A Timed Failure Equivalence Preserving Abstraction for Parametric Time-Interval Automata

Akio Nakata, Tadaaki Tanimoto, Suguru Sasaki, Teruo Higashino

Department of Information Networking,
Graduate School of Information Science and Technology,
Osaka University, Suita, Osaka 565-0871, Japan

Abstract

In the development of real-time communicating hardware/embedded-software systems, it is frequently the case that we want to refine/optimize the system's internal behavior while preserving the external timed I/O behavior. In such a design refinement, modification of the systems' internal branching structures, as well as re-scheduling of internal actions, may frequently occur. Our goal is, then, to ensure that such modification of internal branching structures and re-scheduling of internal actions preserve the systems' external timed behavior, which is typically formalized by the notion of (timed) failure equivalence since it is less sensitive to the difference of internal branching structures than (timed) weak bisimulation. In order to know the degree of freedom of such re-scheduling, parametric analysis is useful. One of the models suitable for such an analysis is a parametric time-interval automaton(PTIA), which is a subclass of the existing model, a parametric timed automaton. It has only a time interval with upper- and lower-bound parameters as a relative timing constraint between consecutive actions. In this paper, at first, we propose an abstraction algorithm of PTIA which preserves timed failure equivalence. Timed failure equivalence is strictly weaker than timed weak bisimulation in the sense that it does not distinguish the difference of the timing when the internal resolution of nondeterminism has occurred, but it does distinguish the difference of the refusals of communicating actions observed by an external environment. Then, we also show that after applying our algorithm, the reduced PTIA has no internal actions, and thus the problem deriving a parameter condition in order that given two models are timed failure equivalent can be reduced to the existing parametric strong bisimulation equivalence checking.
Keywords: real-time communicating systems, parametric timed automata, equivalence checking, timed failure equivalence, abstraction

1 Introduction

In recent years, an effective development methodology for embedded-hardware/software with real-time constraints is desired. Precise implementation of timing constraints for I/O behavior is becoming important not only in embedded systems like mobile phones but also in infrastructure systems for transportation, medicine, finance and defense. For such real-time systems, it is important to verify the equivalence of I/O timing

behavior between the initially designed specification and its refined implementation. In such a refinement process of the system development, it is frequently occur that the formally specified nondeterministic branches in the specification are refined to deterministic ones. Such an implementation may be done by using if-then-else and/or switch-case statements of some imperative programming language such as C, Java, etc. (for softwares), or VHDL, Verilog, etc. (for hardwares), and so on. However, in such an implementation the initially specified branching structure may be modified when it is viewed as a real-time communicating behavior, which is generally important for embedded systems containing I/O actions. For example, a nondeterminisic branch of some I/O actions *a*,*b* and *c* (these actions can be considered to any I/O actions such as read/write to I/O ports of some devices, and so on) in the initial specification may be implemented to the if-then-else statements such as "if (condition1) then *a* else if (condition2) then *b* else *c*". In this case, when we view the real-time communicating behavior of the implementation, the decision whether the action *a* is executed or not is already made after the time that the condition1 is evaluated. In the verification of equivalence between the specification and the refined implementation, we want to consider these behaviors as equivalent.

Some theoretical works on equivalence for real-time communicating systems while considering unobservable internal actions are proposed so far. Timed weak bisimulation equivalence was proposed to determine equivalence of processes considering both time and observability[9], but as pointed out in [4], timed weak bisimulation may not be suitable for equivalence checking of real-time systems when branching structures of a specification are modified in the implementation. Global timed bisimulation equivalence[4] is weaker than timed weak bisimulation equivalence and is less sensitive to the modification of branching structures. Unfortunately, global timed bisimulation equivalence is still too strong compared to timed failure equivalence[16, 3, 15]. Timed failure equivalence is considered to be a sufficient criterion of correct refinement of practical communicating systems in the sense that if two (finite state) systems *P* and *Q* are timed failure equivalent, then for any (finite state) external environment *R*, the composed communicating system of *P* and *R* behaves equivalently to that of *Q* and *R*. Therefore, we adopt timed failure equivalence[1] as the refinement relation between a specification and its refined implementation.

Moreover, it would be useful if we put real-time constraints containing parameters (e.g. upper-/lower-bounds), and derive automatically the constraint (e.g. the minimum or maximum value allowed) of parameters in which the equivalence is preserved. Such an analysis is called a parametric analysis[2, 17]. Parametric analysis is especially useful when the equivalence of the system strongly depends on the timings on its actions and we need to tune the timings to preserve the equivalence. Otherwise we need to do such a tuning in a try-and-error manner, that is, we fix all the timing parameters to some set of values, check the equivalence, and if it is failed, try again for another set of values, and so on. This is generally tedious.

To make such equivalence checking feasible, we abstract away the data dependent part of such an implementation and focus on the control part only. To capture the control flow of such system's specification and implementation with time constraints and perform a parametric analysis, we propose a parametric time-interval automaton (PTIA), which is a subset of a parametric timed automaton[2] having only a time interval with upper- and lower-bound parameters as a relative timing constraint between

[1]There are several variants of timed failure equivalence, some of those consider timed/untimed-stability, finite/infinite time refusals, nonzeno-condition, and so on[3, 15]. In this paper, we disregard those differences and adopt the simplest definition of timed failure equivalence.

consecutive actions. We show that timed failure equivalence checking for PTIAs can be reduced to existing parametric strong timed bisimulation equivalence checking.

There are some proposals of parametric equivalence checking for communicating systems. For bisimulation equivalence without time, parametric strong/weak bisimulation equivalence checking algorithms on STG (Symbolic Transition Graph) and STGA (STG with Assignment) are already proposed[6, 11, 10]. For timed strong bisimulation equivalence (bisimulation equivalence where both time and all actions are considered observable), parametric equivalence checking is proposed in [14]. However, as far as we know, parametric equivalence checking algorithm has not been proposed for any other time related equivalence, even for timed weak bisimulation equivalence (bisimulation equivalence where time is considered observable and internal actions are not considered observable).

In this paper, we propose a method to abstract away the difference of branching structures of internal actions from a given real-time communicating system model written in a PTIA, while preserving timed failure equivalence. Specifically, the proposed method convert a given PTIA that may contain some internal actions into the PTIA which does not contain any internal actions and is timed failure equivalent to the given PTIA. Here, in this paper we assume that the given PTIA does not contain any loops (i.e. its transition graph is a DAG(Directed Acyclic Graph)), its initial action must be an observable action, and every internal action is *observably bounded*, that is, it must appear between some observable actions in any action execution sequences. By combining the proposed abstraction method and the parametric timed strong bisimulation equivalence checking method proposed in [14], we can perform parametric timed failure equivalence checking.

The rest of this paper is organized as follows. In Section 2, we define the PTIA model and its operational semantics by defining a mapping from the model to a timed extension of labelled transition system (timed LTS). Section 3 describes the definition of timed equivalences on the timed LTS, including timed failure equivalence. We propose a transformation algorithm on the PTIA and prove that the transformation preserves timed failure equivalence in Section 4. In Section 5, we propose a parametric timed failure equivalence checking algorithm. Conclusions and future directions are given in Section 6.

2 Parametric Time-Interval Automata

Let Act and Var denote a set of actions and a set of variables, respectively. We denote the set of all real-numbers by $\mathbf{R}$ and the set of all non-negative real-numbers by $\mathbf{R}^+$. Let $\mathbf{N}_0$ [$\mathbf{N}$] be the set of all natural numbers including 0 [excluding 0, respectively]. Let $Intvl(Var)$ denote a set of formulas of the form either $e1 \leq t$, $t \leq e2$, or $e1 \leq t \wedge t \leq e2$, where $e1$ and $e2$ are linear arithmetic expression (that is, only addition and subtraction are allowed) over variables in $Var \setminus \{t\}$ and constants in $\mathbf{R}$, and $t \in Var$ is the special variable representing the elapsed time since the latest visit of the current control state.

Definition 1 *A* parametric time-interval automaton *is a tuple* $\langle S, \{t\}, PVar, E, s_{init}\rangle$*, where* S *is a finite set of* control states *(also referred to as* locations*),* $t \in Var$ *is the* clock variable*,* $PVar \subseteq Var$ *is a finite set of* parameters*,* $E \subseteq S \times (Act \cup \{\tau\}) \times Intvl(Var) \times S$ *is a transition relation,* s_{init} *is the initial state. Note that* τ *represents an* internal action*. On the other hand, every other action in Act represents an* observable action*. We write* $s_i \overset{a@?t[P]}{\longrightarrow} s_j$ *if* $(s_i, a, P, s_j) \in E$. □

Informally, a transition $s_i \xrightarrow{a@?t[P]} s_j$ means that the action a can be executed from s_i when the values of both the clock variable t and parameters satisfy the formula P (called a *guard condition*), and after executed, the state moves into s_j and the clock variable t is reset to zero. In any state s, the value of the clock variable t increases continuously, representing the time passage.

