Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

The Cones and Foci proof Technique for Timed Transition Systems

M.B. van der Zwaag

Software Engineering (SEN)

SEN-R0038 December 31, 2000

Report SEN-R0038
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

The Cones and Foci Proof Technique for Timed Transition Systems

Mark B. van der Zwaag*

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Mark.van.der.Zwaag@cwi.nl

ABSTRACT

We propose an extension of the cones and foci proof technique that can be used to prove timed branching bisimilarity of states in timed transition systems. We prove the correctness of this technique and we give an example verification.

2000 Mathematics Subject Classification: 68Q45; 68Q60; 68Q70

Keywords & Phrases: timed μCRL, verification techniques, timed transition systems

Note: Work carried out under project SEN2.1 "Process Specification and Analysis".

1 Introduction

Time often plays a crucial rôle in process behaviour. For this reason, process algebras such as CCS [18], CSP [15] and ACP [4, 5] have been extended with some notion of time [19, 20, 1, 2]. In general, these approaches tend to be restricted to the syntax and semantics of these formalisms. Disappointingly, protocol verification in timed process algebras has proved to be a complex task.

In this paper, we propose a method that will make larger timed verifications feasible. The method is designed for the extension of the specification language μCRL [11] with time [9, 12]. This formalism is based on ACP; it combines axiomatic process-algebraic reasoning and equational abstract data types. Time is treated as a data type with some total ordering, which provides a powerful but relatively simple way of expressing timing properties.

Groote and Springintveld [13] introduced a method to prove that the transition systems generated by two μCRL process equations – one called the implementation, the other the specification – are (untimed) branching bisimilar [8]. These process equations, brought in a linear format, provide pre-condition, action and effect functions for the transitions in the transition systems associated with them. The proof technique is completely in terms of these functions. This way, one can prove branching bisimilarity without generating the associated transition systems. This technique, referred to as the *cones and foci* proof technique, has been applied successfully in numerous case studies; see e.g. [7, 10, 14, 21].

We give the adaptation of the cones and foci technique for timed branching bisimilarity. The definition of timed branching bisimilarity can differ substantially depending on the assumptions made in modeling timed behavior. In timed μCRL, one of the most prominent assumptions is left open: the time domain can be *any* nonempty totally ordered set. We propose a definition of timed branching bisimilarity that coincides with the definition in discrete time ACP [3] in case a discrete time domain is chosen. In case of a continuous time domain, our definition corresponds to the notion of timed

*Supported by the Netherlands Organisation for Scientific Research (NWO) under contract SION 612-61-002.

branching bisimilarity in the setting of real time ACP with urgent actions [17]. The intuition is always that τ actions are silent/inert if they do not lose possible behaviors.

In timed μCRL, actions may be executed at the same time consecutively. As a consequence, the notion of timed branching bisimilarity is quite different from the one in real time ACP [16, 6], where this is not allowed.

In this paper, we have avoided the use of μCRL syntax, as we regard the proof technique primarily as semantical. We note, however, that in fact the untimed cones and foci technique of [13] is even stronger than indicated above; using a recursive specification principle, the implementation and the specification are proved *derivably* equal. Work in progress is an axiomatization of timed branching bisimilarity in timed μCRL, that allows the same result for the timed technique.

In Section 2, we introduce timed transition systems and timed branching bisimilarity. In Section 3, we introduce so-called process structures, that are the objects represented by timed μCRL linear process equations. We define the timed transition system associated with a process structure. In Section 4, we give the proof technique, and prove its correctness. In Section 5, we give an example of a verification using this technique. Although this verification is evidently quite simple, it shows that larger timed verifications are feasible.

2 Timed transition systems

Let A be a set of actions and let $\tau \notin A$ be a special action that models the execution of an unobservable action. Let $A_\tau = A \cup \{\tau\}$. Let T be a nonempty, totally ordered set of time elements. These sets are fixed throughout this paper. We shall write a to denote an arbitrary element of A_τ, and $u, v, \ldots$ to denote arbitrary elements of T.

Definition 2.1 A *timed transition system* is a triple (S, Tr, U), where

- S is a nonempty set of states,
- $Tr \subseteq S \times A_\tau \times T \times S$ is a set of transitions, and
- $U \subseteq S \times T$ is a *delay relation*, such that always
  - if $u < v$ and $U(s, v)$, then $U(s, u)$, and
  - if $Tr(s, a, u, r)$, then $U(s, u)$.

