Framework of Timed Trace Theoretic Verification Revisited

Bin Zhou
Cadence Design Systems, Japan
binzhou@cadence.com

Tomohiro Yoneda*
Tokyo Institute of Technology
yoneda@cs.titech.ac.jp

Chris Myers[†]
University of Utah
myers@ee.utah.edu

Abstract

This paper develops a framework to support trace theoretic verification of timed circuits and systems. A theoretical foundation for classifying timed traces as either successes or failures is developed. The concept of the semimirror is introduced to allow conformance checking thus supporting hierarchical verification of timed circuits and systems. Finally, we relate our framework to those previously proposed for timing verification.

1 Introduction

For the formal verification of asynchronous circuits, a method called trace theoretic verification is proposed [1]. In this method, both specification and implementation are given by automata, Petri nets, or similar formal models. The advantages of this method are (1) its decision procedure is simple and efficient, (2) performance improvement by partial order reduction can be quite large [2, 3], and (3) verification can be done hierarchically.

Recently, designers often use timed circuits in order to implement fast and compact circuits [4, and others]. Thus, for the formal verification of such timed asynchronous circuits, it is desired to extend the verification methods so that they can handle timed circuits. Since it is not easy to apply symbolic methods based on BDDs to timing verification, partial order reduction is one of the most promising solutions to the state explosion problem. Thus, methods in which partial order reduction is well-suited are preferred. One direction is model checking based on real-time logics and timed transition systems [5, 6, and others]. The partial order reduction is applied to the timed LTL model checking in [7, 8], but it seems that the cost to handle LTL can be quite high. The second direction is the language containment checking based on timed automata [9]. In order to use this method for circuit verification, input generators are often required on the circuit side which makes hierarchical verification difficult. Furthermore, to our knowledge,

partial order reduction has not yet been applied to it in the timed domain. As the third direction, we have proposed a framework of timed trace theoretic verification based on time Petri nets [10, 11, 12]. The works most related to ours are [13, 14] and [15]. However, in the former work, the models are restricted such that a single transition has only one behavioral place, and in the latter work, the notion of mirrors is not discussed.

In [12, 16], safety failures and timing failures are defined, and the notion of pseudo failures is introduced. And then, the decision procedure based on pseudo failures is shown. Although this approach is certainly one extension of the trace theoretic verification, it differs from the original trace theoretic method in the following two points.

1. Failures are defined between modules, i.e., a module is defined without the failure trace set.

2. In timed trace theory, a mirror module obtained by swapping the inputs and the outputs of the original module is not equivalent to maximal environment of the original module. Hence, the decision procedure based on mirroring cannot be used for conformance checking.

In this paper, we reformalize the framework of the timed trace theoretic verification in accordance with the original untimed trace theory, and introduce the concepts of the semimodule and the semimirror to allow conformance checking.

2 Trace theoretic verification

We briefly recall the idea of the trace theoretic verification from [1] for the discussion of the next section.

A *Petri net* N is four-tuple, $N = (P, T, F, \mu_0)$, where P is a finite nonempty set of *places*, T is a finite set of *transitions* $(P \cap T = \emptyset)$, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*, and $\mu_0 \subseteq P$ is the *initial marking* of the net. For any transition t, $\bullet t = \{p \in P \mid (p, t) \in F\}$ and $t \bullet = \{p \in P \mid (t, p) \in F\}$ denote the *source places* and the *destination places* of t, respectively. For a place p, $\bullet p$ and $p \bullet$ are defined similarly.

A *marking* μ of N is any subset of P. A transition t is *enabled* in a marking μ if $\bullet t \subseteq \mu$ (all its source places have tokens in μ); otherwise, it is *disabled*. Let enabled(μ) be the set of transitions enabled in μ. In marking μ, transition $t_f \in$ enabled(μ) can fire, and a new marking $\mu' = (\mu - \bullet t_f) \cup t_f \bullet$ is obtained.

A *run* $\rho = \mu_0 \overset{t_1}{\to} \mu_1 \overset{t_2}{\to} \mu_2 \overset{t_3}{\to} \cdots$ of N is a finite or infinite sequence of markings and transitions such that μ_0 is the initial marking, and μ_{i+1} is obtained from μ_i by firing transition t_{i+1}. We assume that every prefix of a run is also a run. A Petri net is *one-safe*, if in any marking μ_i of any run $\rho = \mu_0 \overset{t_1}{\to} \mu_1 \overset{t_2}{\to} \mu_2 \overset{t_3}{\to} \cdots$, $(\mu_i - \bullet t_{i+1}) \cap t_{i+1} \bullet = \emptyset$. In the sequel, a *net* is always a one-safe Petri net.

