Formal Verification of Timed Systems using Cones and Foci

Wan Fokkink [1]

Department of Software Engineering, CWI
PO Box 94079, 1090 GB Amsterdam, The Netherlands
Department of Theoretical Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

Jun Pang [2]

Department of Software Engineering, CWI
PO Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

The cones and foci verification method from Groote and Springintveld [12] was extended to timed systems by van der Zwaag [24]. We present an extension of this cones and foci method for timed systems, which can cope with infinite τ-sequences. We prove soundness of our approach and give small verification examples.

Key words: Timed μCRL, timed transition systems, verification techniques.

1 Introduction

Process algebras such as CSP [14,15], CCS [18,19] and ACP [4,3,5], for which timed extensions exist (e.g., [20,21,1], respectively), in principle offer an excellent platform for symbolic verification of timed systems. However, more than ten years after the foundations of most timed process algebras were laid, verifications performed within these formalisms require an enormous effort, and tend to be restricted to relatively small case studies (see, e.g., [23]).

μCRL [9] combines the process algebra ACP with equational abstract data types. Groote *et al.* [8,11] introduced a timed extension of μCRL. A labeled transition system is associated to each timed μCRL specification. These specifications are considered modulo *timed strong bisimulation*, which is based on

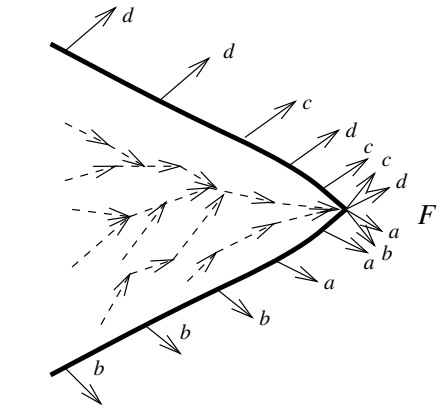
[1] Email: wan@cwi.nl; wanf@cs.vu.nl
[2] Email: pangjun@cwi.nl

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

an equivalence notion put forward in [16,17]. *Timed linear process equations* [22] (timed LPEs) constitute a restricted class of timed μCRL specifications that do not contain parallel and renaming operators. In a timed LPE, the states of the associated labeled transition system are data objects. Usenko [22] presented a transformation algorithm from an important subset of timed μCRL to timed LPEs; this transformation preserves timed branching bisimulation.

So far, verifications in timed μCRL have also been limited to relatively small, manual efforts (see, e.g., [13]). Recently, van der Zwaag [24] introduced the *cones and foci* verification method for timed μCRL. This work is based on an earlier verification method by Groote and Springintveld [12] for untimed μCRL, which played a vital role in the verification of a considerable number of real-life protocols and distributed algorithms, often with the support of a theorem prover; see [10] for an overview. The cones and foci method can be used to prove both safety and liveness of a system.

The main idea of the cones and foci method is that in the implementation of a system, hidden τ-actions usually progress inertly towards a state in which no τ can be executed; such a state is declared to be a *focus point*. The *cone* of a focus point consists of the states that can reach this focus point by the execution of a string of τ's. In the absence of infinite τ-sequences, each state belongs to a cone. This core idea is depicted below. Note that the external actions at the edge of the depicted cone can also be executed in the ultimate focus point F; this is essential for the soundness of the cones and foci method, as otherwise not all τ's in the cone would be inert.



Let the implementation of a system and its desired external behavior both be given as a timed LPE. In the cones and foci method, a *state mapping* ϕ relates each state of the implementation to a state of the desired external behavior. Van der Zwaag [24] formulated *matching criteria*, consisting of relations between data objects, which ensure that states s and $\phi(s)$ are timed branching bisimilar. Thus, the cones and foci method rephrases the question

whether two timed μCRL specifications are timed branching bisimilar in terms of proof obligations on relations between data objects. These proof obligations can then be proved by means of algebraic calculations, in general with the help of invariants that need to be proved separately.

In untimed μCRL, Groote and Springintveld [12] refined their cones and foci method to cope with infinite τ-sequences. They allow the user to indicate whether or not a τ is *progressing*. Only progressing τ's are abstracted away. Finally, a special *fair abstraction rule* [2] can be used to try and eliminate the remaining (non-progressing) τ's. In [7], we proposed an adaptation of the cones and foci method for untimed μCRL in which this cumbersome treatment of infinite τ-sequences is no longer necessary. This adaptation was conceived during the verification of a sliding window protocol [6], where it simplified matters considerably. It allows the user to freely assign which states are focus points (instead of prescribing that they are the states in which no progressing τ's can be performed), as long as each state is in the cone of a focus point. Infinite τ-sequences are allowed. No distinction between progressing and non-progressing τ's is needed, and τ-loops are eliminated without having to resort to a fair abstraction rule.

Van der Zwaag [24] excluded infinite τ-sequences. Thus, we extend the applicability of the cones and foci method from [24]. As in [7], the user can freely assign which states are focus points, as long as each state is in the cone of a focus point. Moreover, infinite τ-sequences are allowed. With respect to the untimed cones and foci method in [7], our timed cones and foci method contains two extra matching criteria, to deal with *timed deadlocks* (as Van der Zwaag did in [24]). We prove soundness of our approach and give two small verification examples. The first example, which originates from [12], contains an infinite τ-sequence, so that it falls outside the scope of [24]. The second example deals with the main case study of [24].

2 Preliminaries

2.1 Timed μCRL

μCRL [9] is a language for specifying distributed systems and protocols in an algebraic style. It is based on process algebra extended with equational abstract data types. In a μCRL specification, one part specifies the data types, while a second part specifies the process behavior. We do not describe the treatment of data types in μCRL in detail. For our purpose it is sufficient that processes can be parametrized with data. We assume the data sort of booleans *Bool* with constant T and F, and the usual connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$. For a boolean b, we abbreviate $b = \mathsf{T}$ to b and $b = \mathsf{F}$ to $\neg b$.

The μCRL specification of a process is constructed from action names, recursion variables and process algebraic operators. Actions and recursion variables carry zero or more data parameters. There are two predefined actions

in μCRL: δ represents deadlock, and τ a hidden (internal) action. These two actions never carry data parameters. In timed μCRL [8], each action happens at a specific time. The time domain *Time* is a nonempty, totally ordered set with a least element 0. Each action is supplied with a time stamp, denoting the time at which it can be executed. Actions can be executed at the same time consecutively, and their execution does not consume any time.