Transitions (s, a, u, r) express that state s evolves into state r by the execution of action a at time u. If $U(s, u)$, then we say that state s can let time pass, or "idle", until time u. We write $s \xrightarrow{a}_u r$ for transitions (s, a, u, r); a transition relation consists of binary relations $\xrightarrow{a}_u$ on the state set. For any $u \in T$, we define the generalised τ-step relation $\Rightarrow_u$ as the reflexive transitive closure of $\xrightarrow{\tau}_u$.

We define timed *branching bisimilarity* of states in timed transition systems. A timed bisimulation R relates states at some times; for a state set S, it is a subset of $S \times T \times S$. We may write sR_ur for $R(s, u, r)$.

Definition 2.2 A relation $R \subseteq S \times T \times S$ is a *timed branching bisimulation* over the timed transition system (S, Tr, U), if whenever sR_ur, then also rR_us, and

i. if $s \xrightarrow{a}_u s'$ for some a and s', then either
   (a) $a = \tau$ and $s'R_ur$, or
   (b) $r \Rightarrow_u r'' \xrightarrow{a}_u r'$ and sR_ur'' and $s'R_ur'$ for some r' and r'';

ii. if $u < v$ and $U(s, v)$ for some v, then, for some $n > 0$, there are r_i, u_i such that $u = u_0$, $v = u_n$, $r = r_0$, $U(r_n, v)$, sR_vr_n, and, for all $i < n$, $r_i \Rightarrow_{u_i} r_{i+1}$, $u_i < u_{i+1}$, $sR_{u_i}r_i$, and $sR_{u_i}r_{i+1}$.

The states s and r are *timed branching bisimilar at* u, if there exists a timed branching bisimulation R with sR_ur. States s and r are *timed branching bisimilar*, if they are timed branching bisimilar at every u in T.

By the first clause in the definition of a branching bisimulation, we treat the behavior of a state at some point in time like untimed behavior (see e.g. [5, 8] for an introduction to untimed branching bisimulation). By the second clause, we demand that time passing in a state s is matched by a related state r with a "τ-idle-path" where all intermediate states are related at the appropriate times with s.

It is straightforward to verify that branching bisimilarity is an equivalence relation. We defined bisimilarity of states in the *same* transition system. States of different transition systems are said to be branching bisimilar at u, if they are branching bisimilar at u in the disjoint union of the transition systems, that is defined straightforwardly.

A state s is *convergent at time* u in a transition system, if that system has no infinite sequence $s_0u_0s_1u_1s_2u_2\cdots$ 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 $u_i \leq u_{i+1}$.

3 Process structures

We introduce *(timed) process structures*, that are represented by timed μCRL linear process equations. We first fix the action names used, and auxiliary sets $D_{\mathtt{a}}$, for the parameters of actions. Let $\delta \notin A_\tau$ model a case of *inaction*.

Definition 3.1 Let Act be a collection of functions $\mathtt{a} : D_{\mathtt{a}} \to A$, where the $D_{\mathtt{a}}$ are nonempty sets. We require that $\tau, \delta \notin Act$ and that $range(\mathtt{a})$ and $range(\mathtt{a}')$ are disjoint for all distinct $\mathtt{a}, \mathtt{a}' \in Act$. We write Act_τ for the set $Act \cup \{\tau\}$, and $Act_{\delta\tau}$ for $Act \cup \{\tau, \delta\}$.

Process structures consist of a *state space* D, of a set of *environments* E, of pre-conditions b of actions, of functions f that give the parameters of actions, and of functions g that give the effect of the execution of actions, i.e. the state that the system evolves into by the execution of the action. The environments are used to provide fresh inputs to a process. Also, the environments allow the description of nondeterministic processes. In Section 3.1, we give an example of a process structure.

Definition 3.2 A *process structure* P over Act is a tuple (D, E, f, b, g), where

- D is a nonempty set called the state space of P,
- E is a nonempty set of environments,
- f is a collection of functions $f_{\mathtt{a}} : D \times E \times T \to D_{\mathtt{a}}$, one for for every $\mathtt{a}$ in Act,
- b is a collection of relations $b_{\mathtt{a}} \subseteq D \times E \times T$, one for every $\mathtt{a}$ in $Act_{\delta\tau}$,
- g is a collection of functions $g_{\mathtt{a}} : D \times E \times T \to D$, one for every $\mathtt{a}$ in Act_τ.

