ be an ϵ-neighboring function from the time interval sequence of $\overline{\tau}$ to that of $\overline{\tau}'$. Then $(\overline{\tau}, \langle t, i\rangle) \models \varphi$ implies that for any $\langle t', i'\rangle \in \mathcal{F}(t, i)$, $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi)$.*

Proof. We show that $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi)$ by induction on the structure of formula φ.
Case 1: $\varphi = p$. By the definition of function $R^{2\epsilon}$, $R^{2\epsilon}(\varphi) = p$. By the interpretation of $MITL_{\mathbb{R}}$ formulas over timed state sequences, $p \in \overline{\tau}(t, i)$. Since $\mathcal{F}$ is an ϵ-neighboring function from the labelled time interval sequence of $\overline{\tau}$ to that of $\overline{\tau}'$, and $\overline{\tau}$ and $\overline{\tau}'$ share the same state sequence, we know that $\overline{\tau}(t, i) = \overline{\tau}'(t', i')$ by the interval consistency property of $\mathcal{F}$. Hence, $p \in \overline{\tau}'(t', i')$. But then, $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi)$.
Case 2: $\varphi = \neg p$. The proof is identical to the previous case.
Case 3: $\varphi = \varphi_1 \vee \varphi_2$. $(\overline{\tau}, \langle t, i\rangle) \models \varphi_1$ or $(\overline{\tau}, \langle t, i\rangle) \models \varphi_2$. By induction we have $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1)$ or $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_2)$. But then $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1) \vee R^{2\epsilon}(\varphi_2) = R^{2\epsilon}(\varphi)$.
Case 4: $\varphi = \varphi_1 \wedge \varphi_2$. The proof is similar to the previous case.
Case 5: $\varphi = \varphi_1 \mathsf{U}_I \varphi_2$. There is some $t_2 \in I$ and j ($1 \leq j \leq m^{\overline{\tau}}_{t+t_2}$) and $\langle t, i\rangle \leq \langle t + t_2, j\rangle$, such that $(\overline{\tau}, \langle t + t_2, j\rangle) \models \varphi_2$ and for all t_1 and k that satisfy $\langle t, i\rangle \leq \langle t_1, k\rangle < \langle t + t_2, j\rangle$, $(\overline{\tau}, \langle t_1, k\rangle) \models \varphi_1$. By induction we have that $(\overline{\tau}', \langle t'_2, j'\rangle) \models R^{2\epsilon}(\varphi_2)$ for any $\langle t'_2, j'\rangle \in \mathcal{F}(t + t_2, j)$.

In case that $\langle t, i\rangle = \langle t + t_2, j\rangle$, the proof is straightforward.

In case that $\langle t, i\rangle < \langle t + t_2, j\rangle$, there are two possible relations between $\mathcal{F}(t + t_2, j)$ and $\mathcal{F}(t, i)$ according to the increasing mapping property of $\mathcal{F}$.

- $\mathcal{F}(t+t_2, j) = \mathcal{F}(t, i)$ and $\mathcal{F}(t, i)$ is singular. Let $\mathcal{F}(t+t_2, j) = \mathcal{F}(t, i) = \{\langle t', i'\rangle\}$. By induction we have $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1)$. Now we will show that $0 \in I \oplus 2\epsilon$. By the ϵ-boundedness property of $\mathcal{F}$, $|\langle t', i'\rangle - \langle t, i\rangle| = |t' - t| < \epsilon$ and $|\langle t', i'\rangle - \langle t + t_2, j\rangle| = |t' - t - t_2| < \epsilon$. By the Triangle Inequality,
$$0 = |\langle t', i'\rangle - \langle t', i'\rangle| = |t' - t'| \geq t_2 - |t' - (t + t_2)| - |t' - t)| \geq t_2 - 2\epsilon$$
Since $t_2 \in I$, it is not hard to check that $0 \in I \oplus 2\epsilon$. Hence, $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1) \mathsf{U}_{I \oplus 2\epsilon} R^{2\epsilon}(\varphi_2) = R^{2\epsilon}(\varphi)$.
- $\mathcal{F}(t, i) < \mathcal{F}(t + t_2, j)$. By Corollary 2 we know that $min\{\mathcal{F}(t + t_2, j)\}$ exists. Let $\langle t^*_2, j^*\rangle = min\{\mathcal{F}(t + t_2, j)\}$. It is easy to see that $\langle t', i'\rangle < \langle t^*_2, j^*\rangle$. By induction we have that $(\overline{\tau}', \langle t^*_2, j^*\rangle) \models R^{2\epsilon}(\varphi_2)$. For any t'_1 and k' ($1 \leq k \leq m^{\overline{\tau}'}_{t'_1}$) that satisfy $\langle t', i'\rangle \leq \langle t'_1, k'\rangle < \langle t^*_2, j^*\rangle$, by the surjection property and the increasing