Formal semantics of parametric time-interval automata is similar to that of the parametric timed automata, which is defined as follows. The values of clocks and parameters are given by a function $\sigma : (\{t\} \cup PVar) \mapsto \mathbf{R}$. We refer to such a function as a *value-assignment*. We represent a set of all value-assignments by Val. We write $\sigma \models P$ if a formula $P \in Intvl(Var)$ is true under a value-assignment $\sigma \in Val$. The semantic behavior of a parametric timed automaton is given as a semantic transition system on *concrete states*. A concrete state is represented by (s, σ), where s is a control state and σ is a value-assignment. Let $CS \stackrel{\text{def}}{=} \{(s, \sigma) | s \in S, \sigma \in Val\}$ be a set of concrete states. The semantic model is a *timed labelled transition system (timed LTS)*, which is defined as follows. A state of a timed LTS is a concrete state in CS. A transition of a timed LTS is either a *delay-transition* or an *action-transition*. A delay transition represents a time passage within the same control state $s \in S$, whereas an action transition represents an execution of an action which changes the control state to the next one s'. Formally, a timed labelled transition system is defined as follows.

Definition 2 *A* timed labelled transition system *(a* timed LTS *for short) for a parametric time-interval automaton is a labelled transition system* $\langle CS, Act \cup \mathbf{R}^+ \cup \{\tau\}, CE, (s_{init}, \sigma_{init}[t \to 0])\rangle$*, where a set of states is* CS*, a set of labels is* $Act \cup \mathbf{R}^+ \cup \{\tau\}$*, an initial state is* $(s_{init}, \sigma_{init}[t \to 0])$*, and a transition relation* $CE \subseteq CS \times (Act \cup \mathbf{R}^+ \cup \{\tau\}) \times CS$ *is defined as the minimum set that satisfies the following conditions (in the following, we write* $(s, \sigma) \xrightarrow{l} (s', \sigma')$ *if* $((s, \sigma), l, (s', \sigma')) \in CE$*):*

- $(s, \sigma) \xrightarrow{v} (s, \sigma + v)$ *if* $v \in \mathbf{R}^+$,
- $(s, \sigma) \xrightarrow{a} (s', \sigma[t \to 0])$ *if* $a \in Act \cup \{\tau\}$, $s \xrightarrow{a@?t[P]} s'$, *and* $\sigma \models P$,

where $\sigma + v$ *and* $\sigma[t \to 0]$ *are the value-assignments derived from* σ*, which is defined as follows:*

$$\text{For } x \in PVar \cup \{t\}$$

$$(\sigma + v)(x) \stackrel{\text{def}}{=} \begin{cases} \sigma(x) + v & \text{if } x \in \{t\}, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

$$(\sigma[t \to 0])(x) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x \in \{t\}, \\ \sigma(x) & \text{otherwise.} \end{cases}$$

□

3 Timed Equivalences

In this section, we briefly recall the definition of timed failure equivalence[16, 3, 15], as well as the definition of the traditional timed weak bisimulation equivalence[1, 9] and its relation to timed failure equivalence.

3.1 Timed Weak Bisimulation Equivalence

In this section, we will briefly give the definition of timed weak bisimulation equivalence.

Definition 3 *For any* $\alpha \in Act \cup \{\tau\} \cup \mathbf{R}^+$, *a* timed weak transition relation $\stackrel{\alpha}{\Longrightarrow}_w$ *is a binary relation on states of a timed LTS* $\langle CS, Act \cup \mathbf{R}^+ \cup \{\tau\}, CE, (s_0, \sigma_0)\rangle$, *that is defined as follows:*

1. $\stackrel{\tau}{\Longrightarrow}_w \stackrel{\text{def}}{=} (\stackrel{0}{\longrightarrow} \cup \stackrel{\tau}{\longrightarrow})^*$

2. $(s,\sigma) \stackrel{v}{\Longrightarrow}_w (s',\sigma')\ (v \in \mathbf{R}^+)$
$$\stackrel{\text{def}}{=} \exists v_1, v_2, ..., v_n \in \mathbf{R}^+ \ [\ v = \textstyle\sum_{i=1}^{n} v_i$$
$$\wedge \exists s_1, \sigma_1, \sigma_1', s_2, \sigma_2, \sigma_2', ..., s_n, \sigma_n, \sigma_n'$$
$$s.t.\ \ (s,\sigma) \stackrel{\tau}{\Longrightarrow}_w (s_1,\sigma_1) \stackrel{v_1}{\longrightarrow} (s_1,\sigma_1') \cdots \stackrel{\tau}{\Longrightarrow}_w (s_n,\sigma_n) \stackrel{v_n}{\longrightarrow}$$
$$(s_n,\sigma_n') \stackrel{\tau}{\Longrightarrow}_w (s',\sigma')\ ]$$

3. $\stackrel{a}{\Longrightarrow}_w\ (a \in Act) \stackrel{\text{def}}{=} \stackrel{\tau}{\Longrightarrow}_w \stackrel{a}{\longrightarrow} \stackrel{\tau}{\Longrightarrow}_w$ □

By using this transition relation, timed weak bisimulation is defined as follows:

Definition 4 *A binary relation R on states of a timed LTS is a* timed weak bisimulation *if the following condition hold:*

If $(s_1,\sigma_1)R(s_2,\sigma_2)$, *then for any* $\alpha \in Act \cup \mathbf{R}^+ \cup \{\tau\}$,

1. $\forall\, s_1', \sigma_1' [\ (s_1,\sigma_1) \stackrel{\alpha}{\Longrightarrow}_w (s_1',\sigma_1')\ \Rightarrow$
$\exists s_2', \sigma_2'\ [\ (s_2,\sigma_2) \stackrel{\alpha}{\Longrightarrow}_w (s_2',\sigma_2') \wedge (s_1',\sigma_1')R(s_2',\sigma_2')\ ]\ ]$,
and,

2. $\forall\, s_2', \sigma_2' [\ (s_2,\sigma_2) \stackrel{\alpha}{\Longrightarrow}_w (s_2',\sigma_2')\ \Rightarrow$
$\exists s_1', \sigma_1'\ [\ (s_1,\sigma_1) \stackrel{\alpha}{\Longrightarrow}_w (s_1,\sigma_1') \wedge (s_1',\sigma_1')R(s_2',\sigma_2')\ ]\ ]$

We say that states (s_1,σ_1) *and* (s_2,σ_2) *are* timed weak bisimulation equivalent, *denoted by* $(s_1,\sigma_1) \equiv_{twb} (s_2,\sigma_2)$ *if and only if there exists a timed weak bisimulation R such that* $(s_1,\sigma_1)\ R\ (s_2,\sigma_2)$. □

3.2 Timed Failure Equivalence

Timed failure equivalence is a kind of equivalence between two (possibly nondeterministic) communicating processes such that their possibilities of communication failures are equivalent. Similar to many process algebras such as CCS[13], CSP[8], etc., we abstract every communication to the synchronization (rendezvous) of actions with the same action label (name) performed by multiple concurrent processes. That is, a communication happens between two processes P and Q if they perform the same action simultanously. We say that an action a in a process P is *offered* for communication if some other process in the external environment of P is ready to perform a and expect that P also performs a. An offered action a is called *refused* by P if P cannot perform a when offered. A communication failure occurs if the external environment has observed some sequence of actions performed by P and it offers some expected action a but P refuses it. This kind of communication failure possibility can be formally described by the pair (*trace*,*refusal*), where a *trace* is an observed sequence of performed

actions by P by now and a *refusal* is a set of actions that may be refused by P after *trace* has been observed. *Failure equivalence* (or *testing equivalence*[5], an operational semantic view of the failure equivalence by the notion of test), is an equivalence between (untimed) communicating processes that such possiblities of communication failures are equivalent. Since we consider nondeterminism, it is possible that traces are the same and refusals are different. Hence, failure equivalence is generally finer than *trace equivalence*, that is, the language equivalence in the traditional automata theory where any states are considered the accepting states. For timed communicating processes, we have to extend the notion of failure to the corresponding timed one. Several such extensions have been proposed in [7, 16, 3, 15, 12]. In this paper, we adopt the definition that is slightly modified from that of [16] for simplicity. Intuitively, a *timed failure* is a pair (*timed trace*, *timed refusal*), where a *timed trace* is an observed sequence of tuples of a performed action and its observed absolute time (the elapsed time from when the system has started), and a *timed refusal* is a set of tuples of a set of refused actions and the absolute time it is refused. The formal definition is as follows.