The functions $f_{\mathtt{a}}, g_{\mathtt{a}}$ may be partial, but must be defined on the elements of $b_{\mathtt{a}}$.

Process structures are the objects that are represented by timed μCRL linear process equations; we have abstracted from the μCRL syntax in order to smoothen the presentation. We feel that this is justified, because, in the end, we want a semantic result: we prove two processes bisimilar. We postpone the task of proving derivable equality in the μCRL proof system until the establishment of an axiomatization of timed branching bisimilarity, which is in progress. The recursive specification principle underlying the derivability result can be adapted to the timed case straightforwardly.

For the remainder of this section, we fix an arbitrary process structure P written as above. With P we associate a transition system as follows.

Definition 3.3 The timed transition system $tts(P)$ is defined as (D, Tr, U), where Tr and U are the smallest sets such that, for all $d \in D$, $\mathtt{a} \in Act_{\delta\tau}$, $e \in E$ and $u, v \in T$,

- if $b_{\mathtt{a}}(d, e, u)$ and $\mathtt{a} \neq \delta$, then $Tr(d, a, u, g_{\mathtt{a}}(d, e, u))$, where $a = \tau$, if $\mathtt{a} = \tau$, and $a = \mathtt{a}(f_{\mathtt{a}}(d, e, u))$ otherwise, and
- if $b_{\mathtt{a}}(d, e, u)$ and $v \leq u$, then $U(d, v)$.

Observe that the environments may be used to describe nondeterminism: it may be that for environments e_1 and e_2, it holds that $b_{\mathtt{a}}(d, e_1, u)$ and $b_{\mathtt{a}}(d, e_2, u)$, while $f_{\mathtt{a}}(d, e_1, u) = f_{\mathtt{a}}(d, e_2, u)$ and $g_{\mathtt{a}}(d, e_1, u) \neq g_{\mathtt{a}}(d, e_2, u)$.

The relation b_δ may be used to specify the presence of so-called time deadlocks. In the untimed case, it is not necessary to specify deadlocks explicitly. Here, time deadlocks determine the process behaviour as follows: if $b_\delta(d, e, u)$, then $U(d, u)$: 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 3.4 The *delay condition* $DC_P \subseteq D \times T$ is defined by $DC_P(d, u)$ if and only if $b_{\mathtt{a}}(d, e, v)$ and $u \leq v$ for some $\mathtt{a} \in Act_{\delta\tau}$, $e \in E$ and $v \in T$.

It holds that $DC_P(d, u)$, if and only if $U(d, u)$ in $tts(P)$; if $DC_P(d, u)$, then in state d time may pass at least until time u.

Definition 3.5 The *focus condition* $FC_P \subseteq D \times T \times T$ is defined by $FC_P(d, u, v)$ if and only if there are no $u' \in T$ and $e \in E$ such that $u \leq u' \leq v$ and $b_\tau(d, e, u')$.

If $FC_P(d, u, v)$, then the state d is called a *focus point* between times u and v; it has no outgoing τ-steps between u and v in $tts(P)$. An untimed focus point is simply a state without outgoing τ-steps. We tried some alternatives for the adaptation to the timed case, including the obvious notion of "focus point at time u", but eventually we found the above definition of a focus point relative to *two* points in time the most convenient.

Definition 3.6 A relation $I \subseteq D \times T$ is an *invariant* of P, if whenever $I(d, u)$ and $b_{\mathtt{a}}(d, e, v)$ and $u \leq u' \leq v$, then $I(d, u')$ and, if $\mathtt{a} \neq \delta$, $I(g_{\mathtt{a}}(d, e, v), v)$.

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

3.1 Example: buffers

We give a process structure B that models the behaviour of a buffer with capacity one. Between the reading and the sending of a message, there is a fixed time delay Δ. Let M be a nonempty set of messages.

Let $Act = \{\mathtt{r}, \mathtt{s}\}$ and $D_{\mathtt{r}} = D_{\mathtt{s}} = M$. An action $\mathtt{s}(m)$ models the sending of message m, and $\mathtt{r}(m)$ models the receiving of message m.