Processes are represented by process terms, which describe the order in which the actions from a set *Act* may happen. A process term consists of action names (with time stamps) and recursion variables, combined by process algebraic operators. $p{\cdot}q$ denotes sequential composition and $p{+}q$ non-deterministic choice, summation $\sum_{d:D} p(d)$ provides the possibly infinite choice over a data type D, and the conditional construct $p \lhd b \rhd q$ with b a data term of sort *Bool* behaves as p if b and as q if $\neg b$. Parallel composition $p \parallel q$ interleaves the actions of p and q; moreover, actions from p and q may also synchronize to a communication action, when this is explicitly allowed by a predefined communication function. Two actions can only synchronize if they occur at the same time, and if their data parameters are semantically the same. Encapsulation $\partial_H(p)$, which renames all occurrences in p of actions from the set H into δ, can be used to force actions into communication. Finally, hiding $\tau_I(p)$ renames all occurrences in p of actions from the set I into τ.

2.2 Timed transition systems

Let *Lab* denote the set of actions from $Act \cup \{\tau\}$ with all possible data parameters.

Definition 2.1 A *timed transition system [11]* is a tuple (S, T, U), where

- S is a set of states, including a special state $\surd$ to represent successful termination;
- $T \subseteq S \times Lab \times Time \times S$ is a set of transitions;
- $U \subseteq S \times Time$ is a *delay relation*, which satisfies:
  - if $T(s, \ell, u, r)$ $(s \xrightarrow{\ell}_u s')$, then $U(s, u)$;
  - if $u < v$ and $U(s, v)$, then $U(s, u)$.

We use $T(s, \ell, u, r)$ (or $U(s, u)$) to denote that (s, ℓ, u, r) (or (s, u)) is an element of T (or U). Transitions (s, ℓ, u, s') express that state s evolves into state s' by the execution of label ℓ at time u. If $U(s, u)$, then state s can let time pass until time u. A transition (s, ℓ, u, s') is denoted by $s \xrightarrow{\ell}_u s'$. For any $u{:}Time$, we define the generalized τ-step relation $\Longrightarrow_u$ as the reflexive transitive closure of $\xrightarrow{\tau}_u$. We assume a special termination predicate $\downarrow$, which holds only in the successful termination state $\surd$.

Definition 2.2 Assume a timed transition system (S, T, U). A symmetric binary relation $B \subseteq S \times Time \times S$ is a *timed branching bisimulation* [24] such that if sB_ut then all of the following hold:

- if $s \xrightarrow{\ell}_u s'$, then
  - either $\ell = \tau$ and $s' B_u t$,
  - or $t \Rightarrow_u \hat{t}$ such that $s B_u \hat{t}$ and $\hat{t} \xrightarrow{\ell}_u t'$ with $s' B_u t'$;
- if $s \downarrow$, then $t \Rightarrow_u t'$ with $t' \downarrow$;
- if $u < v$ and $U(s, v)$, then for some $n \geq 0$, there are $t_0, \cdots t_n \in S$ with $t = t_0$, $u_0 < \cdots < u_n{:}Time$ with $u \leq u_0$ and $v = u_n$, such that $t_i \Rightarrow_{u_i} t_{i+1}$ for $i < n$, $s B_{u_i} t_i$ for $i \leq n$ and $s B_{u_i} t_{i+1}$ for $i < n$, $U(t_n, v)$.

Two states s and t are *timed branching bisimilar at* u, if there is a timed branching bisimulation B with $s B_u t$. States s and t are *timed branching bisimilar*, denoted by $s \underline{\leftrightarrow}_{tb} t$, if they are timed branching bisimilar at any time $u{:}Time$.

By the first clause in the definition of a timed branching bisimulation, we treat the behavior of a state at some point in time like untimed behavior. The second clause deals with successful termination. By the third clause, we demand that time passing in a state s is matched by a related state r with a "τ-path" where all intermediate states are related to s at the appropriate times.

Example 2.3 Consider the following two timed transition system: $s_0 \xrightarrow{a}_2 s_1 \xrightarrow{b}_1 s_2$ and $t_0 \xrightarrow{a}_2 t_1$. We have $s_0 \underline{\leftrightarrow}_{tb} t_0$.

Example 2.4 Consider the following two timed transition system: $s'_0 \xrightarrow{a}_1 s'_1 \xrightarrow{\tau}_2 s'_2 \xrightarrow{b}_3 s'_3$ and $t'_0 \xrightarrow{a}_1 t'_1 \xrightarrow{b}_3 t'_2$. We have $s'_0 \underline{\leftrightarrow}_{tb} t'_0$.

2.3 Timed linear process equations

Timed LPEs constitute a restricted class of timed μCRL specifications. They are one-line timed μCRL specifications consisting of actions, summations, sequential compositions and conditional constructs. In particular, they do not contain any parallel operators, encapsulations or hidings. Usenko [22] showed how an importants class of timed μCRL specifications can be transformed into timed LPEs.

For the sake of presentation, Groote and Springintveld [12] did not consider successful termination. They noted that their cones and foci method extends to a setting with successful termination in a straighforward fashion. For the same reason, we do not consider successful termination here. [3]

[3] To cover timed μCRL specifications with successful termination, timed LPEs in Definition 2.5 should also include a summand

$$\sum_{a \in Act \cup \{\tau\}} \sum_{e:E} \sum_{u:Time} h_a(d, e, u) \rhd a(f_a(d, e, u)).$$

Definition 2.5 A *timed linear process equation* $X(d{:}D)$ is a timed μCRL specification of the form:

$$\sum_{a\in Act\cup\{\tau,\delta\}} \sum_{e:E} \sum_{u:Time} h_a(d,e,u) \rhd a(f_a(d,e,u)); d := g_a(d,e,u)$$

where $f_a : D \times E \times Time \to D$, $g_a : D \times E \times Time \to D$ and $h_a : D \times E \times Time \to Bool$ for each $a \in Act \cup \{\tau, \delta\}$. For presentation convenience, here we present τ and δ with data parameters.