Definition 5 (Timed Failures) *Let* $TA \stackrel{\text{def}}{=} \{(t,a) | t \in \mathbf{R}^+, a \in Act\}$ *denote the set of all* timed actions. *Let* $TT \stackrel{\text{def}}{=} \{\langle (t_1,a_1)\cdots(t_k,a_k)\rangle | k \in \mathbf{N}_0 \forall i \in \{1,\ldots,k\}, (t_i,a_i) \in TA, \forall i,j \in \{1,\ldots,k\}, (i \le j) \Rightarrow (t_i \le t_j)\}$ *denote the set of all* (finite) timed traces. *Let* $TR \stackrel{\text{def}}{=} \{R | R \subseteq \mathbf{R}^+ \times Act\}$ *denote the set of all* timed refusals. *Then, the set of all* timed failures *is denoted by* $TF \stackrel{\text{def}}{=} \{(w,X) | w \in TT, X \in TR\}$. □

Definition 6 (Notations for Timed Traces and Timed Refusals) *For* $w \in TT$, *we denote the length of* w *as* $|w|$, *that is,* $|w| \stackrel{\text{def}}{=} k$ *if* $w = \langle (t_1,a_1)\cdots(t_k,a_k)\rangle$. *For* $X \in TR$ *and* $t \in \mathbf{R}^+$, $X + t \stackrel{\text{def}}{=} \{(t'+t,a) | (t',a) \in X\}$. □

Definition 7 (Timed Failures of Concrete States of Timed LTSs) *For any state* $(s,\sigma) \in CS$ *of a Timed LTS* $\langle CS, Act \cup \mathbf{R}^+ \cup \{\tau\}, CE, (s_{init}, \sigma_{init}[t \to 0])\rangle$, $TT((s,\sigma)) \stackrel{\text{def}}{=} \{\langle (t_1,a_1) \cdots (t_k,a_k)\rangle \mid \exists s_1,\ldots,s_k,s'_1,\ldots,s'_k \in S \;\; \exists \sigma_1,\ldots,\sigma_k,\sigma'_1,\ldots,\sigma'_k \in Val \;\; (s,\sigma) \stackrel{t_1}{\Longrightarrow}_w (s_1,\sigma_1) \stackrel{a_1}{\Longrightarrow}_w (s'_1,\sigma'_1) \cdots (s'_{k-1},\sigma'_{k-1}) \stackrel{t_k - t_{k-1}}{\Longrightarrow}_w (s_k,\sigma_k) \stackrel{a_k}{\Longrightarrow} (s'_k,\sigma'_k)\}$, $TR((s,\sigma)) \stackrel{\text{def}}{=} \{(t,a) | t \in \mathbf{R}^+, a \in Act, \neg[\exists s', s'' \in S, \; \exists \sigma', \sigma'' \in Val \;\; s.t. \;\; (s,\sigma) \stackrel{t}{\Longrightarrow}_w (s',\sigma') \stackrel{a}{\Longrightarrow}_w (s'',\sigma'')]\}$, *and* $TF((s,\sigma)) \stackrel{\text{def}}{=} \{(\langle (t_1,a_1)\cdots(t_k,a_k)\rangle, X) \mid \exists t_{k+1} \in \mathbf{R}^+ \;\; \exists s_1,\ldots,s_{k+1},s'_1,\ldots,s'_k \in S \;\; \exists \sigma_1,\ldots,\sigma_{k+1},\sigma'_1,\ldots,\sigma'_k \in Val \;\; (s,\sigma) \stackrel{t_1}{\Longrightarrow}_w (s_1,\sigma_1) \stackrel{a_1}{\Longrightarrow}_w (s'_1,\sigma'_1) \cdots (s'_{k-1},\sigma'_{k-1}) \stackrel{t_k - t_{k-1}}{\Longrightarrow}_w (s_k,\sigma_k) \stackrel{a_k}{\Longrightarrow}_w (s'_k,\sigma'_k) \stackrel{t_{k+1}-t_k}{\Longrightarrow}_w (s_{k+1},\sigma_{k+1}) \wedge X = TR((s_{k+1},\sigma_{k+1})) + t_{k+1}\}$. □

Definition 8 (Timed Failure Equivalence) *We say that concrete states* (s_1,σ_1) *and* (s_2,σ_2) *are* timed failure equivalent, *denoted by* $(s_1,\sigma_1) \equiv_{tf} (s_2,\sigma_2)$ *if and only if* $TF((s_1,\sigma_1)) = TF((s_2,\sigma_2))$. □

As for the relationship between timed weak bisimulation and timed failure equivalence, the following proposition holds.

Proposition 1 *For any two states* (s_1,σ_1) *and* (s_2,σ_2) *of a timed LTS, if* $(s_1,\sigma_1) \equiv_{twb} (s_2,\sigma_2)$, *then* $(s_1,\sigma_1) \equiv_{tf} (s_2,\sigma_2)$. □

4 Abstraction Algorithm

In this section, we propose some abstraction rules to eliminate internal actions of the PTIA, and show that their rules preserve timed failure equivalence.

In the following, first, we describe some restrictions on PTIAs which ensure the correctness of the proposed abstraction rules. Then, we describe the key idea and the details of the proposed abstraction rules. Finally, we show that the abstraction rules preserve timed failure equivalence.

4.1 Restrictions for Parametric Time-Interval Automata

In order to apply our proposed abstraction rules, we impose the following restrictions [RLoopFree], [RInitStability], and [RObsBounded] for an input PTIA M.

[RLoopFree] M contains no loops, that is, the transition graph of M is a DAG.

[RInitStability] The initial state s_{init} of M must be either a *stable* state, or reachable to a stable state by deterministic internal transitions (i.e. there are no branches along a path from the initial state to a stable state). Here, a *stable* state is a state whose every outgoing transition is observable.

[RObsBounded] Any internal transition contained in M is *observably bounded*, that is, for any execution path π of M, there exists an extension π' of the path π in that any internal transition is appeared between some observable transitions.

Formally, an internal transition $s \xrightarrow{\tau@t[P]} s'$ contained in M is *observably bounded* if for any transition sequence $s_{init} \xrightarrow{\alpha_1@?t[P_1]} s_1 \cdots s_{k-1} \xrightarrow{\alpha_k@?t[P_k]} s \xrightarrow{\tau@t[P]} s'$ ($k \in \mathbf{N}_0$, $\alpha_1, \ldots, \alpha_k \in Act \cup \{\tau\}$, $P_1, \ldots, P_k, P \in Intvl(Var)$) of M, there exists some $i \in \{1, \ldots, k\}$ such that $\alpha_i \in Act$, and for any transition sequence $s \xrightarrow{\tau@?t[P]} s' \xrightarrow{\beta_1@?t[Q_1]} s_1 \cdots s_{m-1} \xrightarrow{\beta_m@?t[Q_k]} s_m$ ($m \in \mathbf{N}_0, \beta_1, \ldots, \beta_m \in Act \cup \{\tau\}$, $Q_1, \ldots, Q_m \in Intvl(Var)$), there exists an extension of the sequence $s_m \xrightarrow{\beta_{m+1}@?t[Q_{m+1}]} s_{m+1} \cdots s_{m+l-1} \xrightarrow{\beta_{m+l}@?t[Q_{m+l}]} s_{m+l}$ ($l \in \mathbf{N}_0, \beta_{m+1}, \ldots, \beta_{m+l} \in Act \cup \{\tau\}$, $Q_{m+1}, \ldots, Q_{m+l} \in Intvl(Var)$) and there exists some $j \in \{1, \ldots, m+l\}$ such that $\beta_j \in Act$.

4.2 Abstraction Rules for Parametric Time-Interval Automata

For any internal transition that directly follows some observable transition, we can eliminate the internal transition based on the following principles:

1. The internal nondeterminism caused by the internal transition can be converted into the corresponding nondetermistic choice of the directly preceding observable transition, just the same as the classical equational theory of testing equivalence[5].

2. On the contrary, the time passage caused by the internal transition can be moved into those of the directly succeeding transitions which are either internal or observable.

This is the key idea for preserving timed failure equivalence. Since any internal action is observably bounded, by the restriction [RObsBounded], the sequence of internal actions can be completely eliminated from the beginning internal action (which directly follows some observable action) to the ending one (which is directly followed by some observable action).

However, if we want to transform some subgraph of the entire transition graph of M, we must ensure that such a subgraph transformation preserves equivalence of the entire transition graph. To make the discussion simple, we impose the restriction

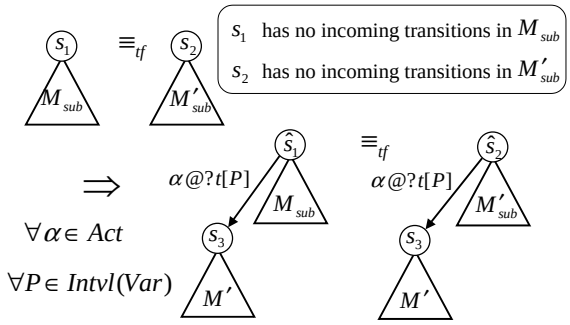

Figure 1: Congruence Property w.r.t. Timed Failure Equivalence

[RLoopFree]. By this restriction, we focus on the case that the transition graph is a DAG. Furthermore, we prove that if the context of the subgraph under transformation is in some form, such subgraph transformation preserves equivalence of the entire transition system M. To ensure that all internal transition can be eliminated by such a context-sensitive transformation, we need the restriction [RInitStability].