A *trace* is a finite or infinite sequence of transitions. A run $\rho = \mu_0 \overset{t_1}{\to} \mu_1 \overset{t_2}{\to} \cdots$ generates a trace $t_1 t_2 \cdots$. Let trace(N) be a set of all the traces generated by a Petri net N. Note that the set trace(N) is prefix-closed and always includes an empty trace.

A module is a tuple (I, O, N), where I is a set of input wires, O is a set of output wires, and $N = (P, T, F, \mu_0)$ is a net, satisfying $I \cup O = T$. Note that we may call a transition a wire, since the firing of a transition t represents the change of value on a wire. We use a module as a formal model for a circuit element (e.g., a gate) or a specification. In the rest of this section, we use A (or $A_2, \cdots$) to represent $I \cup O$ (or $I_2 \cup O_2, \cdots$). A (canonical) *trace structure* of a module $M = (I, O, N)$, denoted by $\mathcal{T}(M)$, is a tuple (I, O, S, F), where S is the set of success traces, and F is the set of failure traces, satisfying

1. $S = $ trace(N), and

2. $F = \{x\eta \mid x \in (\{yi \mid y \in S, i \in I\} - S), \eta \in A^*\}$.

The above definition of failure traces can be interpreted such that if a trace obtained by appending an input to a success trace is not a success, then the trace is considered to be a failure. That is, an input which cannot be accepted always causes a failure. Thus, for $y \in S$, $i \in I$, $o \in O$, and $P = S \cup F$,

$$yi \notin S \quad \Rightarrow \quad yi \in F, \tag{1}$$

$$yo \notin S \quad \Rightarrow \quad yo \notin P, \tag{2}$$

hold. A trace in P is called *possible*. Furthermore, a trace extended from a failure is also a failure, and a trace extended from an impossible trace is also impossible. Hence, for $a \in A$, we have

$$y \in F \quad \Rightarrow \quad ya \in F, \tag{3}$$

$$y \notin P \quad \Rightarrow \quad ya \notin P. \tag{4}$$

A trace structure is called *failure-free* if its failure set is empty.

We use the following definitions in this paper. For a trace $x = e_1 e_2 e_3 \cdots$ and a set D of wires,

$$\text{del}(D, x) = \begin{cases} y & \text{if } e_1 \in D \\ e_1 y & \text{else,} \end{cases}$$

where $y = \text{del}(D, e_2 e_3 \cdots)$. For a set X of traces, $\text{del}^{-1}(D, X)$ is the set $\{x \mid \text{del}(D, x) \in X\}$. For X whose elements do not contain any wires in D, $\text{del}^{-1}(D, X)$ is the set of all traces that can be generated by inserting members of D^* between any consecutive wires in traces of X. This is extended for a trace structure $T = (I, O, S, F)$ such that $\text{del}^{-1}(D, T) = (I \cup D, O, \text{del}^{-1}(D, S), \text{del}^{-1}(D, F))$. Note that the inserted wires are always considered to be the inputs. The *intersection* of two trace structures $T_1 = (I_1, O_1, S_1, F_1)$ and $T_2 = (I_2, O_2, S_2, F_2)$ such that $A_1 = A_2$ and $O_1 \cap O_2 = \emptyset$ is a trace structure

$$T_1 \cap T_2 = (I_1 \cap I_2, O_1 \cup O_2, S_1 \cap S_2, (F_1 \cap P_2) \cup (P_1 \cap F_2)).$$

The *composition* of T_1 and T_2 such that $O_1 \cap O_2 = \emptyset$ is a trace structure

$$T_1 \| T_2 = \text{del}^{-1}(A_2 - A_1, T_1) \cap \text{del}^{-1}(A_1 - A_2, T_2).$$

Now, we show the core notions of the trace theoretic verification. For two modules $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$, we say M_1 *conforms* to M_2, if $I_1 \subseteq I_2$, $O_1 \supseteq O_2$, and for any (composable) trace structure $E = (I_E, O_E, S_E, F_E)$ such that $I_2 \subseteq O_E$ and $O_2 \supseteq I_E$, if $E \| \mathcal{T}(M_2)$ is failure-free, so is $E \| \mathcal{T}(M_1)$. This is the correctness criterion of trace theoretic verification. That is, we consider that a circuit M_C is correct with respect to a specification M_S, if M_C conforms to M_S. For a trace structure $T = (I, O, S, F)$, its maximum environment is called a *mirror* of T, denoted by T^m, which is also a trace structure (I', O', S', F') satisfying $I' = O$, $O' = I$, $S' = S$, and $F' = A^* - P$. The following lemma holds.