[7] We call two timed state sequences ϵ-neighboring if they are $\equiv_s$-equivalent and the distance between their time interval sequences is less than ϵ.

mapping property of $\mathcal{F}$, it is not hard to prove there is some $\langle t_1'', k''\rangle \in \mathbb{T}^{\overline{\mathcal{I}}}$ such that $\langle t_1', k'\rangle \in \mathcal{F}(t_1'', k'')$ and $\langle t, i\rangle \leq \langle t_1'', k''\rangle < \langle t + t_2, j\rangle$. By induction we have that $(\overline{\tau}', \langle t_1', k'\rangle) \models R^{2\epsilon}(\varphi_1)$. Similar to the previous case, by the ϵ-boundedness property of $\mathcal{F}$ and the Triangle Inequality, it is not hard to prove that $|t_2^* - t| \in I \oplus 2\epsilon$. Hence, $(\overline{\tau}', \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1) \mathsf{U}_{I\oplus 2\epsilon} R^{2\epsilon}(\varphi_2) = R^{2\epsilon}(\varphi)$.

Case 6: $\varphi = \varphi_1 \mathsf{V}_I \varphi_2$. The proof is similar to the previous case. A brief proof is given as follows.
For all $t_2' \in I \ominus 2\epsilon$, $1 \leq j' \leq m^{\overline{\tau}'}_{t'+t_2'}$ and $\langle t', i'\rangle \leq \langle t' + t_2', j'\rangle$, we can prove that there exist $t_2^* \in I$ and $1 \leq j^* \leq m^{\overline{\tau}}_{t+t_2}$, such that $\langle t + t_2^*, j^*\rangle = max\{min\{\mathcal{F}^{-1}(t' + t_2', j')\}, \langle t, i\rangle\}$. There are two possibilities:

- $(\overline{\tau}, \langle t + t_2^*, j\rangle) \models \varphi_2$. By induction we have $(\overline{\tau}', \langle t' + t_2', j'\rangle) \models R^{2\epsilon}(\varphi_2)$.
- There is some t_1 and k ($1 \leq k \leq m^{\overline{\tau}}_{t_1}$) that satisfy $\langle t, i\rangle \leq \langle t_1, k\rangle < \langle t + t_2^*, j\rangle$, such that $(\overline{\tau}, \langle t + t_1, k\rangle) \models \varphi_1$. It is not hard to prove that there exists $\langle t_1^*, k^*\rangle = max\{min\{\mathcal{F}(t + t_1, k)\}, \langle t', i'\rangle\}$ and $\langle t', i'\rangle \leq \langle t_1^*, k^*\rangle < \langle t' + t_2', j'\rangle$, where $1 \leq k^* \leq m^{\overline{\tau}'}_{t_1^*}$. By induction we have $(\overline{\tau}', \langle t_1^*, k^*\rangle) \models R^{2\epsilon}(\varphi_1)$.

Hence, $(\overline{\tau}, \langle t', i'\rangle) \models R^{2\epsilon}(\varphi_1) \mathsf{V}_{I\ominus 2\epsilon} R^{2\epsilon}(\varphi_2) = R^{2\epsilon}(\varphi)$.
This completes our inductive proof of Lemma 7.