The proposed abstraction rules are the followings:

1. abstraction for sequential structures
2. abstraction for branching structures

The details are described in the following sections.

4.2.1 Context Sensitivity of Abstraction Rules

Consider that some subgraph M_{sub} of the entire transition graph of PTIA M is replaced into some equivalent subgraph M'_{sub}. Such a transformation does not always preserve equivalence of the entire transition graph. We will show that if M_{sub} appears in the context shown in Fig 1, then the subgraph transformation from M_{sub} to M'_{sub} preserves timed failure equivlaence

Theorem 1 (Congruence Property w.r.t. Timed Failure Equivalence) *Let $M = \langle S, \{t\}, PVar, E, s_{init}\rangle$ be a PTIA, and $s_1, s_2 \in S$ which have no incoming transitions from some state s that is reachable from s_1 and s_2 in M. Let $\hat{s_1}$ and $\hat{s_2}$ be new states obtained by adding the same observable branch to s_1 and s_2, that is, $\hat{s_1} \overset{\alpha@?t[P]}{\longrightarrow} s_3$ and $\hat{s_2} \overset{\alpha@?t[P]}{\longrightarrow} s_3$ for $\alpha \in Act$, $s_3 \in S$, $P \in Intvl(Var)$, $\hat{s_1} \overset{\beta@?t[Q_1]}{\longrightarrow} s'_1$ if $s_1 \overset{\beta@?t[Q_1]}{\longrightarrow} s'_1$ for $\beta \in Act \cup \{\tau\}$, $Q_1 \in Intvl(Var)$, and $\hat{s_2} \overset{\beta@?t[Q_2]}{\longrightarrow} s'_2$ if $s_2 \overset{\beta@?t[Q_2]}{\longrightarrow} s'_2$ for $\beta \in Act \cup \{\tau\}$, $Q_2 \in Intvl(Var)$.*

Then, for any $\sigma \in Val$, $(\hat{s_1}, \sigma) \equiv_{tf} (\hat{s_2}, \sigma)$ if $(s_1, \sigma) \equiv_{tf} (s_2, \sigma)$.

Proof. *We will only prove that $TF((\hat{s_1}, \sigma)) \subseteq TF((\hat{s_2}, \sigma))$, since the converse is similar by symmetry and hence the theorem will be proved. Choose arbitrary timed failure $(w, X) \in TF((\hat{s_1}, \sigma))$. We will show that $(w, X) \in TF((\hat{s_2}, \sigma))$. $w \in TT((\hat{s_1}, \sigma))$ can be expressed as $w = \langle (t_1, a_1) \cdots (t_k, a_k)\rangle$ for some $k \in \mathbf{N}_0$, $t_1, \ldots, t_k \in \mathbf{R}^+$, and $a_1, \ldots, a_k \in Act$. From Definition 7, $(\hat{s_1}, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_1^{(1)}, \sigma_1^{(1)}) \overset{t_2-t_1}{\Longrightarrow}_w \overset{a_2}{\Longrightarrow}_w \cdots \overset{t_k-t_{k-1}}{\Longrightarrow}_w \overset{a_k}{\Longrightarrow}_w (s_1^{(k)}, \sigma_1^{(k)}) \overset{t_{k+1}-t_k}{\Longrightarrow}_w (s_1^{(k+1)}, \sigma_1^{(k+1)})$ for some $t_{k+1} \in \mathbf{R}^+$, $s_1^{(1)}, \ldots, s_1^{(k+1)} \in S$, and*

$\sigma_1^{(1)}, \ldots, \sigma_1^{(k+1)} \in Val$. *From the definition of* $\hat{s_1}$, *either (0)* $k = 0$, *that is,* $w = \langle\rangle$ *(the empty trace) and* $(\langle\rangle, X) \in TF((\hat{s_1}, \sigma))$, *(1)* $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_1^{(1)}, \sigma_1^{(1)})$, *or (2)* $(\hat{s_1}, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_3, \sigma_1^{(1)})$ *holds.*

For the case (0), $(\hat{s_1}, \sigma) \overset{t_1}{\Longrightarrow}_w (s_1^{(1)}, \sigma_1^{(1)})$ *and* $X = TR((s_1^{(1)}, \sigma_1^{(1)})) + t_1$ *hold. If* $s_1^{(1)} = s_3$, *then obviously* $(\langle\rangle, X) \in TF((\hat{s_2}, \sigma))$ *holds because* $(\hat{s_2}, \sigma) \overset{t_1}{\Longrightarrow}_w (s_3, \sigma_1^{(1)})$ *and* $TR((s_1^{(1)}, \sigma_1^{(1)})) = TR((s_3, \sigma_1^{(1)}))$. *Otherwise, from the definition of* $\hat{s_1}$, $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w (s_1^{(1)}, \sigma_1^{(1)})$ *holds. Then, since* $(s_1, \sigma) \equiv_{tf} (s_2, \sigma)$, $(\langle\rangle, X) \in TF((s_2, \sigma))$ *holds and thus there exists some* $s_2^{(1)} \in S$ *and* $\sigma_2^{(1)} \in Val$ *such that* $(s_2, \sigma) \overset{t_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)})$ *and* $TR((s_2^{(1)}, \sigma_2^{(1)})) = TR((s_1^{(1)}, \sigma_1^{(1)}))$. *Then, from the definition of* $\hat{s_2}$, $(\hat{s_2}, \sigma) \overset{t_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)})$. *Therefore,* $(\langle\rangle, X) \in TF((\hat{s_2}, \sigma))$.

For the case (1), we can assume that $s_1^{(i)} \neq \hat{s_1}$ *for any* $i \in \{1, \ldots, k+1\}$, *because otherwise, from the assumption that there no incoming transitions from some state* s *that is reachable from* s_1, $k = 0$ *must hold and the proof is reduced to the case (0). Then* $(s_1^{(k+1)}, \sigma_1^{(k+1)})$ *is reachable from* (s_1, σ). *Thus,* $w \in TT((s_1, \sigma))$ *and* $X = TR((s_1^{(k+1)}, \sigma_1^{(k+1)})) + t_{k+1}$. *Hence,* $(w, X) \in TF((s_1, \sigma))$. *From the assumption that* $(s_1, \sigma) \equiv_{tf} (s_2, \sigma)$ *and Definition 8,* $(w, X) \in TF((s_2, \sigma))$. *Thus, from Definition 7, there exists some* $s_2^{(1)}, \ldots, s_2^{(k+1)}, \sigma_2^{(1)}, \ldots, \sigma_2^{(k+1)}$ *such that* $(s_2, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)}) \cdots \overset{t_k - t_{k-1}}{\Longrightarrow}_w \overset{a_k}{\Longrightarrow}_w (s_2^{(k)}, \sigma_2^{(k)}) \overset{t_{k+1} - t_k}{\Longrightarrow}_w (s_2^{(k+1)}, \sigma_2^{(k+1)})$ *and* $TR((s_2^{(k+1)}, \sigma_2^{(k+1)})) = TR((s_1^{(k+1)}, \sigma_1^{(k+1)}))$. *Then, from the definition of* $\hat{s_2}$, $(\hat{s_2}, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)}) \cdots \overset{t_k - t_{k-1}}{\Longrightarrow}_w \overset{a_k}{\Longrightarrow}_w (s_2^{(k)}, \sigma_2^{(k)}) \overset{t_{k+1} - t_k}{\Longrightarrow}_w (s_2^{(k+1)}, \sigma_2^{(k+1)})$ *also holds. Hence,* $(w, X) \in TF((\hat{s_2}, \sigma))$.

Similarly, $(w, X) \in TF((\hat{s_2}, \sigma))$ *can be proved for the case (2).*

Hence, $TF((\hat{s_1}, \sigma)) \subseteq TF((\hat{s_2}, \sigma))$. *By symmetry,* $TF((\hat{s_2}, \sigma)) \subseteq TF((\hat{s_1}, \sigma))$ *also holds. Therefore,* $(\hat{s_1}, \sigma) \equiv_{tf} (\hat{s_2}, \sigma)$.