Lemma 1 Suppose that $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$ are modules such that $I_1 \subseteq I_2$ and $O_1 \supseteq O_2$. M_1 conforms to M_2 iff $\mathcal{T}(M_1) \| \mathcal{T}(M_2)^m$ is failure free.

Lemma 1 is very important, because we can check conformance without considering all possible environments E. But, it is useful only when the mirror of a trace structure is easily obtained. This is guaranteed by the following lemma.

Lemma 2 For a module $M = (I, O, N)$, $\mathcal{T}(M)^m$ is $\mathcal{T}(M')$, where $M' = (O, I, N)$.

We say that a module (O, I, N) is the mirror of a module $M = (I, O, N)$, denoted by M^m. Note that M^m is simply obtained by swapping inputs and outputs of M. These two lemmas straightforwardly derive the following theorem, which actually allows us to implement the decision procedure for conformance checking.

Theorem 1 Suppose that $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$ are modules such that $I_1 \subseteq I_2$ and $O_1 \supseteq O_2$. M_1 conforms to M_2 iff $\mathcal{T}(M_1) \| \mathcal{T}(M_2^m)$ is failure free.

3 Timed trace theoretic verification

3.1 Definitions

Let $\mathbf{Q}^+$ be the domain of nonnegative rational numbers for *time-points*. For a transition t and $\tau \in \mathbf{Q}^+$, (t, τ) is called a *timed event*. τ represents the time when the transition t fires, or the corresponding wire changes. A *timed trace* x is a finite or infinite sequence of timed events $x = e_0 e_1 \cdots$ where $e_i = (t_i, \tau_i)$ such that *monotonicity* (i.e., $\tau_i \leq \tau_{i+1}$ for all $i \geq 0$) and *progress* (i.e., if x is infinite, then for any $\tau \in \mathbf{Q}^+$ there exists an i such that $\tau_i > \tau$) are satisfied.

A *time Petri net* N is a six-tuple, $N = (P, T, F, \mathsf{Eft}, \mathsf{Lft}, \mu_0)$, where P, T, F, and μ_0 are the same as a Petri net, and $\mathsf{Eft} : T \to \mathbf{Q}^+$, $\mathsf{Lft} : T \to \mathbf{Q}^+ \cup \{\infty\}$ are functions for the *earliest* and *latest firing times* of transitions, satisfying $\mathsf{Eft}(t) \leq \mathsf{Lft}(t)$ for all $t \in T$.

A *state* σ of a time Petri net is a pair (μ, clock), where μ is a marking and clock is a function $T \to \mathbf{Q}^+$. The *initial state* σ_0 is $(\mu_0, \mathsf{clock}_0)$, where $\mathsf{clock}_0(t) = 0$ for all $t \in T$. The states of a time Petri net change, if time passes or if a transition fires. In state $\sigma = (\mu, \mathsf{clock})$, time $\tau \in \mathbf{Q}^+$ can pass, if for all $t \in \mathsf{enabled}(\mu)$, $\mathsf{clock}(t) + \tau \leq \mathsf{Lft}(t)$. In this case, state $\sigma' = (\mu', \mathsf{clock}')$ is obtained from σ by passing τ, if $\mu' = \mu$, and for all $t \in T$, $\mathsf{clock}'(t) = \mathsf{clock}(t) + \tau$. In state $\sigma = (\mu, \mathsf{clock})$, transition $t_f \in T$ can fire, if $t_f \in \mathsf{enabled}(\mu)$ and $\mathsf{clock}(t_f) \geq \mathsf{Eft}(t_f)$. In this case, state $\sigma' = (\mu', \mathsf{clock}')$ is obtained from σ by firing t_f, if $\mu' = (\mu - \bullet t_f) \cup t_f \bullet$, and for all $t \in T$, $\mathsf{clock}'(t) = 0$ for newly enabled transitions t and $\mathsf{clock}'(t) = \mathsf{clock}(t)$ for the others.