A timed LPE expresses that state d can perform $a(f_a(d,e,u))$ at time u, to end up in state $g_a(d,e,u)$, under the condition that $h_a(d,e,u)$ is true. The data type E gives timed LPEs a more general form, as not only the data parameter $d{:}D$ and $t{:}Time$ but also the data parameter $e{:}E$ can influence the parameter of action a, the condition h_a and the resulting state g_a.

With $X(d{:}D)$ we associate a timed transition system as follows.

Definition 2.6 The timed transition system $tts(X)$ for a timed LPE $X(d{:}D)$ is defined as (D, T, U), where T and U are the smallest sets such that, for all $d{:}D$, $a \in Act \cup \{\tau, \delta\}$, $e{:}E$ and $u, v{:}Time$,

- if $h_a(d,e,u)$ and $a \neq \delta$, then $T(d, a(f_a(d,e,u)), u, g_a(d,e,u))$;
- if $h_a(d,e,u)$ and $v \leq u$, then $U(d,v)$.

The relation h_δ may be used to specify the presence of so-called timed deadlocks. In the untimed case, it is not necessary to specify deadlocks explicitly. Here, timed deadlocks determine process behavior as follows: if $h_\delta(d,e,u)$, then $U(d,u)$, meaning that in state d time may pass at least until time u. Such a state d cannot be related to a state that cannot let time pass until u.

Definition 2.7 The *delay condition* $DC_X : D \times Time \to Bool$ is defined on a timed LPE $X(d{:}D)$ if and only if $h_a(d,e,v)$ and $u \leq v$ for some $a \in Act \cup \{\tau, \delta\}$, $e{:}E$ and $u, v{:}Time$.

Lemma 2.8 *Given a timed LPE* $X(d{:}D)$ *and* $u{:}Time$, $DC_X(d,u) \Leftrightarrow U(d,u)$.

This lemma explains that $DC_X(d,u)$ if and only if $U(d,u)$ in $tts(X)$, meaning that if $DC_X(d,u)$, then in state d time may pass at least until time u.

Definition 2.9 A mapping $\mathcal{I} : D \times Time \to Bool$ is an *invariant* of timed LPE $X(d{:}D)$, written as in Definition 2.5, if whenever $\mathcal{I}(d,u)$ and $h_a(d,e,v)$, for any $u'{:}Time$ such that $u \leq u' \leq v$, then $\mathcal{I}(d,u')$ and, if $a \neq \delta$, $\mathcal{I}(g_a(d,e,v),v)$.

If $\mathcal{I}$ is an invariant of P with $\mathcal{I}(d,u)$, then it holds by definition of $tts(P)$ that, whenever $d \xrightarrow{a}_u d'$, then also $\mathcal{I}(d',u)$, and whenever $U(d,v)$ and $u < v$, then also $\mathcal{I}(d,v)$.

3 Cones and foci

In this section, we present our version of the cones and foci method for timed transition systems. Suppose that we have a timed LPE $X(d{:}D)$ (containing τ) that specifies the implementation of a system, and a timed LPE $Y(d'{:}D')$ (without occurrences of τ) that specifies the desired external behavior of this system. Furthermore, assume an invariant $\mathcal{I} : D \times \mathit{Time} \to \mathit{Bool}$ characterizing the reachable states of the implementation. We want to prove that the implementation exhibits the desired external behavior.

In the cones and foci method, a *state mapping* $\phi : D \to D'$ relates each state of the implementation X to a state of the desired external behavior Y. Furthermore, for each $u{:}\mathit{Time}$ some states in D are designated to be *focus points* at time u. If a number of *matching criteria* for $d{:}D$ are fulfilled, consisting of relations between data objects, then the states d and $\phi(d)$ are guaranteed to be timed branching bisimilar.

In the timed case, the intuition behind the cones and focus points is obscured by the timing of transitions, but still is the guiding intuition. We begin with defining the predicate FC. Next we express the matching criteria relative to a state at a time element.

Definition 3.1 A *focus condition* is a mapping $FC : D \times \mathit{Time} \to \mathit{Bool}$. If $FC(d, u)$, then d is called a *focus point* at time u.

Definition 3.1 makes that our method differs from [24], in which a focus point is defined on a time interval.

Definition 3.2 A *state mapping* is of the form $\phi : D \to D'$.

Definition 3.3 Let the timed LPE $X(d{:}D)$ be of the form

$$\sum_{a \in Act \cup \{\tau, \delta\}} \sum_{e:E} \sum_{u:\mathit{Time}} h_a(d, e, u) \rhd a(f_a(d, e, u)); d := g_a(d, e, u)$$

Furthermore, let the timed LPE $Y(d'{:}D')$ be of the form

$$\sum_{a \in Act \cup \{\delta\}} \sum_{e:E} \sum_{u:\mathit{Time}} h'_a(d', e, u) \rhd a(f'_a(d', e, u)); d' := g_a(d', e, u)$$

Let $\phi : D \to D'$. We say that ϕ satisfies the *matching criteria* for a state d and a time element u if for all $a \in \mathit{Act}$, $e{:}E$ and $v{:}\mathit{Time}$ the following conditions hold.

I $h_\tau(d, e, u) \Rightarrow \phi(d) = \phi(g_\tau(d, e, u)) \wedge DC_Y(\phi(d), u)$;

II $h_a(d, e, u) \Rightarrow h'_a(\phi(d), e, u)$;

III $FC(d, u) \wedge h'_a(\phi(d), e, u) \Rightarrow h_a(d, e, u)$;

IV $h_a(d, e, u) \Rightarrow f_a(d, e, u) = f'_a(\phi(d), e, u)$;

V $h_a(d, e, u) \Rightarrow \phi(g_a(d, e, u)) = g'_a(\phi(d), e, u)$;

VI $h_\delta(d,e,u) \Rightarrow DC_Y(\phi(d),u)$;

VII $v > u \wedge FC(d,v) \wedge h'_\delta(\phi(d),e,v) \Rightarrow DC_X(d,v)$.