4.2.2 Abstraction for Sequential Structures

The abstraction for sequential structures is illustrated in Fig. 2. In the left half of Fig 2, the transition $s_2 \overset{\tau@?t[x_2 \leq t \leq y_2]}{\longrightarrow} s_3$ is internal and its directly preceding transition $s_1 \overset{\alpha@?t[x_1 \leq t \leq y_1]}{\longrightarrow} s_2$ is observable. Its sibling $s_2 \overset{\beta@?t[x_3 \leq t \leq y_3]}{\longrightarrow} s_4$ and its directly succeeding transition $s_3 \overset{\gamma@?t[x_4 \leq t \leq y_4]}{\longrightarrow} s_5$ may be either internal or observable. In this case, we eliminate the internal transition $s_2 \overset{\tau@?t[x_2 \leq t \leq y_2]}{\longrightarrow} s_3$ as the right half of Fig 2. In this abstraction, the nondetermistic behavior of the eliminated internal transition is converted into the nondeterministic choice of the directly preceding observable transitions $s_1 \overset{\alpha@?t[x_1 \leq t \leq y_1]}{\longrightarrow} s_2$ and $s_1 \overset{\alpha@?t[x_1 \leq t \leq y_1]}{\longrightarrow} s_3$. On the other hand, the time passage of the eliminated internal transition is merged into the directly succeeding transitions $s_2 \overset{\gamma@?t[x_2 + x_4 \leq t \leq y_2 + y_4]}{\longrightarrow} s_5$ and $s_3 \overset{\gamma@?t[x_2 + x_4 \leq t \leq y_2 + y_4]}{\longrightarrow} s_5$.

Formally, the operation for merging two time intervals is defined as follows:

Definition 9 Θ *is a binary operator on* $Intvl(Var)$ *such that for any* $P, Q \in Intvl(Var)$, $(P \Theta Q)(t) \overset{\text{def}}{=} \exists t_1, t_2[P(t_1) \wedge Q(t_2) \wedge t = t_1 + t_2]$. □

Then, the following lemma holds:

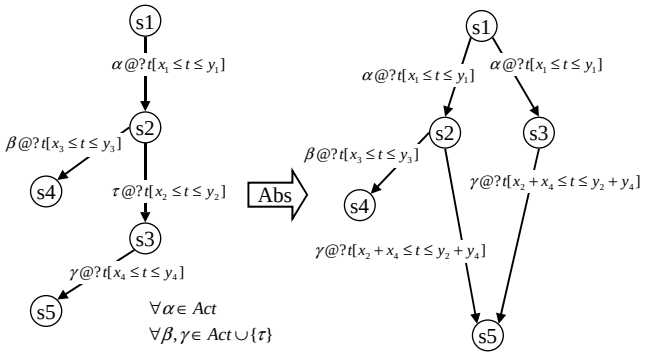

Figure 2: Abstraction for Sequential Structures

Lemma 1 *Let $\alpha \in Act$ and $\beta, \gamma \in Act \cup \{\tau\}$. Let M and M' be PTIAs whose sets of transtions are $\{ s_1 \xrightarrow{\alpha@?t[P_1]} s_2,\ s_2 \xrightarrow{\tau@?t[P_2]} s_3,\ s_2 \xrightarrow{\beta@?t[P_3]} s_4,\ s_3 \xrightarrow{\gamma@?t[P_4]} s_5 \}$ and $\{ s'_1 \xrightarrow{\alpha@?t[P_1]} s'_2,\ s'_1 \xrightarrow{\alpha@?t[P_1]} s'_3,\ s'_2 \xrightarrow{\beta@?t[P_3]} s_4,\ s'_2 \xrightarrow{\gamma@?t[P_2 \Theta P_4]} s_5,\ s'_3 \xrightarrow{\gamma@?t[P_2 \Theta P_4]} s_5 \}$, respectively. Then, for any assignment $\sigma \in Val$, $(s_1, \sigma) \equiv_{tf} (s'_1, \sigma)$.*

Proof. *Choose arbitrary timed failure $(w, X) \in TF((s_1, \sigma))$. We will show that $(w, X) \in TF(s'_1, \sigma)$. $w \in TT((s_1, \sigma))$ can be expressed as $w = \langle (t_1, a_1) \cdots (t_k, a_k) \rangle$ for some $k \in \mathbf{N}_0$, $t_1, \ldots, t_k \in \mathbf{R}^+$, and $a_1, \ldots, a_k \in Act$. From Definition 7, $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_1^{(1)}, \sigma_1^{(1)}) \overset{t_2 - t_1}{\Longrightarrow}_w \overset{a_2}{\Longrightarrow}_w \cdots \overset{t_k - t_{k-1}}{\Longrightarrow}_w \overset{a_k}{\Longrightarrow}_w (s_1^{(k)}, \sigma_1^{(k)}) \overset{t_{k+1} - t_k}{\Longrightarrow}_w (s_1^{(k+1)}, \sigma_1^{(k+1)})$ for some $t_{k+1} \in \mathbf{R}^+$, $s_1^{(1)}, \ldots, s_1^{(k+1)} \in S$ and $\sigma_1^{(1)}, \ldots, \sigma_1^{(k+1)} \in Val$ and $X = TR((s_1^{(k+1)}, \sigma_1^{(k+1)})) + t_{k+1}$. From the definition of M, $a_1 = \alpha$ if $k \geq 1$ and a_2 may be β [γ] if $k \geq 2$ and β [γ, resp.] is observable (i.e., $\beta \in Act$ [$\gamma \in Act$, resp.]).*

[Case of $k = 0$]

If $k = 0$, then $w = \langle\rangle$ and $X = TR((s_1^{(1)}, \sigma_1^{(1)})) + t_1$. From the definition of M, $s_1^{(1)} = s_1$, $\sigma_1^{(1)} = \sigma[t \to t_1]$, and $TR((s_1, \sigma_1[t \to t_1])) = \{(t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\}$. Thus $X = TR((s_1, \sigma_1[t \to t_1])) + t_1 = \{(t_1 + t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\}$. Then, from the definition of M', $(s'_1, \sigma) \overset{t_1}{\Longrightarrow}_w (s'_1, \sigma[t \to t_1])$ and $(w, X') \in TF((s'_1, \sigma)$ where $X' = TR((s'_1, \sigma[t \to t_1])) + t_1 = \{(t_1 + t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\} = X$. Therefore, $(w, X) \in TF((s'_1, \sigma))$.

[Case of $k = 1$ and $\beta, \gamma \in Act$]

If $k = 1$ and $\beta, \gamma \in Act$, then $w = \langle (t_1, \alpha) \rangle$, $\sigma[t \to t_1] \models P_1$, $s_1^{(1)} = s_2$, $\sigma_1^{(1)} = \sigma$, $s_1^{(2)}$ is either s_2 or s_3, and $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_2, \sigma) \overset{t_2 - t_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)})$.

If $s_1^{(2)} = s_2$, then $\sigma_2^{(1)} = \sigma[t \to t_2 - t_1]$ and $X = TR((s_2, \sigma[t \to t_2 - t_1])) + t_2 = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge \exists t'_1, t'_2[(t' = t'_1 + t'_2) \wedge (\sigma[t \to t_2 - t_1 + t'_1] \models P_2) \wedge (\sigma[t \to t'_2] \models P_4)]]\}$. Since $(s'_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s'_2, \sigma) \overset{t_2 - t_1}{\Longrightarrow}_w (s'_2, \sigma[t \to t_2 - t_1])$, $(w, X') \in TF((s'_1, \sigma))$ where $X' = TR((s'_2, \sigma[t \to t_2 - t_1])) + t_2 = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_2 \Theta P_4)]\}$. Since $(P_2 \Theta P_4)(t) \equiv \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)]$ and $\exists t'_1, t'_2[(t' = t'_1 + t'_2) \wedge (\sigma[t \to t_2 - t_1 + t'_1] \models P_2) \wedge (\sigma[t \to t'_2] \models P_4)]$ if and only if $\sigma[t \to t_2 - t_1 + t'] \models \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)]$ for any $t_2, t' \in \mathbf{R}^+$, $X' = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1 + t'] \models \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)])]\} = X$. Thus, $(w, X) \in TF((s'_1, \sigma))$.