A *run* $\rho = \sigma_0 \xrightarrow{t_1} \sigma_1 \xrightarrow{t_2} \sigma_2 \xrightarrow{t_3} \cdots$ of N is a finite or infinite sequence of states and transitions such that σ_0 is the initial state, and σ_{i+1} is obtained from σ_i by passing some time and then firing transition t_{i+1}. $\mathsf{time}_i(\rho)$ is the sum of all times passed between $\sigma_0(\rho)$ and $\sigma_i(\rho)$; that is, $\mathsf{time}_0(\rho) = 0$ and $\mathsf{time}_{i+1}(\rho) = \mathsf{time}_i(\rho) + \mathsf{clock}_{i+1}(t) - \mathsf{clock}_i(t)$ for some t which is not newly enabled in μ_{i+1}.[1] Thus, a run ρ generates a timed trace $(t_1, \mathsf{time}_1(\rho))(t_2, \mathsf{time}_2(\rho)) \cdots$. Let $\mathsf{trace}(N)$ be a set of all timed traces generated by a time Petri net N. In order to satisfy the progress condition, we assume that time certainly passes in any loop structure that N contains. In the sequel, a *time net* is always a one-safe time Petri net satisfying the above restriction.

Suppose that a timed trace y is an element of $\mathsf{trace}(N)$. Let $\mathsf{enabled}(y, N)$ denote a set of transitions (of N) which are enabled in the state obtained by y. For a transition $t \in \mathsf{enabled}(y, N)$, $\mathsf{EN\_time}(t, y, N)$ is the time when t got newly enabled in y most recently. $\mathsf{TL}(y, N) = min\{ \mathsf{EN\_time}(t, y, N) + \mathsf{Lft}(t) \mid t \in \mathsf{enabled}(y, N)\}$ is the latest time until when the firings of all enabled transitions can be postponed. If $\mathsf{enabled}(y, N)$ is empty, we assume that

---

[1] In order to define $\mathsf{time}_i(\rho)$ precisely, we may consider an auxiliary transition t_{aux} which never becomes enabled.

$\mathsf{TL}(y, N)$ is ∞. $\mathsf{limit}(y, N) = \{ t \mid t \in \mathsf{enabled}(y, N), \mathsf{EN\_time}(t, y, N) + \mathsf{Lft}(t) = \mathsf{TL}(y, N) \}$ is a set of enabled transitions which determine $\mathsf{TL}(y, N)$. We say that a timed trace $y(w, \tau)$ is *overstepped*, if it satisfies $\tau > \mathsf{TL}(y, N)$.

3.2 Timed trace structures

A module is defined similarly except that N is a time net. A timed trace structure for a module (I, O, N) is also a tuple (I, O, S, F), but S and F are now sets of timed traces. We classify non-overstepped timed traces into S, F, or neither of them in the same way as the untimed cases. That is, S always includes an empty timed trace, and for a timed trace $y(w, \tau)$ such that $y \in \mathsf{trace}(N)$ and $\tau \leq \mathsf{TL}(y, N)$,

- if $y(w, \tau) \in \mathsf{trace}(N)$, then $y(w, \tau) \in S$,

- else if $w \in I$, then $y(w, \tau) \in F$ (from property (1)),

- else (i.e., $w \in O$), $y(w, \tau) \notin P$ (from property (2)).

On the other hand, it has never been considered which set an overstepped timed trace belongs to. Actually, S and F cannot be determined as simply as in the untimed cases, because some timed events cannot occur after a certain time point due to the timing constraints on the transitions. Thus, before classifying overstepped timed traces, we intuitively discuss when failures should and should not occur.

We basically consider that modules interacting with each other have some kind of failure, if either an output produced by a module cannot be accepted by some other module or an input expected by a module is not given in time by any other module. In the example shown in Figure 1, the output of M_1 is safely accepted by M_2, and the input of M_1 is given in time. Note that input transitions fire in synchronization with the corresponding output transitions. In the example shown in Figure 2, the output w of M_2 can be accepted by M_1, if it occurs before u, and otherwise, the firing of w is disabled by that of u. Since these properties hold for the other transitions, it is natural to consider that there exist no failures in these examples. On the other hand, in the modules shown in Figure 3(a),(b), the input expected by M_1 is never given by M_2, because the corresponding output is disabled. Neither does the input u of M_2 in Figure 3(a). This is a situation called deadlock, and so, there must exist some kind of failure between them. In Figure 4, $(w, 8)$, for example, can be produced by M_2, but it cannot be accepted by M_1 due to its latest firing time. Thus, a failure exists also in this case.