Theorem 5. *Let $\epsilon \in \mathbb{R}^{\geq 0}$ and $\varphi \in MITL_{\mathbb{R}}$. Further let $\overline{\tau}$ and $\overline{\tau}'$ be two ϵ-neighbouring timed state sequences, then $\overline{\tau} \models \varphi$ implies $\overline{\tau}' \models R^{2\epsilon}(\varphi)$.*

Proof. It is not hard to prove by Lemma 7.

5.2 Real-time property preservation between timed systems

In the previous section, we examined the real-time property preservation between timed state sequences. However, in practice, we often need to analyze real-time properties of a timed system instead of those of a single timed state sequence. Since a timed system consists of a set of timed state sequences, the problem of examining the satisfaction of a formula φ in a timed system S is equivalent to examining its satisfaction in all of the timed state sequences in S. Real-time property preservation between timed state sequences can also be extended to analyze real-time properties between timed systems. The following theorems give these extensions.

Theorem 6. *Let S be a set of timed state sequences and let φ be an $MITL_{\mathbb{R}}$ formula. For any $\epsilon \in \mathbb{R}^{\geq 0}$, if $S \models \varphi$, then $S \models R^{\epsilon}(\varphi)$.*

Proof. Follows from Definition 2 and Theorem 2.

Theorem 7. *Let S_1, S_2 be two sets of timed state sequences and let φ be an $MITL_{\mathbb{R}}$ formula. Assume that there is some $\epsilon \in \mathbb{R}^{\geq 0}$, such that for any timed state sequence $\overline{\tau}$ in S_2, there exists a sequence $\overline{\tau'}$ in S_1, such that $\overline{\tau}$ and $\overline{\tau'}$ are ϵ-neighboring. Then $S_1 \models \varphi$ implies $S_2 \models R^{2\epsilon}(\varphi)$.*

Proof. The theorem follows from Definition 2 and Theorem 5.

6 An application

The property-preservation between concurrent real-time systems serves as the basis to bridge the gap between a model and its implementation. By establishing a proper linkage between the semantics of design and implementation languages, it is possible to generate automatically a correctness-preserving implementation from its model. In this way, a property verified in the model can still be guaranteed in the implementation.

In the remainder, we first examine the gap between the semantics of design languages and implementation languages, which often lead to inconsistency [12] in the real-time system development. Afterwards, a way to bridge the gap is proposed based on the property-preservation results we have proven.

6.1 Gap between the semantics of design and implementation languages

A commonly adopted timing semantics in design languages treats system progress and time progress in an orthogonal way, namely, system actions are timeless (without any time progress) and the time progress is actionless (without preforming any action). As a consequence, this timing semantics is associated with a virtual time (a system variable) instead of the physical time. Since this semantics simplifies the reasoning of real-time systems, variations of the semantics have been used in different formal frameworks to model and analyze real-time systems, such as timed automata[3], time process algebra ATP[15] and real-time transition systems[10]. Recently, the semantics is also integrated into different design languages, such as SDL-2000 [19] supported by TAU G2 [17] and POOSL supported by SHESim [16][18].

The integration of this semantics into a design language for real-time systems offers many benefits [12]. To mention just a few: 1) each component of the system is not affected by the non-deterministic factors of a platform, 2) the behavior of each component does not change during the integration and 3) debugging and analysis code does not change the timing behavior of the system. Furthermore, verification techniques (such as model-checking and simulation) based on this semantics have been developed, which can assist designers to check the correctness of a design model. However, devising a correct design solution within such a semantic framework does not guarantee a successful implementation, due to the gap between two semantic domains of design languages and implementation languages.

The semantics of an implementation language is usually related with and constrained by the underlying platform. The difficulty of maintaining correctness between the design model and its implementation is attributed to several reasons. First, certain assumptions are often taken on the semantics of design languages in order to efficiently explore the design space, such as instantaneous actions for timing behaviors. These assumptions are valid at a certain level of abstraction, but they do not directly fit to the semantics of implementation languages. For example, every action does take a certain amount of execution time in all implementation languages. Second, some primitives and operations in design languages do not have direct correspondence in implementation languages. For example, during system generation, the default concurrency operation is often represented by a specific thread mechanism offered by the underlying operating system, whose semantics is not always consistent with that in design languages.