Matching criterion I requires that if state d can do a τ-step at time u, then d and the resulting state have the same ϕ-image, and this ϕ-mapping must be able to let time pass until u. Matching criteria II,IV and V express that each visible transition of state d at time u can be simulated by $\phi(d)$ at the same time. Matching criterion III says that at time u, in a focus point d of the timed LPE X, a visible transition can be performed if it is enabled in timed LPE Y at the same time. The first five criteria are adaptations of the criteria for the untimed case. The last two had to be added in order to deal with explicit timed deadlocks, that do not exist in the setting without time. Matching criterion VI and VII express that if d has a timed deadlock at u, then its ϕ-mapping must be able to let time pass until u, and vice versa.

Theorem 3.4 *Assume timed LPEs $X(d{:}D)$ and $Y(d'{:}D')$ as in the Definition 2.5. Let $\mathcal{I} : D \times Time \to Bool$ be an invariant for X. Suppose that for all $d{:}D$ and $u{:}Time$ with $\mathcal{I}(d,u)$:*

(i) *$\phi : D \to D'$ satisfies the matching criteria, and*

(ii) *if $h'_a(\phi(d),e,v)$, for some $a \in Act \cup \{\delta\}$, $e{:}E$, and $u \leq v$, then for some $n \geq 0$, there are $t_0,\cdots,t_n{:}D$ with $d = t_0$, and elements $u_0 < \cdots < u_n{:}Time$ with $u \leq u_0$ and $v = u_n$ such that $t_i \Rightarrow_{u_i} t_{i+1}$, for $i < n$, and $FC(t_n,u_n)$.*

Then for any $d_0{:}D$ and $u_0{:}Time$ with $\mathcal{I}(d_0,u_0)$, it holds that d_0 and $\phi(d_0)$ are timed branching bisimilar at u_0.

Note that we can consider the untimed cones and foci method [7] as a special case of this timed one, with all actions at time 0. Condition 2 states the reachability of a focus point at time v from d at u by a sequence of τ-steps.

Proof. We assume without loss of generality that D and D' are disjoint. Define $B \subseteq (D \cup D') \times Time \times (D \cup D')$ by $dB_u\phi(d)$ and $\phi(d)B_u d$ if and only if $\mathcal{I}(d,u)$. B is symmetric. We show that B is a timed branching bisimulation relation.

- Action step:

  Let sB_ut and $s \xrightarrow{\ell}_u s'$. This step must be matched by t. First, consider the case where $\phi(s) = t$. By definition of B, $\mathcal{I}(s,u)$.

  · If $\ell = \tau$, then $h_\tau(s,e,u)$ and $s' = g_\tau(s,e,u)$, for some $e{:}E$. By matching criterion I, $\phi(g_\tau(s,e,u)) = t$. Moreover, $\mathcal{I}(s,u)$ and $h_\tau(s,e,u)$ together imply $\mathcal{I}(g_\tau(s,e,u),u)$. Hence, $g_\tau(s,e,u)B_ut$.

  · If $\ell \neq \tau$, then $h_a(s,e,u)$, $s' = g_a(s,e,u)$ and $\ell = a(f_a(s,e,u))$ for some $a \in Act$ and $e{:}E$. By matching criteria II and IV, $h'_a(t,e,u)$ and $f_a(s,e,u) = f'_a(t,e,u)$. Hence, $t \xrightarrow{a(f_a(s,e,u))}_u g'_a(t,e,u)$. Moreover, $\mathcal{I}(s,u)$ and $h_a(s,e,u)$ imply $\mathcal{I}(g_a(s,e,u),u)$, and matching criterion V yields $\phi(g_a(s,e,u)) = g'_a(t,e,u)$, so $g_a(s,e,u)B_ug'_a(t,e,u)$.

Next, consider the case where $s = \phi(t)$. By definition of B, $\mathcal{I}(t,u)$. Since $s \xrightarrow{\ell}_u s'$, it holds that $h'_a(s,e,u)$, $s' = f'_a(s,e,u)$ and $\ell = a(f'_a(s,e,u))$ for some $a \in Act$ and $e{:}E$. By $h'_a(\phi(t),e,u)$, $\mathcal{I}(t,u)$ and condition ii (with $n=0$ and $u = u_0$), there is a $\hat{t}{:}D$ such that $t \Longrightarrow_u \hat{t}$ and $FC(\hat{t},u)$. Invariant $\mathcal{I}$ and the matching criteria hold for all states on this τ-path at time u. Repeatedly applying matching criterion I we get $\phi(\hat{t}) = \phi(t) = s$, so $sB_u\hat{t}$. Furthermore, matching criterion III, $FC(\hat{t},u)$ and $h'_a(s,e,u)$ yield $h_a(\hat{t},e,u)$. Then by matching criterion IV, $f_a(\hat{t},e,u) = f'_a(s,e,u)$, so $\hat{t} \xrightarrow{a(f'_a(s,e,u))}_u g_a(\hat{t},e,u)$. Moreover, $\mathcal{I}(\hat{t},u)$ and $h_a(\hat{t},e,u)$ together imply $\mathcal{I}(g_a(\hat{t},e,u),u)$, and matching criterion V yields $\phi(g_a(\hat{t},e,u)) = g'_a(s,e,u)$, so $g'_a(s,e,u)B_u g_a(\hat{t},e,u)$.