The key criterion to classify overstepped timed traces is whether $\mathsf{limit}(y, N)$ includes an output or not. If an output is included in $\mathsf{limit}(y, N)$, then N itself does not allow time to pass after $\mathsf{TL}(y, N)$ without firing any transitions. It is under the control of N. Thus, we should consider that the overstepped timed traces are not in P. On the other hand, if every wire in $\mathsf{limit}(y, N)$ is an input, N can only wait for the other modules to produce the expected outputs. In other
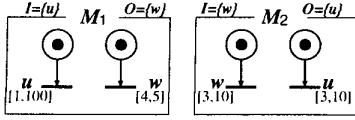
Figure 1. Example 1.

Figure 2. Example 2.

(a)
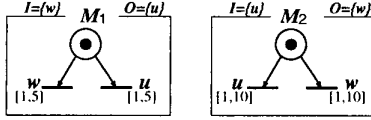

(b)

Figure 3. Example 3.
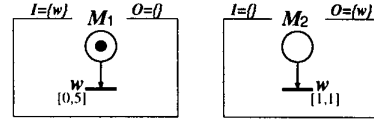

Figure 4. Example 4.

words, if no outputs are produced, the only action that N can do is to fail. Thus, it is natural to consider that those overstepped timed traces are in F.

Consider the modules M_1 and M_2 shown in Figure 1. We have $\mathsf{TL}(\varepsilon, N_1) = 5$ and $\mathsf{TL}(\varepsilon, N_2) = 10$, and $\mathrm{limit}(\varepsilon, N_1)$ has an output w and $\mathrm{limit}(\varepsilon, N_2)$ has an output u. Thus, for $\tau > 5$, either (w, τ) or (u, τ) is not in P_1, and for $\tau > 10$, either (w, τ) or (u, τ) is not in P_2. From this, the overstepped timed traces of M_1 and M_2 cannot be in the failure set of $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$. The non-overstepped timed traces cannot be, either. Hence, we can conclude that $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$ is failure-free. Thus, the above criterion is consistent to our intuition. A similar discussion holds for the example in Figure 2. Note that $y(w, \tau)$ is not in P independent of the attribute (i.e., input or output) of w, if it is overstepped and $\mathrm{limit}(y, N) \cap O \neq \emptyset$.

In the modules shown in Figure 3(a), on the other hand, $\mathrm{limit}(\varepsilon, N_1)$ contains only an input w. According to the above criterion, we consider that $(w, \tau) \in F_1$ for $\tau > 5$. However, if $(w, \tau) \notin P_2$ holds, we cannot put this timed trace into the failure set of $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$. Thus, we propose to consider that $(w, \tau) \in F_2$, even if w is disabled in M_2. Note that this is consistent to the above criterion, because $\mathrm{limit}(\varepsilon, N_2)$ contains only an input, and (w, τ) is overstepped in M_2. Then, we have $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$ whose failure set contains (w, τ) for $\tau > 5$. The same discussion holds for (u, τ). Unfortunately, we still have a problem in the example shown in Figure 3(b) where some failure must exist too. Since there are no enabled transitions in M_2, which implies $\mathsf{TL}(y, N_2) = \infty$, it's impossible to put some timed trace except for ε into P_2, and so, it's difficult to force $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$ to have a failure. In order to overcome this problem, we further propose to assume that every module has a virtual output which is always disabled, and that every other module has the corresponding virtual input which is also always disabled. In the example of Figure 3(b), we assume that M_1 has a disabled virtual output v_1 and M_2 has a disabled virtual input v_1 as shown in Figure 5. Similarly, v_2 is the disabled virtual output of M_2 and the disabled virtual input of M_1. Then, according to the
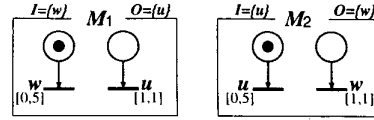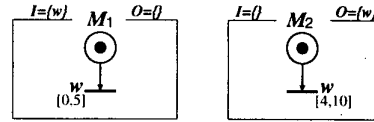
above criteria, we have $(v_1, \tau) \in F_1$ with $\tau > 5$ because this is overstepped, and we have $(v_1, \tau) \in F_2$ because this is neither overstepped nor in $\mathrm{trace}(N_2)$.