To bridge the semantic gap between design languages and implementation languages, a formal linkage between two semantic domains (design and implementation) is built based on the ϵ-hypothesis, which requires that:

1. The observable action sequences of the implementation should be the same as one of those in the model.
2. The time deviation between activations of the corresponding actions in the implementation and the model should be less than or equal to ϵ seconds.

If the ϵ-hypothesis is complied during the transformation from the model to the implementation, we can predict properties of the implementation from those of the model based on the property-preservation between two timed state sequences. For example, if an upper bound of the time deviation between the implementation and the model is 0.01 second and $\Box(p \rightarrow \Diamond_{[3,5]} q)$ is satisfied in the model, we can conclude that property $\Box(p \rightarrow \Diamond_{[2.98,5.02]} q)$ holds in the implementation. However, the ϵ-hypothesis might be ineffective when expensive data computations are involved. Due to the possible large values of ϵ, quantitative real-time properties of the implementation could be "far" away from the properties of the model. Further research is needed to investigate this issue.

6.2 A predictable development approach

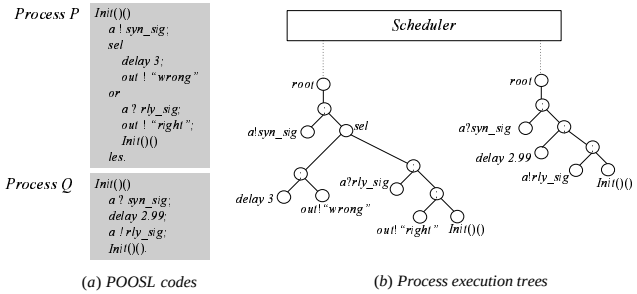
In this section, we introduce a predictable development approach. It provides an adequate design language (POOSL) for modelling concurrent real-time systems and a tool (Rotalumis) to generate an implementation from a POOSL model by trying to comply with the ϵ-hypothesis.

POOSL: POOSL (Parallel Object-Oriented Specification Language) integrates a process part based on a timing and probability extension of CCS and a data part based on a traditional object-oriented language [18]. The semantics of POOSL is based on a two-phase execution model, where the state of a system can change either by asynchronously executing some atomic actions, such as communication and data computation without time passing (phase 1) or by letting time pass synchronously without any action being performed (phase 2).

The implementation of the two-phase execution model in tool SHESim is achieved by adopting the so-called process execution tree (PET). The state of each process is represented by a tree structure, where each leaf is a statement or a recursively defined process method. For example, Fig. 4 shows a system, in which two processes P and Q communicate with each other (more details are given in Section 7.1). The PETs of $P \parallel Q$ is shown in Fig. 4b. During the evolution of the system, each PET provides its candidate actions to the PET scheduler and dynamically adjusts its state according to the choice made by the PET scheduler. More details about PET can be found in [5]. The correctness of PET with respect to the semantics of the POOSL language is formally proven in [8].

Rotalumis: Rotalumis takes the POOSL model during the design stage as its input and automatically generates the executable code for the target platform. The ϵ-hypothesis is incorporated into Rotalumis by applying the following techniques:

Process execution tree: POOSL language provides ample facilities to describe system characteristics such as parallelism, nondeterministic choice, delay and communication that are not directly supported by C++ or other implementation languages. In order



(*a*) *POOSL codes* (*b*) *Process execution trees*

Fig. 4. The $P \parallel Q$ system in POOSL

to provide a correct and smooth mapping from a POOSL model to a C++ implementation, each PET in the model is directly transferred into a PET in C++ structure, whose behavior is the same as the PET implemented in SHESim. As a result, the generated implementation exhibits exactly the same behavior as that in the model, if interpreted in the virtual time domain. On the other hand, the implementation of a system needs to interact with the outside world and its behavior has to be interpreted in the physical time domain. Since the progress of the virtual time is monotonic increasing, which is consistent with the progress of the physical time, the event order observed in the virtual time domain is consistent with that in the physical time domain. That is, the scheduler of PETs ensures that the implementation always has the same event order as observed in the POOSL model.