- Delay behavior:
  Suppose that $u < v$ and $U(s,v)$. This delay behavior must be matched by t. First, consider the case where $\phi(s) = t$. Since $U(s,v)$, $h_a(s,e,v')$ for some $a \in Act \cup \{\tau,\delta\}$, $e{:}E$ and $v'{:}Time$ with $v' \geq v > u$. By definition of B, $\mathcal{I}(s,u)$. So $\mathcal{I}(s,v)$ and $\mathcal{I}(s,v')$. By definition of B, sB_vt.
  · If $a = \tau$, matching criterion I yields $DC_Y(t,v')$.
  · If $a = \delta$, matching criterion VI yields $DC_Y(t,v')$.
  · Otherwise, $a \in Act$, matching criterion II yields $h'_a(t,e,v')$.
  So $DC_Y(t,v')$ or $h'_a(t,e,v')$. By Lemma 2.8, $U(t,v')$, and hence $U(t,v)$.
  Next, consider the case where $s = \phi(t)$. By definition of B, $\mathcal{I}(t,u)$. If $U(t,v)$, since $u < v$, we have $\mathcal{I}(t,v)$ and sB_vt. Otherwise, since $U(s,v)$, $h'_a(s,e,v')$ for some $a \in Act \cup \{\delta\}$, $e{:}E$ and $v'{:}Time$ with $v' \geq v$. By condition ii, for some $n \geq 0$, there are $t_0,\cdots,t_n{:}D$ with $t = t_0$, and $u_0 < \cdots < u_n{:}Time$ with $u \leq u_0$, $v' = u_n$ such that $t_i \Longrightarrow_{u_i} t_{i+1}$, for $i < n$, and $FC(t_n,u_n)$. Since $\mathcal{I}(t_i,u_i)$ for $i < n$, and $\mathcal{I}(t_n,u_{n-1})$, by repeatedly applying matching criterion I we get $\phi(t_i) = \phi(t) = s$ for $i \leq n$. By definition of B, $sB_{u_i}t_i$ for $i \leq n$ and $sB_{u_i}t_{i+1}$ for $i < n$.
  · If $a = \delta$, then matching criterion VII together with $FC(t_n,u_n)$ (note that $u_n = v'$) and $h'_\delta(s,e,v')$ yields $DC_X(t_n,v')$. By Lemma 2.8, $U(t_n,v')$, and hence $U(t_n,v)$. Note that since $U(t_n,v')$ we have $\mathcal{I}(t_n,v)$, which implies sB_vt_n.
  · Otherwise, $a \in Act$. Matching criterion III together with $FC(t_n,u_n)$ and $h'_a(s,e,v')$ yields $h_a(t_n,e,v')$ (note that $u_n = v'$). Hence, $DC_X(t_n,v')$. Then $U(t_n,v')$, and hence $U(t_n,v)$. Note that since $h_a(t_n,e,v')$, we have $\mathcal{I}(t_n,v')$ and $\mathcal{I}(t_n,v)$, which implies sB_vt_n.

Concluding, B is a timed branching bisimulation relation. So $\mathcal{I}(d_0,u_0)$ implies that d_0 and $\phi(d_0)$ are timed branching bisimilar at time u_0.

□

4 Examples

4.1 Tossing a coin

In [24], van der Zwaag requires that the timed transition system is convergent at any time u. [4] In this section, we give a small example, where our method can play a role for the verification task, while [24] cannot. (Or at least, it would require some notion of progressing τ's as in [12], which was not considered by van der Zwaag.)

We take the example of tossing a coin from [12] and add time to it. $Sides = \{head, tail\}$ denotes the sides of the coin. We give a timed LPE (the implementation) to describe a person who tosses a coin (modeled by a τ) at any time. When *head* turns up, the person gives a smile (modeled by an external action sm) and tosses the coin again. When *tail* turns up the person can toss again. We give another timed LPE (the desired external behavior) to describe a person who can give a smile at any time.

4.1.1 The implementation

Let $Act = \{sm\}$ and $D = (Sides \times Time) \cup \{\varepsilon\}$ to denote the state space of the implementation.

$$I_{coin}(d{:}D) := \sum_{e \in Sides} \sum_{u:Time} h_{sm}(d,e,u) \rhd sm; d := g_{sm}(d,e,u) + \sum_{e \in Sides} \sum_{u:Time} h_{\tau}(d,e,u) \rhd \tau; d := g_{\tau}(d,e,u)$$

with

$$h_{sm}(d,e,u) \Leftrightarrow d = (head, u)$$
$$g_{sm}(d,e,u) = \varepsilon$$
$$h_{\tau}(d,e,u) \Leftrightarrow d = \varepsilon$$
$$g_{\tau}(d,e,u) = \begin{cases} \varepsilon & \text{if } e = tail \\ (e,u) & \text{otherwise, } (\Leftrightarrow e = head) \end{cases}$$

We simplify the representation, since there is only one external action sm and it contains no data parameter. It is clear that the implementation is not convergent at any time.

4.1.2 The desired external behavior

The specification only contains one state $D' = \{\varrho\}$. Its timed LPE $S_{coin}(d'{:}D')$ is defined as follows.

$$S_{coin}(d'{:}D') := \sum_{u:Time} \mathsf{T} \rhd sm; d' := d'$$

[4] In [24], a state s is *convergent at time* u in a timed transition system, if that system has no infinite sequence $s_0 u_0 s_1 u_1 s_2 u_2 \cdots$ such that $s = s_0$, $u \leq u_0$, and, for all $i \geq 0$, $s_i \xrightarrow{\tau}_{u_i} s_{i+1}$ and $u_i \leq u_{i+1}$.

4.1.3 Verification

For any $d{:}D$, the state mapping $\phi : D \to D'$ is defined by $\phi(d) = \varrho$. The invariant of the implementation is defined as $\mathcal{I}(d, v)$: for any $v{:}Time$, $d = (s, u) \Rightarrow s = head$. It is straightforward to check that $\mathcal{I}$ is indeed an invariant of the implementation.

Definition 4.1 For all $u{:}Time$, the focus condition $FC(d, u)$ for $I_{coin}(d{:}D)$ is defined by $d = (head, u)$.

Lemma 4.2 *For each $d{:}D$ and $u{:}Time$, together with $\mathcal{I}(d, u)$, if $h'_{sm}(\phi(d), v)$ and $u \leq v$, then for some $n \geq 0$, there are $t_0, \cdots, t_n{:}D$ with $d = t_0$, and elements $u_0 < \cdots < u_n{:}Time$ with $u \leq u_0$ and $v = u_n$ such that $t_i \Longrightarrow_{u_i} t_{i+1}$, for $i < n$, and $FC(t_n, u_n)$.*

Proof. If $\neg FC(d, u)$, by definition of FC, $d = \varepsilon$. Given a $v{:}Time$, $u \leq v$, and we have $\mathcal{I}(d, v)$. Since $h_\tau(d, head, v)$ holds, state d can perform a τ to $(head, v)$, and $\mathcal{I}((head, v), v)$. By definition of FC, $FC((head, v), v)$ holds.

□

Lemma 4.3 $I_{coin}(\varepsilon) \leftrightarrow_{tb} S_{coin}(\phi(\varepsilon))$.