In Figure 4, $\mathrm{limit}(\varepsilon, N_1)$ includes only an input, while $\mathrm{limit}(\varepsilon, N_2)$ has an output. Thus, we have $(w, \tau) \in F_1$ for $\tau > 5$ and $(w, \tau') \notin P_2$ for $\tau' > 10$. Our intuition is satisfied also in this case, because (w, τ'') for $5 < \tau'' \leq 10$ is included in F_1 and S_2, and is in the failure set of $\mathcal{T}(M_1) \| \mathcal{T}(M_2)$.

Finally, criteria C1 can be used for both non-overstepped and overstepped timed traces. 2(b) and (c) are for the overstepped timed traces. 2(b) is for the case that the limit set consists of only inputs, and handles the failures explained for Figure 3 and Figure 4. 3 and 4 below are from properties (3) and (4) of Section 2.

C1: 1. $S = \mathrm{trace}(N)$

    2. for $y \in S$ and $y(w, \tau) \notin S$,

        (a) if $\tau \leq \mathsf{TL}(y)$, then

            i. if $w \in I$, then $y(w, \tau) \in F$

            ii. else $y(w, \tau) \notin P$

        (b) else if $\mathrm{limit}(y, N) \subseteq I$, then $y(w, \tau) \in F$

        (c) else $y(w, \tau) \notin P$

    3. for $y \in F$, $y(w, \tau) \in F$

    4. for $y \notin P$, $y(w, \tau) \notin P$

    where $w \in I \cup O$, and $\tau \in \mathbf{Q}^+$.

3.3 Mirroring

For untimed systems, Theorem 1 allows us to decide the conformance between a circuit and a specification very easily by using a mirror module. Here, we present a similar approach for timed systems.
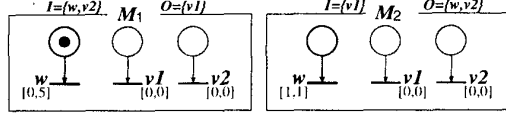
Figure 5. Virtual transitions for Example 3(b).

Also for a timed trace structure (I, O, S, F), its mirror is defined as a timed trace structure $(O, I, S, A^* - P)$, where $P = S \cup F$, $A = \{(w, \tau) \mid w \in I \cup O, \tau \in \mathbf{Q}^+\}$, and A^* is a set of all timed traces over A. Thus, Lemma 1 holds also in timed cases. However, Lemma 2 does not hold in general. Suppose that $M_1 = (\{w\}, \{u\}, N_1)$ is the module shown in Figure 2, and $\mathcal{T}(M_1) = (\{w\}, \{u\}, S_1, F_1)$ with $P_1 = S_1 \cup F_1$. According to C1 above, we have, for example, $(w, 6) \notin P_1$. If we consider $M_1' = (\{u\}, \{w\}, N_1)$, and $\mathcal{T}(M_1') = (\{u\}, \{w\}, S_1', F_1')$ with $P_1' = S_1' \cup F_1'$. we have $(w, 6) \notin P_1'$ again from C1. This derives $(w, 6) \notin F_1'$ and so $F_1' \neq A_1^* - P_1$, and that $\mathcal{T}(M_1')$ is not a mirror of $\mathcal{T}(M_1)$. Hence, Lemma 2 does not hold in this case. Actually, there does not exist a module whose timed trace structure is exactly the same as the mirror of this $\mathcal{T}(M_1)$.

However, we can still construct an algorithm to decide the conformance of timed systems as follows. We first define a *semimodule* (I, O, N), which is the same as a module except that its timed trace structure is defined differently from that of a module. The timed trace structure (I, O, S, F), denoted by $\mathcal{T}_s(M)$, of a semimodule $M = (I, O, N)$ is defined as follows.

C2:
1. $S = \text{trace}(N)$
2. for $y \in S$ and $y(w, \tau) \notin S$,
   (a) if $\tau \leq \text{TL}(y)$, then
      i. if $w \in I$, then $y(w, \tau) \in F$
      ii. else $y(w, \tau) \notin P$
   (b) else if $\text{limit}(y, N) \subseteq O$, then $y(w, \tau) \notin P$
   (c) else $y(w, \tau) \in F$
3. for $y \in F, y(w, \tau) \in F$
4. for $y \notin P, y(w, \tau) \notin P$

where $w \in I \cup O$, and $\tau \in \mathbf{Q}^+$.

Note that only 2(b),(c) are different from C1. If we consider a semimodule $(\{u\}, \{w\}, N_1)$ in the previous example, its timed trace structure is exactly a mirror of $\mathcal{T}(M_1)$. Actually, we have the following lemma.