Synchronization between virtual time and physical time: To obtain the same (or similar) quantitative timing behavior in the physical time domain as in the model, the scheduler of PETs tries to synchronize the virtual time and the physical time during the running of the implementation, which ensures that the execution of the implementation is always as close as possible to a trace in the model with regard to the distance between timed state sequences. Due to the physical limitation of the platform, the scheduler may fail to guarantee that the implementation is ϵ-close to the model, for a fixed ϵ value. In this case, designers can get the information about the missed actions from the scheduler. Correspondingly, they can either change the model to reduce the computation cost of a certain virtual time moment [8], or replace the target platform with a platform of better performance.

7 Case studies

7.1 Two parallel processes P and Q

Before giving a case study of a railroad crossing system, we first demonstrate that a automatic generation tool, such as TAU G2, is prone to generate a "wrong" implementation, if the semantic gap between design languages and implementation languages has not been considered carefully. This is shown by a case study of two parallel processes P and Q. Their behaviors are shown in Fig. 5, and more details can be found in [12]. We made models in both TAU G2 and SHESim, where P process always outputs message "correct". However, the implementation generated automatically from models by TAU G2 exhibits an unexpected behavior as shown in Fig. 6, which is not the case in the implementation generated from Rotalumis.

[8] The time deviation between the model and the implementation is mainly due to the computation cost of actions occurring at a certain virtual time moment (as shown in Fig. 12a).

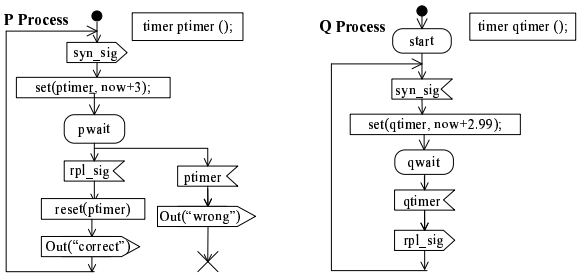


Fig. 5. Two parallel process P And Q

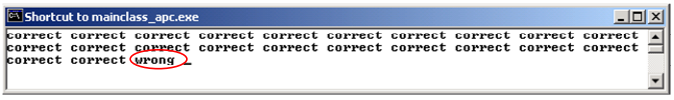


Fig. 6. An implementation of $P \parallel Q$

7.2 A railroad crossing system

In this section, we are going to present a railroad crossing system. As shown in Fig. 7, four stations are connected by two orthogonal tracks. Two trains (a and b) run back and forth between station 1 and 2 and station 3 and 4 respectively. Four sensors (A, B, C and D) are installed at some distance to the crossing to detect the passing of the trains. To compensate for the deceleration time of the train and avoid the stopping of the train inside the crossing area, a critical zone is defined as shown in Fig. 7. When a train approaches a border of the critical zone from outside of the zone, it has to request for permission to enter the crossing. If the request is denied, the train has to stop immediately and wait until the permission is granted. When the train leaves the crossing, it has to release the crossing. As a consequence, the crossing is free and available for the other train. We assume that the speed of the train is constant between two sensor points (e.g. from point A to B) if no stop occurs.

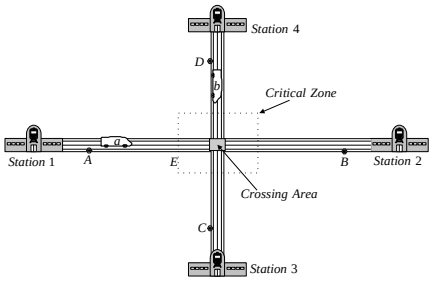


Fig. 7. The railroad crossing system

One important requirement is that the trains should never collide and the system should operate as efficient as possible. Due to the limited space, in this paper, we only focus on a subtask in the whole system development, the synthesis of an implementation from an adequate model. For a more detailed description of the complete system development, interested readers are referred to [13].