If $s_1^{(2)} = s_3$*, then there exists some* $t_1' \in \mathbf{R}^+$ *such that* $\sigma[t \to t_1' - t_1] \models P_2$, $\sigma[t \to t_2 - t_1'] \models P_4$, $\sigma_2^{(1)} = \sigma[t \to t_2 - t_1']$, $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_2, \sigma) \overset{t_1'-t_1}{\Longrightarrow}_w (s_2, \sigma[t \to t_1']) \overset{t_2-t_1'}{\Longrightarrow}_w (s_3, \sigma[t \to t_2 - t_1'])$*, and* $X = TR((s_3, \sigma[t \to t_2 - t_1'])) + t_2 = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1' + t'] \models P_4)]\}$*. Since* $\sigma[t \to t_1' - t_1] \models P_2$ *and* $\sigma[t \to t_2 - t_1'] \models P_4$ *for some* $t_1' \in \mathbf{R}^+$, $\sigma[t \to t_2 - t_1] \models P_2 \Theta P_4$ *holds. Thus,* $(s_1', \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_3', \sigma) \overset{t_2-t_1}{\Longrightarrow}_w (s_3', \sigma[t \to t_2 - t_1])$ *holds. Moreover,* $(w, X') \in TF((s_1', \sigma))$ *where* $X' = TR((s_3', \sigma[t \to t_2 - t_1])) + t_2 = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_2 \Theta P_4)]\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge \exists t_3, t_4[(t_2 - t_1 = t_3 + t_4) \wedge (\sigma[t \to t_3] \models P_2) \wedge (\sigma[t \to t_4] \models P_4)]]\}$*. By letting* $t_3 = t_1' - t_1$ *and* $t_4 = t_2 - t_1'$, $X' = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge \exists t_1'[(\sigma[t \to t_1' - t_1] \models P_2) \wedge (\sigma[t \to t_2 - t_1'] \models P_4)]]\} = X$*. Therefore,* $(w, X) \in TF((s_1', \sigma))$*.*

Similarly, we can prove that $(w, X) \in TF((s_1', \sigma))$ *if* $k \geq 2$, $\beta = \tau$, *or* $\gamma = \tau$*. Therefore,* $TF((s_1, \sigma)) \subseteq TF((s_1', \sigma))$*.*

Conversely, choose arbitrary timed failure $(w, X) \in TF((s_1', \sigma))$*. We will show that* $(w, X) \in TF(s_1, \sigma)$*. Similar to the proof of* $TF((s_1, \sigma)) \subseteq TF((s_1', \sigma))$, $w = \langle (t_1, a_1) \cdots (t_k, a_k) \rangle$ *for some* $k \in \mathbf{N}_0$, $t_1, \ldots, t_k \in \mathbf{R}^+$, *and* $a_1, \ldots, a_k \in Act$*. From Definition 7,* $(s_1', \sigma) \overset{t_1}{\Longrightarrow}_w \overset{a_1}{\Longrightarrow}_w (s_1'^{(1)}, \sigma_1^{(1)}) \overset{t_2-t_1}{\Longrightarrow}_w \overset{a_2}{\Longrightarrow}_w \cdots \overset{t_k-t_{k-1}}{\Longrightarrow}_w \overset{a_k}{\Longrightarrow}_w (s_1'^{(k)}, \sigma_1^{(k)}) \overset{t_{k+1}-t_k}{\Longrightarrow}_w (s_1'^{(k+1)}, \sigma_1^{(k+1)})$ *for some* $t_{k+1} \in \mathbf{R}^+$, $s_1'^{(1)}, \ldots, s_1'^{(k+1)} \in S$ *and* $\sigma_1^{(1)}, \ldots, \sigma_1^{(k+1)} \in Val$ *and* $X = \{(t_{k+1} + t, a) | (t, a) \in TR((s_1'^{(k+1)}, \sigma_1^{(k+1)}))\}$*. From the definition of* M', $a_1 = \alpha$ *if* $k \geq 1$ *and* a_2 *may be* β *[*γ*] if* $k \geq 2$ *and* β *[*γ*, resp.] is observable (i.e.,* $\beta \in Act$ *[*$\gamma \in Act$*, resp.]).*

[Case of $k = 0$**]**

If $k = 0$*, then* $w = \langle\rangle$ *and* $X = \{(t_1 + t', a) | (t', a) \in TR((s_1'^{(1)}, \sigma_1^{(1)}))\}$*. From the definition of* M', $s_1'^{(1)} = s_1'$, $\sigma_1^{(1)} = \sigma[t \to t_1]$, *and* $TR((s_1', \sigma_1[t \to t_1])) = \{(t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\}$*. Thus* $X = \{(t_1 + t', a) | (t', a) \in TR((s_1', \sigma_1[t \to t_1]))\} = \{(t_1 + t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\}$*. Then, from the definition of* M, $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w (s_1, \sigma[t \to t_1])$ *and* $(w, X') \in TF((s_1, \sigma)$ *where* $X' = \{(t_1 + t', a) | (t', a) \in TR((s_1, \sigma[t \to t_1]))\} = \{(t_1 + t', a) | a \in Act \wedge \neg[(a = \alpha) \wedge (\sigma[t \to t_1 + t'] \models P_1)]\} = X$*. Therefore,* $(w, X) \in TF((s_1, \sigma))$*.*

[Case of $k = 1$ **and** $\beta, \gamma \in Act$**]**

If $k = 1$ *and* $\beta, \gamma \in Act$*, then* $w = \langle (t_1, \alpha) \rangle$, $\sigma[t \to t_1] \models P_1$, $s_1'^{(1)} = s_1'^{(2)}$, $\sigma_1^{(1)} = \sigma$, $s_1^{(2)}$ *is either* s_2' *or* s_3'*, and* $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_2^{(1)}, \sigma) \overset{t_2-t_1}{\Longrightarrow}_w (s_2^{(1)}, \sigma_2^{(1)})$*.*

If $s_1^{(1)} = s_1^{(2)} = s_2'$*, then* $\sigma_2^{(1)} = \sigma[t \to t_2 - t_1]$ *and* $X = \{(t_2 + t', a) | (t', a) \in TR((s_2', \sigma[t \to t_2 - t_1]))\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_2 \Theta P_4)]\}$*. Since* $(s_1, \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_2, \sigma) \overset{t_2-t_1}{\Longrightarrow}_w (s_2, \sigma[t \to t_2 - t_1])$, $(w, X') \in TF((s_1, \sigma))$ *where* $X' = \{(t_2 + t', a) | (t', a) \in TR((s_2, \sigma[t \to t_2 - t_1]))\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge \exists t_1', t_2'[(t' = t_1' + t_2') \wedge (\sigma[t \to t_2 - t_1 + t_1'] \models P_2) \wedge (\sigma[t \to t_2'] \models P_4)]]\}$*. Since* $(P_2 \Theta P_4)(t) \equiv \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)]$ *and* $\exists t_1', t_2'[(t' = t_1' + t_2') \wedge (\sigma[t \to t_2 - t_1 + t_1'] \models P_2) \wedge (\sigma[t \to t_2'] \models P_4)]$ *if and only if* $\sigma[t \to t_2 - t_1 + t'] \models \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)]$ *for any* $t_2, t' \in \mathbf{R}^+$, $X' = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \beta) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_3)] \wedge \neg[(a = \gamma) \wedge \sigma[t \to t_2 - t_1 + t'] \models \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)])]\} = X$*. Thus,* $(w, X) \in TF((s_1, \sigma))$*.*

If $s_1'^{(1)} = s_1'^{(2)} = s_3'$*, then* $\sigma_1'^{(2)} = \sigma[t \to t_2 - t_1]$, $\sigma[t \to t_2 - t_1] \models P_2 \Theta P_4$, $(s_1', \sigma) \overset{t_1}{\Longrightarrow}_w \overset{\alpha}{\Longrightarrow}_w (s_3', \sigma) \overset{t_2-t_1}{\Longrightarrow}_w (s_3', \sigma[t \to t_2 - t_1])$*, and* $(w, X) \in TF((s_1', \sigma))$ *where* $X = \{(t_2 + t', a) | (t', a) \in TR((s_3', \sigma[t \to t_2 - t_1]))\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a =$

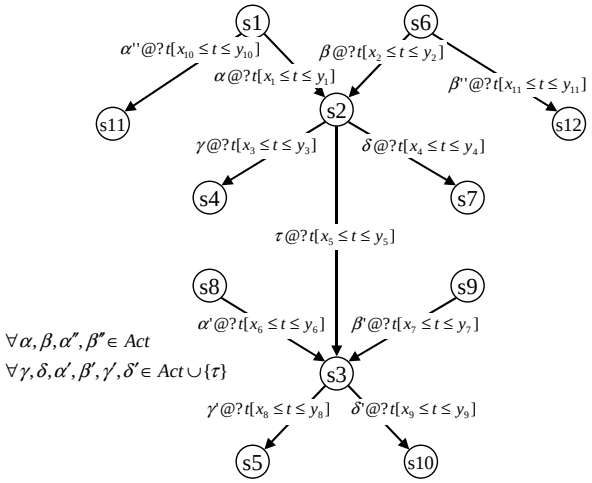

Figure 3: Abstraction for Sequential Structures in Presence of Outgoing/Incoming Transitions (Before Abstraction)

$\gamma) \wedge (\sigma[t \to t_2 - t_1 + t'] \models P_2 \Theta P_4)]\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge \exists t_3, t_4[(t_2 - t_1 = t_3 + t_4) \wedge (\sigma[t \to t_3] \models P_2) \wedge (\sigma[t \to t_4] \models P_4)]]\}$. *Then, since* $\sigma[t \to t_2 - t_1] \models P_2 \Theta P_4$, *there exists some* $t_1' \in \mathbf{R}^+$ *such that* $\sigma[t \to t_1' - t_1] \models P_2$, $\sigma[t \to t_2 - t_1'] \models P_4$, $(s_1, \sigma) \stackrel{t_1}{\Longrightarrow}_w \stackrel{\alpha}{\Longrightarrow}_w (s_2, \sigma) \stackrel{t_1' - t_1}{\Longrightarrow}_w (s_2, \sigma[t \to t_1']) \stackrel{t_2 - t_1'}{\Longrightarrow}_w (s_3, \sigma[t \to t_2 - t_1'])$, *and* $(w, X') \in TF((s_1, \sigma))$ *where* $X' = \{(t_2 + t', a) | (t', a) \in TR((s_3, \sigma[t \to t_2 - t_1'])\} = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge (\sigma[t \to t_2 - t_1' + t'] \models P_4)]\}$. *By letting* $t_3 = t_1' - t_1$ *and* $t_4 = t_2 - t_1'$, $X' = \{(t_2 + t', a) | a \in Act \wedge \neg[(a = \gamma) \wedge \exists t_3, t_4[t = t_3 + t_4 \wedge P_2(t_3) \wedge P_4(t_4)])]\} = X$. *Therefore,* $(w, X) \in TF((s_1, \sigma))$.