Lemma 3 For a module $M = (I, O, N)$ and a semimodule $M' = (O, I, N), \mathcal{T}(M)^m = \mathcal{T}_s(M')$ holds.
Proof: Let $\mathcal{T}(M) = (I, O, S, F)$ and $\mathcal{T}_s(M') = (I', O', S', F')$, where $I' = O$ and $O' = I$. Then, $\mathcal{T}(M)^m = (O, I, S, A^* - P)$. From $S = \text{trace}(N)$ and $S' = \text{trace}(N)$, we have $S' = S$. Thus, we only have to show $F' = A^* - P$,

that is, both $x \in F' \Rightarrow x \in A^* - P$ and $x \notin F' \Rightarrow x \notin A^* - P$. If x is non-overstepped, the proof is similar to that of Lemma 2. Thus, we only consider the case that x is overstepped. This can be proven by induction on the length of η, denoted by $|\eta|$, where $x = y\eta$ and y is the longest prefix of x such that $y \in S'$. First, suppose $x \in F'$. We show that $x \in A^* - P$ holds. If $|\eta| = 1$, then from 2(b) and 2(c) of C2, $\text{limit}(y, N) \not\subseteq O' = I$ must hold in $\mathcal{T}_s(M')$. Thus, in $\mathcal{T}(M)$, from 2(c) of C1, $x \notin P$ holds, and it implies that $x \in A^* - P$ holds. If $|\eta| > 1$, let $x = y\eta = y'a$ with $a \in A$. From 4 of C2 and $y\eta \in F'$, we have $y' \in F'$. From the induction hypothesis, we can assume that $y' \in A^* - P$. This implies that $y' \notin P$. Thus, from 4 of C1, $y'a \notin P$, which derives $y'a \in A^* - P$. Hence, we have shown $x \in F' \Rightarrow x \in A^* - P$. Similarly, we suppose $x \notin F'$ and show that $x \notin A^* - P$ holds. If $|\eta| = 1$, then from 2(b) and 2(c) of C2, $\text{limit}(y, N) \subseteq O' = I$ must hold in $\mathcal{T}_s(M')$. Thus, in $\mathcal{T}(M)$, from 2(b) of C1, $x \in F$ holds, and it implies that $x \notin A^* - P$ holds. If $|\eta| > 1$, let $x = y\eta = y'a$ with $a \in A$. From 3 of C2 and $y\eta \notin F'$, we have $y' \notin F'$. From the induction hypothesis, we can assume that $y' \notin A^* - P$. This implies that $y' \in P$ and so $y' \in F$. Thus, from 3 of C1, $y'a \in F$, which derives $y'a \notin A^* - P$. Hence, we have shown $x \notin F' \Rightarrow x \notin A^* - P$. Q.E.D.

We say that a semimodule (O, I, N) is the *semimirror* of a module $M = (I, O, N)$, denoted by M^{sm}. From Lemma 3 and the timed version of Lemma 1, we have the following theorem.

Theorem 2 Suppose that $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$ are modules such that $I_1 \subseteq I_2$ and $O_1 \supseteq O_2$. M_1 conforms to M_2 iff $\mathcal{T}(M_1) \| \mathcal{T}_s(M_2^{sm})$ is failure free.

From this theorem, the decision procedure of conformance between timed systems can be actually implemented, if the failure trace sets of modules and semimodules are constructed differently.

4 Comparison with the previous framework

In this section, we compare the above framework with the one we proposed in [11]. The related works are presented in [12, 16]. In [11], when a set of modules are given, the safety failures and timing failures are defined among those modules. Then, the conformance is defined similarly. Let $\text{failure}(M_1, M_2)$ denote all safety and timing failures between M_1 and M_2. In order to distinguish the conformance defined in this paper and the one defined in [11], we call the latter *p-conformance*. That is, for two modules $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$, we say that M_1 *p-conforms* to M_2, if $I_1 \subseteq I_2$, $O_1 \supseteq O_2$, and for any (composable) module $E = (I_E, O_E, N_E)$ such that $I_2 \subseteq O_E$ and $O_2 \supseteq I_E$, if $\text{failure}(E, M_2)$ is empty, so is $\text{failure}(E, M_1)$.

441

According to [11], if an input u with $\mathrm{Lft}(u) = \infty$ is enabled in M_1, and M_2 has no enabled transitions, then failure(M_1, M_2) has a timing failure. On the other hand, $\mathcal{T}(M_1)||\mathcal{T}(M_2)$ is failure-free from C1. Whether a failure should exist in this case or not depends on the purpose of verification. Since we want to compare both frameworks independent of this situation, we assume below that every transition has a finite latest firing time. This does not mean that those frameworks are insufficient to handle transitions with infinite latest firing times.