Proof. Take any d and u such that $\mathcal{I}(d, u)$ and $\phi(d) = \varrho$; we show $dB_u\phi(d)$. The criteria VI and VII hold trivially, since $h_\delta = \emptyset$. The other five criteria are also trivial. By Theorem 3.4 and Lemma 4.2, $dB_u\phi(d)$. Hence, $I_{coin}(\varepsilon) \leftrightarrow_{tb} S_{coin}(\phi(\varepsilon))$.

□

4.2 Two serial buffers

Van der Zwaag [24] proved correctness of the two serial buffers as an application of his cones and foci method for timed transition systems. Here we redo his correctness proof using our version of the cones and foci method for timed transition systems.

First, we give a timed LPE for a buffer with capacity one. Let $Act = \{r, s\}$, M the set of messages, action $s(m)$ models the sending of message m, and $r(m)$ models the receiving of message m. Between the reading and the sending of a message, there is a fixed time delay Δ. Let $D = \{\varepsilon\} \cup (M \times Time)$. The timed LPE $Buffer(d{:}D)$ is defined as follows.

$$Buffer(d{:}D) := \sum_{m:M} \sum_{u:Time} h_r(d, m, u) \rhd r(f_r(d, m, u)); d := g_r(d, m, u)$$
$$+ \sum_{u:Time} h_s(d, u) \rhd s(f_s(d, u)); d := g_s(d, u)$$

with

$$
\begin{aligned}
h_r(d,m,u) &\Leftrightarrow d = \varepsilon \\
f_r(\varepsilon,m,u) &= m \\
g_r(d,m,u) &= (m,u) \\
h_s(d,u) &\Leftrightarrow d = (m, u-\Delta) \\
f_s((m,u-\Delta),u) &= m \\
g_s(d,u) &= \varepsilon.
\end{aligned}
$$

A buffer in state ε is empty and ready to read any message at any time, since $h_r(\varepsilon, m, u)$ holds for all $m \in M$ and $u{:}\mathit{Time}$. A buffer in a state (m, v) has read message m at time v, and will send the message at time $v + \Delta$.

We look at the parallel operation of two serial buffers; one buffer reads a message from the environment at time u. It sends the message to the other buffer at time $u+\Delta$. The communication between the buffers occurs along an internal port and is modeled by a τ. After the communication of the message, the first buffer returns to the empty state. The second buffer outputs the message at time $u + 2\Delta$.

4.2.1 The implementation

To simplify the example (for the sake of its presentation), we assume that the set M of messages is a singleton $\{m\}$; we abstract from the identity of messages. Consequently, we can represent $\{\varepsilon\} \cup (M \times \mathit{Time})$ (the state space of a single buffer) by the set $\mathit{Time}_\varepsilon = \mathit{Time} \cup \{\varepsilon\}$. We represent the states of two serial buffers by (d_1, d_2). The implementation is a timed LPE $I_{TB}((d_1, d_2){:}D)$ with $D = \mathit{Time}_\varepsilon \times \mathit{Time}_\varepsilon$.

$$
\begin{aligned}
I_{TB}((d_1,d_2){:}D) := &\sum_{u:\,\mathit{Time}} h_r((d_1,d_2),u) \rhd r; (d_1,d_2) := g_r((d_1,d_2),u) \\
&+ \sum_{u:\,\mathit{Time}} h_s((d_1,d_2),u) \rhd s; (d_1,d_2) := g_s((d_1,d_2),u) \\
&+ \sum_{u:\,\mathit{Time}} h_\tau((d_1,d_2),u) \rhd \tau; (d_1,d_2) := g_\tau((d_1,d_2),u)
\end{aligned}
$$

with

$$
\begin{aligned}
&h_r((d_1,d_2),u) \Leftrightarrow d_1 = \varepsilon \wedge \beta_2(u) \\
&g_r((d_1,d_2),u) = (u, d_2) \\
&h_s((d_1,d_2),u) \Leftrightarrow d_2 = u - \Delta \wedge \beta_1(u) \\
&g_s((d_1,d_2),u) = (d_1, \varepsilon) \\
&h_\tau((d_1,d_2),u) \Leftrightarrow d_1 = u - \Delta \wedge d_2 = \varepsilon \\
&g_\tau((d_1,d_2),u) = (\varepsilon, u).
\end{aligned}
$$

Since there is only one message, we do not write the second function argument "e", and use a to abbreviate $a(m)$. The conditions $\beta_i(u)$, with $i \in \{1, 2\}$, abbreviate $(d_i = \varepsilon \vee u \leq d_i + \Delta)$.

4.2.2 The desired external behavior

The specification of the two serial buffers has the same state space as the implementation, but the roles of the constituents of states are different. In a state (d_f, d_s), d_f is the time the *first* contained message was received, and d_s is the time the second contained message was received. If the system is empty, then $d_f = d_s = \varepsilon$. An invariant of the specification is that $d_f = \varepsilon \Rightarrow d_s = \varepsilon$. Its timed LPE $S_{TB}((d_f, d_s) : D)$ is defined as follows.

$$\begin{aligned} S_{TB}((d_f, d_s){:}D) := &\sum_{u:Time} h'_r((d_f, d_s), u) \rhd r; (d_f, d_s) := g'_r((d_f, d_s), u) \\ &+ \sum_{u:Time} h'_s((d_f, d_s), u) \rhd s; (d_f, d_s) := g'_s((d_f, d_s), u) \end{aligned}$$

with

$$\begin{aligned} &h'_r((d_f, d_s), u) \Leftrightarrow d_f \neq \varepsilon \Rightarrow (d_s = \varepsilon \wedge d_f + \Delta \leq u \leq d_f + 2\Delta) \\ &g'_r((d_f, d_s), u) = \begin{cases} (u, \varepsilon) & \text{if } d_f = \varepsilon \\ (d_f, u) & \text{otherwise} \end{cases} \\ &h'_s((d_f, d_s), u) \Leftrightarrow d_f = u - 2\Delta \\ &g'_s((d_f, d_s), u) = (d_s, \varepsilon). \end{aligned}$$

4.2.3 Verification

We take the definition of the state mapping and invariants from [24]. The state mapping $\phi : D \to D$ is defined by

$$\phi((d_1, d_2)) = \begin{cases} (d_1, d_2) & \text{if } d_2 = \varepsilon, \\ (d_2 - \Delta, d_1) & \text{otherwise.} \end{cases}$$

The invariant of the implementation is defined as follows. Let $\mathcal{I}((d_1, d_2), u)$ be the conjunction of $\mathcal{I}_1 : d_1 \neq \varepsilon \Rightarrow u \leq d_1 + \Delta$, $\mathcal{I}_2 : d_2 \neq \varepsilon \Rightarrow d_2 \leq u$, and $\mathcal{I}_3 : d_1 \neq \varepsilon \Rightarrow (d_2 \neq \varepsilon \Rightarrow d_2 \leq d_1)$. It is straightforward to check that $\mathcal{I}$ is indeed an invariant of the implementation.