Similarly, we can prove that $(w, X) \in TF((s_1, \sigma))$ *if* $k \ge 2$, $\beta = \tau$, *or* $\gamma = \tau$. *Therefore,* $TF((s_1', \sigma)) \subseteq TF((s_1, \sigma))$.

Since $TF((s_1, \sigma)) \subseteq TF((s_1', \sigma))$, $TF((s_1, \sigma)) = TF((s_1', \sigma))$ *holds. From Definition 8,* $(s_1, \sigma) \equiv_{tf} (s_1', \sigma)$.

More general case is that there are some outgoing/incoming transitions on s_1, s_2 and s_3, as shown in Fig 3. In this case, all the source states of the incoming transition of s_2 must be stable in order to satisfy the congruence property in Theorem 1. In Fig 3, the outgoing transitions of the state s_1 and s_6 are observable. If this is satisfied, then all the incoming transitions of s_2 are converted into the nondeterministic choice, as shown in Fig 4. The internal transition $s_2 \stackrel{\tau@?t[x_5 \le t \le y_5]}{\longrightarrow} s_3$ is eliminated similarly, but since s_3 has some incoming transitions $s_8 \stackrel{\alpha'@?t[x_6 \le t \le y_6]}{\longrightarrow} s_3$ and $s_9 \stackrel{\beta'@?t[x_7 \le t \le y_7]}{\longrightarrow} s_3$, these transitions and the original outgoing transitions $s_3 \stackrel{\gamma'@?t[x_8 \le t \le y_8]}{\longrightarrow} s_5$ and $s_3 \stackrel{\delta'@?t[x_9 \le t \le y_9]}{\longrightarrow} s_{10}$ of the state s_3 are preserved. Then, the time passage of the transition $s_2 \stackrel{\tau@?t[x_5 \le t \le y_5]}{\longrightarrow} s_3$ are moved into the transitions $s_2 \stackrel{\gamma'@?t[x_5 + x_8 \le t \le y_5 + y_8]}{\longrightarrow} s_5$, $s_2' \stackrel{\gamma'@?t[x_5 + x_8 \le t \le y_5 + y_8]}{\longrightarrow} s_5$, $s_2 \stackrel{\delta'@?t[x_5 + x_9 \le t \le y_5 + y_9]}{\longrightarrow} s_{10}$, and $s_2' \stackrel{\delta'@?t[x_5 + x_9 \le t \le y_5 + y_9]}{\longrightarrow} s_{10}$, similarly.

Formally, the abstraction rule for sequential structures is defined as follows:

Definition 10 *The* Abstraction Rule for Sequential Structures *for PTIA is defined as a*

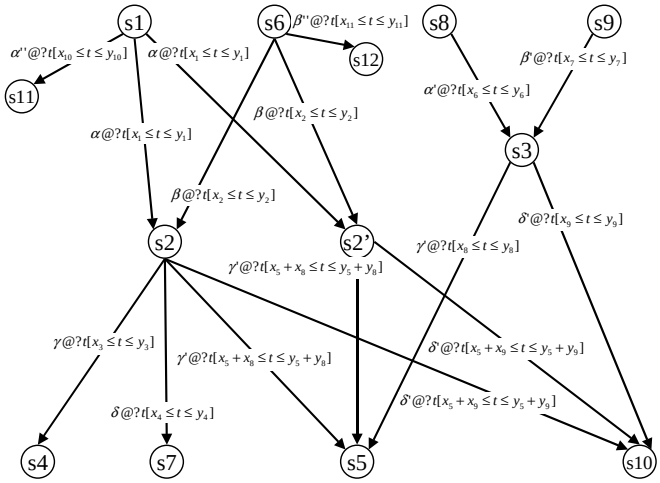

Figure 4: Abstraction for Sequential Structures in Presence of Outgoing/Incoming Transitions (After Abstraction)

subgraph transformation function AbsS eq(M) *on a PTIA M that transforms some subgraph* M_{sub} *of the transition graph of M into* M'_{sub}*, where* M_{sub} *consists of a set of transitions* $\bigcup_{i\in I}\{s_{1,i} \xrightarrow{\alpha_i@?t[P_{1,i}]} s_2\} \cup \{s_2 \xrightarrow{\tau@?[P_2]} s_3\} \cup \bigcup_{j\in J}\{s_2 \xrightarrow{\beta_j@?[P_{3,j}]} s_{4,j}\} \cup \bigcup_{k\in K}\{s_3 \xrightarrow{\gamma_k@?[P_{4,k}]} s_{5,k}\}$*,* M'_{sub} *consists of* $\bigcup_{i\in I}\{s_{1,i} \xrightarrow{\alpha_i@?t[P_{1,i}]} s_2,\ s_{1,i} \xrightarrow{\alpha_i@?t[P_{1,i}]} s_3\} \cup \bigcup_{j\in J}\{s_2 \xrightarrow{\beta_j@?[P_{3,j}]} s_{4,j}\} \cup \bigcup_{k\in K}\{s_2 \xrightarrow{\gamma_k@?[P_2\Theta P_{4,k}]} s_{5,k},\ s_3 \xrightarrow{\gamma_k@?[P_2\Theta P_{4,k}]} s_{5,k}\}$*, each* $s_{1,i}$ $(i \in I)$ *is a stable state,* $\alpha_i \in Act$*,* $\beta_j, \gamma_k \in Act \cup \{\tau\}$*, and* $P_{1,i}, P_2, P_{3,j}, P_{4,k} \in Intvl(Var)$ *for any* $i \in I$*,* $j \in J$*,* $k \in K$*.*

If there are some other incoming transitions $s'_6 \xrightarrow{\delta^l@?t[Q^l]} s_3$ $(l \in L)$*, then a new state* s'_3 *is created and all the incoming and outgoing transitions of* s_3 *is copied into those of* s'_3 *before applying this rule.* □

If it is not confused, we also consider the abstraction function *AbsS eq*() as the mapping from control states of M to the corresponding states of *AbsS eq*(M).

Then, we have the following theorem:

Theorem 2 *For any stable state s, if* $s \xrightarrow{\alpha@?t[P]} s_1 \xrightarrow{\tau@?t[Q]} s_2$*, any path* π *beginning with s contains no loops, and the internal transition* $s_1 \xrightarrow{\tau@?t[Q]} s_2$ *is observably bounded, then for any* σ*,* $(s,\sigma) \equiv_{tf} (AbsS\,eq(s),\sigma)$*.*

Proof. *(sketch) From Theorem 1 and Lemma 1, we can easily prove the general case by using induction on the number of the branches and using the congruence property (Theorem 1).*

4.2.3 Abstraction for Branching Structures

The abstraction for branching structures is illustrated in Fig. 5. It is clear that any external observer cannot find which branch is selected if each branch consists of one transition with the same action name, the same time constraint, and the same destination state. Thus, we leave just one of these branches. More generally, if there are

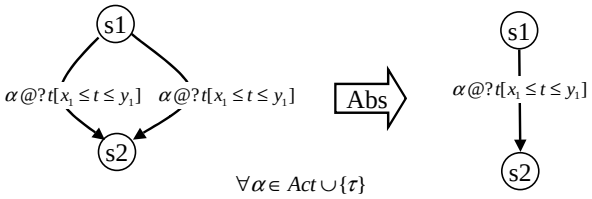

Figure 5: Abstraction for Branching Structures

multiple branches $s_1 \overset{a@?t[P_i]}{\longrightarrow} s_2$ ($i \in I$), then they are abstracted to just one transition $s_1 \overset{a@?t[\vee_{i\in I} P_i]}{\longrightarrow} s_2$.