Under this restriction, failure(M_1, M_2) is still not equal to the failure trace set of $\mathcal{T}(M_1)||\mathcal{T}(M_2)$. However, we can prove the following lemma.

Lemma 4 Let $T_1 = (I_1, O_1, S_1, F_1)$ and $T_2 = (I_2, O_2, S_2, F_2)$ be the timed trace structures of $M_1 = (I_1, O_1, N_1)$ and $M_2 = (I_2, O_2, N_2)$ such that $I_1 \subseteq O_2$ and $I_2 \subseteq O_2$. $T_1||T_2$ is failure-free, iff failure$(M_1, M_2) = \emptyset$.

The proof of this lemma can be found in [17]. This lemma directly derives the following theorem, which shows the equivalence between the two frameworks.

Theorem 3 M_1 conforms to M_2, iff M_1 p-conforms to M_2.

5 Conclusion

In this paper, we have proposed a framework to support trace theoretic verification of timed circuits. First, we have developed a criteria to classify timed traces of time Petri nets as successes, failures, or impossible, and then introduced the notions of semimodules and semimirrors which allow us to implement conformance checking procedures. This framework is simpler and more comprehensible than what we proposed previously, but the theorem shown in this paper shows that these two frameworks are equivalent in the final decisions.

The direct implementation of the conformance checking procedure shown in this paper is not difficult, but we know that it is easily faced with the state explosion problem. In order to overcome this problem, we are now developing a partial order reduction algorithm for conformance checking using this framework, and we will implement it in our tool VINAS-P[18] in the future.

References

[1] D. L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits.* MIT press, 1988.

[2] T. Yoneda and T. Yoshikawa. Using partial orders for trace theoretic verification of asynchronous circuits. *Proc. of Second International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 152–163, 1996.

[3] T. Yoneda and H. Ryu. Timed trace theoretic verification using partial order reduction. *Proc. of Fifth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 108–121, 1999.

[4] K. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, and M. Roncken. An asynchronous instruction length decoder. *IEEE Journal of Solid-State Circuits*, 35(2):217–228, February 2001.

[5] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. *Proc. of 5th IEEE LICS*, 1990.

[6] R. Alur and T. A. Henzinger. A really temporal logic. *Proc. of 30th IEEE FOCS*, 1989.

[7] T. Yoneda and H. Schlingloff. Efficient verification of parallel real–time systems. *Formal Method in System Design*, pages 187–215, 1997.

[8] M. Minea. *Partial order reduction for verification of timed systems.* PhD thesis, Carnegie Mellon University, 1999.

[9] R. Alur and D. Dill. The theory of timed automata. *LNCS 443 (17th ICALP)*, pages 322–335, 1990.

[10] T. Yoneda, B. Zhou, and H. Schlingloff. Verification of bounded delay asynchronous circuits with timed traces. *LNCS 1548 AMAST'98*, pages 59–73, 1999.

[11] B. Zhou and T. Yoneda. Verification of asynchronous circuits with bounded delay model. *IEICE Trans. (in Japanese) Vol.J82-D-I, No.7*, pages 819–833, 1999.

[12] B. Zhou, T. Yoneda, and H. Schlingloff. Conformance and mirroring for timed asynchronous circuits. *Proc. of ASP-DAC'01*, pages 341–346, 2001.

[13] T. Rokicki. *Representing and modeling circuits.* PhD thesis, Stanford University, 1993.

[14] T. Rokicki and C. Myers. Automatic verification of timed circuits. *LNCS 818 Computer Aided Verification*, pages 468–480, 1994.

[15] J. R. Burch. *Trace Algebra for Automatic Verification of Real-Time Concurrent Systems.* PhD thesis, Carnegie Mellon University, 1992.

[16] B. Zhou, T. Yoneda, and H. Schlingloff. Conformance and mirroring for timed asynchronous circuits. *TIT CS Technical Report*, TR01-0001, 2001.

[17] B. Zhou, T. Yoneda, and C. Myers. Framework of timed trace theoretic verification revisited. *TIT CS Technical Report*, TR01-0015, 2001.

[18] T. Yoneda. VINAS-P: A tool for trace theoretic verification of timed asynchronous circuits. *LNCS 1855 Computer Aided Verification*, pages 572–575, 2000.