The focus condition for $I_{TB}((d_1, d_2){:}D)$ is obtained by taking the disjunction of the summands that deal with an action in Act.

Definition 4.4 For all $u{:}Time$, the focus condition for $I_{TB}((d_1, d_2){:}D)$ is

$$FC((d_1, d_2), u) \Leftrightarrow h_r((d_1, d_2), u) \vee h_s((d_1, d_2), u).$$

Lemma 4.5 *For each $(d_1, d_2){:}D$ and $u{:}Time$, if $h'_a(\phi((d_1, d_2)), v)$ and $u \leq v$, together with $\mathcal{I}((d_1, d_2), u)$, for some $a \in Act$, then for some $n \geq 0$, there are $t_0, \cdots, t_n{:}D$ with $(d_1, d_2) = t_0$, and elements $u_0 < \cdots < u_n{:}Time$ with $u = u_0$ and $v = u_n$ such that $t_i \Longrightarrow_{u_i} t_{i+1}$, for $i < n$, and $FC(t_n, u_n)$.*

Proof. Let $h'_a(\phi((d_1,d_2)),v)$ for $a \in \{s,r\}$. We need to distinguish two cases.

- $a = s$. By definition of h'_s, $d_f = v - 2\Delta$. If $d_s = \varepsilon$, then by definition of ϕ either
  - $d_2 = d_s = \varepsilon$ and $d_1 = d_f = v - 2\Delta$. By $\mathcal{I}_1((d_1,d_2),u)$, $u \leq v - \Delta$. Hence $h_\tau((d_1,d_2),v-\Delta)$ holds, (d_1,d_2) can perform a τ to $(\varepsilon, v-\Delta)$, and $\mathcal{I}((\varepsilon,v-\Delta),v)$. Since $h_r((\varepsilon,v-\Delta),v)$, $FC((\varepsilon,v-\Delta),v)$; or
  - $d_1 = d_s = \varepsilon$ and $d_f = v-2\Delta = d_2-\Delta$. Then $d_2 = v-\Delta$. By $\mathcal{I}_2((d_1,d_2),u)$, $u \geq v-\Delta$. Since $\mathcal{I}((d_1,d_2),v)$ and $h_r((d_1,d_2),v)$, we have $FC((d_1,d_2),v)$.

  If $d_s \neq \varepsilon$, then by definition of ϕ, $d_s = d_1$ and $d_f = d_2-\Delta$. Since $d_f = v-2\Delta$, we have $v = d_2+\Delta$. By $\mathcal{I}_3((d_1,d_2),u)$, $d_2 \leq d_1$. Since $v = d_2+\Delta \leq d_1+\Delta$, $\beta_1(v)$. $\mathcal{I}((d_1,d_2),v)$ and $h_s((d_1,d_2),v)$ hold, hence $FC((d_1,d_2),v)$.
- $a = r$. By definition of ϕ, $d_f = \varepsilon$ implies $d_1 = d_2 = \varepsilon$. $\mathcal{I}((d_1,d_2),v)$ and $h_r((d_1,d_2),v)$ hold, hence $FC((d_1,d_2),v)$. If $d_f \neq \varepsilon$, then by definition of h'_r, $d_s = \varepsilon$ and $d_f + \Delta \leq v \leq d_f + 2\Delta$. By definition of ϕ, we know that either
  - $d_1 = \varepsilon$, $d_2 \neq \varepsilon$ and $d_f = d_2 - \Delta$. By $v \leq d_f + 2\Delta$, $v \leq d_2 + \Delta$, we have $\beta_2(v)$. By $\mathcal{I}((d_1,d_2),v)$ and $h_r((d_1,d_2),v)$, $FC((d_1,d_2),v)$ holds; or
  - $d_2 = \varepsilon$ and $d_f = d_1$. Since $d_f + \Delta \leq v$, $d_1 + \Delta \leq v$. By $\mathcal{I}_1((d_1,d_2),u)$, $u \leq d_1 + \Delta \leq v$. Since $h_\tau((d_1,d_2),d_1+\varepsilon)$, by performing action τ at time $d_1 + \Delta$, we can reach a state $(\varepsilon, d_1+\Delta)$ and $\mathcal{I}((\varepsilon,d_1+\Delta),v)$. Since $v \leq d_1 + 2\Delta$, $\beta_2(v)$ holds. So $h_r((\varepsilon,v),v)$, hence $FC((\varepsilon,v),v)$.

□

Lemma 4.6 $I_{TB}((\varepsilon,\varepsilon)) \underline{\leftrightarrow}_{tb} S_{TB}(\phi((\varepsilon,\varepsilon)))$.

Proof. Take any (d_1,d_2) and u such that $\mathcal{I}(d,u)$; let $\phi((d_1,d_2)) = (d_f,d_s)$. We show that $(d_1,d_2)B_u\phi((d_1,d_2))$. The criteria VI and VII hold trivially, since $h_\delta = \emptyset$. The first five criteria are shown as follows.

(i) Suppose that $h_\tau(d,u)$. We show that $\phi((d_1,d_2)) = \phi(g_\tau((d_1,d_2),u))$ and $DC_{S_{TB}}(\phi((d_1,d_2)),u)$. By definition of h_τ, $d_1 = u - \Delta$ and $d_2 = \varepsilon$. And by definition of ϕ, hence $\phi((d_1,d_2)) = (d_1,d_2)$. Also $\phi(g_\tau((d_1,d_2),u)) = \phi(\varepsilon,u) = (u-\Delta,\varepsilon) = (d_1,d_2)$. Since $h'_s(\phi(d_1,d_2),d_1+2\Delta)$, it follows that $DC_{S_{TB}}(\phi((d_1,d_2)),u)$.