7.3 System design

As shown in Fig. 8, the system consists of environmental parts (including physical trains, sensors and a LEGO® Dacta controller) and control software parts running on a computer. The Dacta controller provides low level device drivers to control the LEGO® motors and sensors. The designed control software interacts with the physical trains and sensors through the Dacta controller. The physical connection between the PC and the Dacta controller is the standard RS-232.

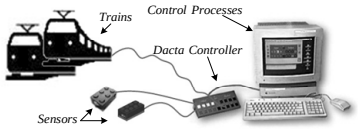


Fig. 8. The architecture of the railroad crossing system

Fig. 9 illustrates an analysis model of the whole system. The control part of the system (in the rectangle with dotted lines) consists of three parallel processes: $Crossing$, $TrainA\_Image$ and $TrainB\_Image$. The environment parts are modelled by processes $TrainA\_Actor$ and $TrainB\_Actor$. The code shown in Fig. 9 describes a part of the behavior of process $TrainB\_Image$, starting from the moment it receives a message ($sensor$) till it releases the crossing. T_1, T_2 and T_3 in the code are parameters determined by the speed of the train. In the case of $TrainB\_Image$, T_1 is 0.910 seconds. To avoid the collision of the trains and still keep system efficiency, we want

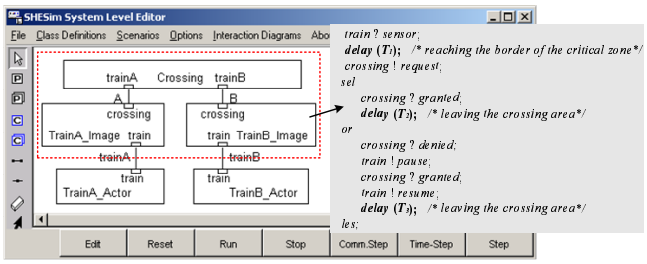


Fig. 9. The analysis model of the system

the control part of the system to satisfy the following quantitative real-time property. $TrainB\_Image$ should send a $pause$ message to the physical train between 0.900 and 0.920 seconds after it receives the first sensor message on each journey of the train, if its request for entering the crossing is denied. If the $pause$ message is sent earlier than 0.900 seconds, the train may stop unnecessarily and the system may not be efficient enough. On the other hand, if the $pause$ command is sent later than 0.920 seconds, the train may enter the crossing area incorrectly. This quantitative real-time property P can be formalized by using an $MITL_{\mathbb{R}}$ formula $\Box(r \rightarrow (p \rightarrow \Diamond_{[0.900,0.920]}q))$, where p, q and r represent the following atomic propositions: 1) $TrainA\_Image$ receives the first $sensor$ message on each journey, 2) $TrainA\_Image$ sends a $pause$ message to the physical train and 3) $TrainA\_Image$ receives a $denied$ message from $Crossing$. Since the model is relatively simple, we could manually check that the control part does

satisfy property P_1: $\Box(r \rightarrow (p \rightarrow \Diamond_{[T_1,T_1]}q))$, where T_1 is 0.910. It is also possible to verify property P_1 by simulation or model-checking techniques.

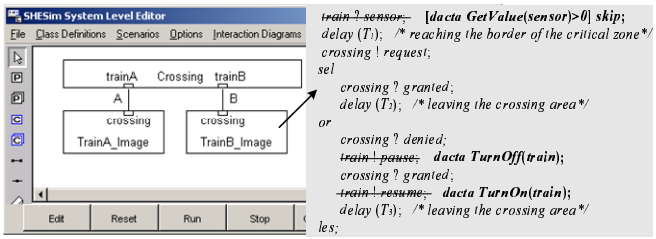


Fig. 10. The synthesis model of the system

To facilitate automatic system synthesis, the environmental parts have to be excluded from the rest of the model. In addition, the synchronization between the control parts and the environmental parts need to be replaced with a synthesizable interface. After these refinements, we obtain a synthesis model as shown in Fig. 10. The code given in Fig. 10 shows that the synchronization of $TrainB\_Image$ are replaced by virtual interfaces (primitive data methods of Dacta class). These interfaces have been implemented in Rotalumis and can be transferred automatically into the real interactions with the outside world during the system synthesis.