Formally, the abstraction rule for branching structures is defined as follows:

Definition 11 *The* Abstraction Rule for Branching Structures *for PTIA is defined as a subgraph transformation function AbsBranch(M) on a PTIA M that transforms some subgraph M_{sub} of the transition graph of M into M'_{sub}, where M_{sub} consists of a set of transitions $\bigcup_{i\in I}\{s_1 \overset{\alpha@?t[P_i]}{\longrightarrow} s_2\}$, M'_{sub} consists of $\{s_1 \overset{\alpha@?t[\vee_{i\in I} P_i]}{\longrightarrow} s_2\}$, $\alpha \in Act \cup \{\tau\}$, and $P_i \in Intvl(Var)$ for any $i \in I$.* □

Similar to *AbsSeq*(), we consider the abstraction function *AbsBranch*() as the mapping from a PTIA M to the modified PTIA M'

Since the following theorem is rather straightforward, here we only show the results.

Theorem 3 *For any $s \in S$ and $\sigma \in Val$, $(s, \sigma) \equiv_{tf} (AbsBranch(s), \sigma)$.* □

4.3 Terminating Property of Abstraction Algorithm

Our proposed abstraction algorithm is to apply repeatedly the abstraction rules in Section 4.2 until no changes occur. In this section, we show that this abstraction algorithm is ensured to terminate.

Firstly, we define the abstraction algorithm more precisely.

Definition 12 Abstraction Algorithm *is defined as follows:*

1. *Input PTIA M.*
2. *Apply Abstraction Rule for Sequential Structures to M.*
3. *Apply Abstraction Rule for Branching Structures to M.*
4. *Repeat (2)-(3) until no changes occurred in M.*
5. *Output PTIA M.* □

Definition 13 *Let Abs() be the abstraction function which represents the application of either AbsSeq() or AbsBranch().* □

Then, the following theorem holds.

Theorem 4 *For any PTIA M, there exists some natural number n such that $Abs^n(M)$ contains no internal transitions. Here, $Abs^n(M)$ means the PTIA to which the abstraction rules are applied n times.*

Proof. *(sketch) From Definition 13, it can be proven that the function $Abs()$ generally monotonically decreases the number of internal transitions. Moreover, it can be shown that any internal transitions can be eliminated by the proposed abstraction rules if their directly preceding transitions are observable and they are observably bounded. Furthermore, since the transition graph is a DAG and the initial state is stable, we can repeatedly apply the abstraction rules from the top to the bottom of the DAG. From the fact above, we can prove the theorem.*

From this theorem, the following corollary immediately holds.

Corollary 1 *The abstraction algorithm in Definition 12 eventually terminates for any input M.* □

5 Equivalence Checking

In this section, we show that parametric timed failure equivalence checking on PTIA is reduced to parametric timed strong bisimulation checking on PTIA without internal transitions.

By applying the algorithm of Definition 12 to two PTIAs M_1 and M_2, we obtain two PTIAs $Abs(M_1)$ and $Abs(M_2)$, which have no internal transitions and timed failure equivalent to M_1 and M_2, respectively. On the other hand, from the result of Ref.[14], we can obtain the parameter condition in order that $Abs(M_1)$ and $Abs(M_2)$ are timed strong bisimulation equivalent. Since timed strong bisimulation equivalence implies timed failure bisimulation equivalence, and timed failure equivalence satisfies the transitive law, the obtained parameter condition is also the parameter condition in order that M_1 and M_2 are timed failure equivalent.

Definition 14 *A binary relation R on states of a timed LTS is a* timed strong bisimulation *if the following condition hold:*

If $(s_1, \sigma_1)R(s_2, \sigma_2)$, then for any $\alpha \in Act \cup \mathbf{R}^+ \cup \{\tau\}$,

1. $\forall\, s_1', \sigma_1'[\ (s_1, \sigma_1) \xrightarrow{\alpha} (s_1', \sigma_1')\ \Rightarrow$
   $\exists s_2', \sigma_2'\ [\ (s_2, \sigma_2) \xrightarrow{\alpha} (s_2', \sigma_2') \wedge (s_1', \sigma_1')R(s_2', \sigma_2')\ ]\ ]$,
   and,
2. $\forall\, s_2', \sigma_2'[\ (s_2, \sigma_2) \xrightarrow{\alpha} (s_2', \sigma_2')\ \Rightarrow$
   $\exists s_1', \sigma_1'\ [\ (s_1, \sigma_1) \xrightarrow{\alpha} (s_1, \sigma_1') \wedge (s_1', \sigma_1')R(s_2', \sigma_2')\ ]\ ]$

We say that states (s_1, σ_1) and (s_2, σ_2) are timed strong bisimulation equivalent, *denoted by $(s_1, \sigma_1) \equiv_{tsb} (s_2, \sigma_2)$ if and only if there exists a timed strong bisimulation R such that $(s_1, \sigma_1)\ R\ (s_2, \sigma_2)$.*

The following relationship holds among timed strong bisimulation equivalence, timed weak bisimulation equivalence, and timed failure equivalence

Proposition 2 *For any two concrete states (s_1, σ_1) and (s_2, σ_2) of a timed LTS, $(s_1, \sigma_1) \equiv_{tsb} (s_1, \sigma_1)$ implies $(s_1, \sigma_1) \equiv_{twb} (s_1, \sigma_1)$, and $(s_1, \sigma_1) \equiv_{tsb} (s_1, \sigma_1)$ implies $(s_1, \sigma_1) \equiv_{tf} (s_1, \sigma_1)$.* □

From the discussions above, the following theorem holds.

Theorem 5 *For any PTIAs M_1 and M_2, if there exists some natural numbers n and m such that $Abs^n(M_1)$ and $Abs^m(M_2)$ contain no internal transitions, and $Abs^n(M_1) \equiv_{tsb} Abs^m(M_2)$ if and only if $M_1 \equiv_{tf} M_2$.* □

6 Conclusion

In this paper, we proposed a parametric time-interval automaton(PTIA) and its transformation algorithm to eliminate internal actions while preserving timed failure equivalence, and showed that parametric timed failure equivalence checking on PTIAs can be reduced to the existing parametric timed strong bisimulation equivalence checking method without internal transitions.

The future work is to relax some of the restrictions imposed on target PTIAs, especially for the loops. For preserving timed failure equivalence, we confirmed that abstraction is still possible by the proposed abstraction rules in some cases containing loops, but there are some weird examples the proposed abstraction rules cannot be applied. On the other hand, for preserving timed trace equivalence, we are successfully developed the abstraction algorithm for unrestricted PTIAs. We are currently working on PTIAs containing various loop structures and developing more general abstraction algorithms for preserving timed failure equivalence and/or timed trace equivalence.

References

[1] R. Alur, C. Courcoubetis, and T. A. Henzinger. The observational power of clocks. In *Proc. of CONCUR'94*, volume 836 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1994.

[2] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proc. 25th ACM Annual Symp. on the Theory of Computing (STOC'93)*, pages 592–601, 1993.

[3] J. Davies and S. Schneider. A brief history of timed CSP. *Theoretical Comput. Sci.*, 138:243–271, 1995.

[4] D. de Frutos, N. López, and M. Núñez. Global timed bisimulation: An introduction. In *Proc. of Joint Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols XII, and Protocol Specification, Testing, and Verification XIX (FORTE/PSTV'99)*, pages 401–416, 1999.

[5] R. de Nicola and M. Hennessy. Testing equivalence for processes. *Theoretical Comput. Sci.*, 34:83–133, 1984.

[6] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Comput. Sci.*, 138:353–389, 1995.

[7] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.

[8] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[9] K. G. Larsen and Y. Wang. Time abstracted bisimulation: Implicit specifications and decidability. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Proc. of 9th Int'l Conf. on Mathematical Foundations of Programming Semantics (MFPS'93)*, volume 802 of *Lecture Notes in Computer Science*, pages 160–175. Springer-Verlag, Apr. 1993.

[10] Z. Li and H. Chen. Computing strong/weak bisimulation equivalences and observation congruence for value-passing processes. In *Proc. of Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 300–314, 1999.

[11] H. Lin. Symbolic transition graph with assignment. In *Proc. of CONCUR'96*, Lecture Notes in Computer Sciences. Springer-Verlag, Aug. 1996.

[12] G. Lowe and J. Ouaknine. On timed models and full abstraction. In *Proc. of the 21st Int. Conf. on Mathematical Foundations of Programming Semantics (MFPS2005)*, 2005.

[13] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[14] A. Nakata, T. Higashino, and K. Taniguchi. Time-action alternating model for timed LOTOS and its symbolic verification of bisimulation equivalence. In R. Gotzhein and J. Bredereke, editors, *Proc. of Joint Int'l Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification (FORTE/PSTV'96)*, pages 279–294. IFIP, Chapman & Hall, 1996.

[15] G. M. Reed and A. W. Roscoe. The timed failures-stability model for CSP. *Theoretical Comput. Sci.*, 211:85–127, 1999.

[16] S. Schneider. An operational semantics for timed CSP. *Information and Computation*, 116(2):193–213, 1995.

[17] F. Wang. Parametric timing analysis for real-time systems. *Information and Computation*, 130(2):131–150, 1996.