(ii) Suppose that $h_a((d_1,d_2),u)$. We show that $h'_a(\phi((d_1,d_2)),u)$.

2.1 $a = s$: By definition of h_s, $d_2 = u - \Delta$. By definition of ϕ, $d_2 \neq \varepsilon$, and $d_f = d_2 - \Delta$. Since $d_2 = u - \Delta$, $d_f = u - 2\Delta$, hence $h'_a(\phi((d_1,d_2)),u)$.

2.2 $a = r$: By definition of h_r, $d_1 = \varepsilon$ and $\beta_2(u)$.

2.2.1 $d_2 = \varepsilon$: By definition of ϕ, $d_f = d_1 = \varepsilon$, hence $h'_r((d_f,d_s),u)$.

2.2.2 $d_2 \neq \varepsilon$: By definition of ϕ, $d_f = d_2 - \Delta$ and $d_s = d_1 = \varepsilon$. By $\beta_2(u)$, $u \leq d_2 + \Delta$. By $\mathcal{I}_2((d_1,d_2),t)$, $d_2 \leq u$. Since $d_2 \leq u \leq d_2 + \Delta$, hence $h'_r(\phi((d_1,d_2)),u)$.

(iii) Trivial, by Definition 4.4.

(iv) Trivial, since M is a singleton set.

(v) Suppose that $h_a((d_1, d_2), u)$. We need to show that $\phi(g_a((d_1, d_2), u)) = g'_a(\phi((d_1, d_2)), u)$.

5.1 $a = s$: Then $d_2 \neq \varepsilon$, and $\phi(g_s((d_1, d_2), u)) = \phi((d_1, \varepsilon)) = (d_1, \varepsilon) = g'_s((d_2 - \Delta, d_1), u) = g'_s(\phi((d_1, d_2)), u)$.

5.2 $a = r$: Then $d_1 = \varepsilon$.

5.2.1 $d_2 = \varepsilon$: Then $\phi(g_r((\varepsilon, \varepsilon), u)) = \phi(u, \varepsilon) = (u, \varepsilon) = g'_r((\varepsilon, \varepsilon), u) = g'_r(\phi(\varepsilon, \varepsilon), u) = g'_r(\phi((d_1, d_2)), u)$.

5.2.2 $d_2 \neq \varepsilon$: Then $\phi(g_r((d_1, d_2), u)) = (d_2 - \Delta, u) = g'_r((d_2 - \Delta, \varepsilon), u) = g'_r(\phi(\varepsilon, d_2), u) = g'_r(\phi((d_1, d_2)), u)$.

By Theorem 3.4 and Lemma 4.5, $(d_1, d_2)B_u\phi((d_1, d_2))$. Hence, $I_{TB}((\varepsilon, \varepsilon)) \leftrightarrow_{tb} S_{TB}(\phi((\varepsilon, \varepsilon)))$.

□

Acknowledgement

We thank Mark van der Zwaag for pointing out some serious technical flaws in an earlier version of this paper. We are also indebted to the anonymous referees for their helpful comments. This research is supported by the Dutch Technology Foundation STW under the project CES5008 – "Improving the quality of embedded systems by formal design and systematic testing".

References

[1] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2): 142–188, 1991.

[2] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51: 129–176, 1987.

[3] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.

[4] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1-3):109–137, 1984.

[5] W.J. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000.

[6] W.J. Fokkink, J.F. Groote, J. Pang, B. Badban, and J.C. van de Pol. Verifying a sliding window protocol in μCRL. In *Proc. 10th Conference on Algebraic Methodology and Software Technology*, Lecture Notes in Computer Science 3116, pp. 148-163. Springer, 2004.

[7] W.J. Fokkink and J. Pang. Cones and foci for protocol verification revisited. In *Proc. 6th Conference on Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science 2620, pp. 267–281, Springer, 2003.

[8] J.F. Groote. The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, Amsterdam, 1997.

[9] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. In *Proc. 1st Workshop on the Algebra of Communicating Processes*, Workshops in Computing Series, pp. 26–62. Springer, 1995.

[10] J.F. Groote and M.A. Reniers. Algebraic process verification. In *Handbook of Process Algebra*, pp. 1151–1208. Elsevier, 2001.

[11] J.F. Groote, M.A. Reniers, J.J. van Wamel, and M.B. van der Zwaag. Completeness of timed μCRL. *Fundamenta Informaticae*, 50(3/4): 361–402, 2002.

[12] J.F. Groote and J. Springintveld. Focus points and convergent process operators. A proof strategy for protocol verification. *Journal of Logic and Algebraic Programming*, 49(1/2): 31–60, 2001.

[13] J.F. Groote and J.J. van Wamel. Analysis of three hybrid systems in timed μCRL. *Science of Computer Programming*, 39(2/3): 215–247, 2001.

[14] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

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

[16] A.S. Klusener. Abstraction in real time process algebra. In *Proc. 2nd Conference on Concurrency Theory*, Lecture Notes in Computer Science 527, pp. 376–392. Springer, 1991.

[17] A.S. Klusener. The silent step in time. In *Proc. 3rd Conference on Concurrency Theory*, Lecture Notes in Computer Science 630, pp. 421–435. Springer, 1992.

[18] R. Milner. *A Calculus of Communicating Systems.* LNCS 92, Springer, 1980.

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

[20] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58(1/3): 249–261, 1988.

[21] Y. Wang. CCS + time = an interleaving model for real time systems. In *Proc. 18th Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 510, pp. 217–228. Springer, 1991.

[22] Y.S. Usenko. *Linearization in μCRL.* PhD thesis, Eindhoven University of Technology, 2002.

[23] J.J. Vereijken. *Discrete-Time Process Algebra.* PhD thesis, Eindhoven University of Technology, 1997.

[24] M.B. van der Zwaag. The cones and foci proof technique for timed transition systems. *Information Processing Letters*, 80(1): 33–40, 2001.