7.4 System synthesis

In this phase, the synthesis model devised during the design stage is automatically transformed into an executable implementation. In order to generate a correct and effective implementation, two important tasks should be carried out.

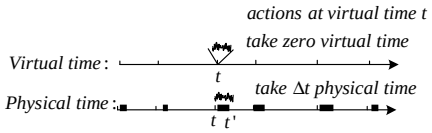
```
static PDO *PDM_TurnOff(PDO **LV)
{
    if (LV[1]->Class != PDC_Char)
        DisplayError("The 1st Operand of dacta.TurnOff should be a Char.");
    port=(unsigned char)LV[1]->i;
    outp(SerialPort, 0x21);   /*Serial port communication to turn off a motor*/
    outp(SerialPort, port);
    return LV[0];
}
```

Fig. 11. The implementation of the interface $Dacta\ \ TurnOff\,(train)$

Implementation of the interface data: Interface data class $Dacta$ defined in the synthesis model only provides a virtual interface for control processes. The actual implementation of the interface data for the communication with the Dacta controller is provided by the Rotalumis tool. For example, the interface for stopping a train ($Dacta\ \ TurnOff\,(train)$) can be implemented by using a segment of C++ codes as shown in Fig. 11. The interface implementation should be non-blockable because its timing behavior can affect the deviation between the model and its implementation. In our case, the computation cost of the interface codes is less than 10^{-5} seconds.

Measuring the upper bound of timing deviations: Since the upper bound of timing deviations ϵ between the model and the implementation has direct influence on the quantitative real-time properties of the implementation, we need to estimate it properly. In practice, we can obtain the ϵ value in various ways: 1) Developers can run the implementation and let the scheduler record the timing deviation between the virtual time

and the physical time (such as shown in Fig. 12a), which is caused by the execution of all actions occurring at a certain virtual time moment, then the ϵ value can be estimated according to recorded values. 2) Developers can model the platform and the time duration of each action, and then estimate the ϵ value in the integrated model of the platform and the system. More details about the estimation of the ϵ value can be found in [7].



(*a*) *Time deviations* (*such as t'-t*)

Time deviation (10^{-5} second)	1 ~ 5	5~ 10	10~ 30
Frequency	298079	1925	24

(*b*) *Statistics of time deviations*

Fig. 12. Measuring ϵ (the upper bound of timing deviations)

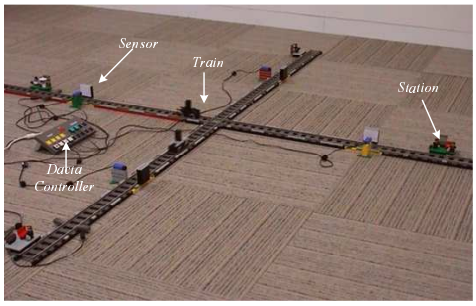


Fig. 13. The controlled physical system

In our example, we estimate the ϵ value using the first way, due to the fact that it is difficult to model the underlying platform (Windows 98) adequately. We recorded around 300000 time deviations between the virtual time and the physical time during a 44-hour continuous running. As shown in Fig. 12b, most timing errors fall between $1 \sim 5\times10^{-5}$ seconds, and only a very few time errors reach around 3×10^{-4} seconds, which are actually caused by the background activities of the operating system. Therefore, the value of ϵ can be safely estimated as 0.001 seconds in our example. Now we can predict that the implementation satisfies a 2ϵ-weakened property of P_1, namely $\Box(r \rightarrow (p \rightarrow \Diamond_{[0.908,0.912]}q))$, which is stronger than required property P. A correct implementation was generated automatically from the model. Fig. 13 gives a snapshot of the controlled physical system.

8 Conclusions

In this paper, we have investigated the property-preservation between two concurrent real-time systems. This result can serve as the basis to bridge the semantic gap between design languages and implementation languages and control the inconsistencies between a model and its implementation. To this end, the ϵ hypothesis is introduced, which assumes that the execution trace of the implementation is always ϵ-close to a trace in the model. A predictable development approach was developed based on the

ϵ-hypothesis. Case studies performed confirm that the proposed approach can generate a consistent implementation from the model within the ϵ deviation. The value of ϵ is usually dependent on various platforms and can be estimated by different techniques.

References

1. Alur, R., Henzinger, T,A.: Logics and Models of Real Time: A Survey, In Proc. of the Real-Time: Theory in Practice, REX Workshop, pp. 74 - 106, (1992).
2. Alur, R., Henzinger, T,A.: Real-time logics: complexity and expressiveness. Information and Computation, Vol. 104(1). pp. 35 - 77 (1993).
3. Alur, R., Dill, D.L.: A theory of timed automata, Theoretical Computer Science, Vol 126, pp. 183 - 235, (1994).
4. Baeten, J.C.M.: The total order assumption, In Proc. 1st North American Process Algebra Workshop, Stony Brook, pp. 231 - 240, (1993).
5. Bokhoven, L.J. van, Voeten, J.P.M., Geilen, M.C.W., Software Synthesis for System Level Design Using Process Execution Trees, In Proc. 25th Euromicro Conf. pp. 463 - 467, (1999).
6. Bouajjani, A. and Laknech, Y.: Temporal Logic + Timed Automata: Expressiveness and Decidability, In Proc. 6th CONCUR'95 Concurrency Theory, LNCS 962, pp. 531-546, (1995).
7. Florescu, O., Voeten, J.P.M., Huang, J., Corporaal, H.: Error Estimation in Model-Driven Development for Real-Time Software, Accepted by the FDL04, (2004).
8. Geilen, M.C.W.: Formal Techniques for Verification of Complex Real-time Systems, PhD thesis, Eindhoven Univ. of Tech. (2002)
9. Gupta, V., Henzinger, T.A., Jagadeesan, R.: Robust Timed Automata, Hybrid and Real-Time Systems, In Proc. of Int. Workshop HART'97, LNCS 1201, pp. 331 - 345, (1997).
10. Henzinger, T., Manna, Z., Pnueli, A.: An interleaving model for real time, In Proc. of the 5th Jerusalem conf. on Information tech. pp. 717 - 730, (1990).
11. Huang, J., Voeten, J.P.M., Geilen, M.C.W.: Real-time Property Preservation in Approximations of Timed Systems. In Proc. of 1st ACM & IEEE Int. Conf. on Formal Methods and Models for Codesign. pp. 163 - 172 (2003).
12. Huang, J., Voeten, J.P.M.,Bokhoven, L.J. van, Ventevogel, A.: Platform-independent design for embedded real-time systems, To appear in Languages for System Specification, Kluwer Academic Publishers, (2004).
13. Huang, J., Voeten, J.P.M., Putten, P.H.A. van der: Predictability in Real-time System Development (2) A Case Study, accepted by FDL04, (2004).
14. Nicollin, X.,Sifakis, J.:An overview and synthesis on timed process algebras, In Proc. of the 3rd workshop on Computer-Aided Verification, (1991).
15. Nicollin, X., Sifakis, J.: The algebra of timed processes ATP: theory and application, Information and Computation, Vol. 114(1), pp. 131-178, (1994).
16. Putten, P.H.A. van der,Voeten, J.P.M.: Specification of Reactive Hardware/Software Systems, PhD thesis, Eindhoven University of Technology, (1997).
17. TAU Generation 2, http://www.taug2.com (2003).
18. Voeten, J.P.M., Putten, P.H.A. van der, Geilen, M.C.W., Stevens, M.P.J.: System Level Modelling for Hardware/Software Systems, In Proc. of EUROMICRO'98, 154 - 161, (1998).
19. Z.100 Annex F1: Formal Description Techniques (FDT)–Specification and Description Language (SDL), Telecom. standardization sector of ITU, (2000).