A buffer B is the process structure (D, E, f, b, g) over Act with state space

$$D = \{\varepsilon\} \cup (M \times T),$$

$E = M$, and f, b, g defined as follows.

$$\begin{array}{rcll} f_{\mathtt{s}}((m,v),e,u) & = & m & \\ f_{\mathtt{r}}(\varepsilon,e,u) & = & e & \\ b_{\mathtt{s}}(d,e,u) & \Leftrightarrow & d=(m,u-\Delta) & \\ b_{\mathtt{r}}(d,e,u) & \Leftrightarrow & d=\varepsilon & \\ b_{\mathtt{a}} & = & \emptyset & \text{if } \mathtt{a}\in\{\delta,\tau\} \\ g_{\mathtt{s}}(d,e,u) & = & \varepsilon & \\ g_{\mathtt{r}}(d,e,u) & = & (e,u) & \end{array}$$

A buffer in state ε is empty and ready to read any message at any time; this is true because $b_{\mathtt{r}}(\varepsilon,m,u)$ holds for all $m \in M$ and $u \in T$. This case also illustrates the use of the set $E = M$ for the provision of inputs. By making no restrictions on m, we enable the input of any message.

A buffer in a state (m,v) has read message m at time v, and will send the message at time $v+\Delta$. Observe that, for all u and m, $tts(B)$ has transitions

$$\varepsilon \xrightarrow{\mathtt{r}(m)}_u (m,u) \xrightarrow{\mathtt{s}(m)}_{u+\Delta} \varepsilon.$$

Also observe that $DC_B(\varepsilon,u)$ for all u, and $DC_B((m,v),u)$ for all $u \le v+\Delta$.

4 Cones and foci

In the untimed technique, a focus point is a state that has no outgoing τ-transitions. The idea is that, in convergent transition systems[1], every state of the implementation must, after a number of τ-steps, reach a focus point. The part of the state space from which a focus point can be reached is referred to as its *cone*. A mapping from states of the implementation to states of the specification must be given, where the specification does not have τ-transitions. A focus point is given the same image as the elements of its cone. If this mapping satisfies certain criteria, that are referred to as the *matching criteria*, then it induces a branching bisimulation.

In the timed case, this visualisation of cones and focus points is obscured by the timing of transitions, but still the guiding intuition. Here, we express the matching criteria relative to a state at some time.

Let Act be written as in Definition 3.1 and let $P = (D,E,f,b,g)$ and $Q = (D',E,f',b',g')$ be process structures over Act with $b'_\tau = \emptyset$; so $tts(Q)$ does not have τ-transitions.

Definition 4.1 Let h be a mapping from D to D'. We say that h satisfies the *matching criteria* for an element d of D and a time element u, notation $C_h(d,u)$, if, for all $\mathtt{a} \in Act$, $e \in E$ and $v \in T$, the following conditions hold.

1. d is convergent at u in $tts(P)$,
2. If $b_\tau(d,e,u)$, then $h(d) = h(g_\tau(d,e,u))$ and $DC_Q(h(d),u)$.

   If a state can do a τ-step at time u, then the resulting state has the same image. Also, this image should be able to let time pass until u.
3. If $b_{\mathtt{a}}(d,e,u)$, then $b'_{\mathtt{a}}(h(d),e,u)$.

   If a state has an $\mathtt{a}$-step at time u, then its image also has some $\mathtt{a}$-step at time u.

[1] In [13], also an extended technique is presented that deals with τ-divergence using the fairness principle CFAR.

4. If $b'_{\mathtt{a}}(h(d), e, v)$ and $u \leq v$ and $FC_P(d, u, v)$, then $b_{\mathtt{a}}(d, e, v)$.

   If the image of d has an $\mathtt{a}$-step at some time v later dan u, and d is a focus point between u and v, then d also has some $\mathtt{a}$-step at time v.

5. If $b_{\mathtt{a}}(d, e, u)$, then $f_{\mathtt{a}}(d, e, u) = f'_{\mathtt{a}}(h(d), e, u)$.

   If a state can do some $\mathtt{a}$-action at time u for some e, then its image can do the same action at time u.

6. If $b_{\mathtt{a}}(d, e, u)$, then $h(g_{\mathtt{a}}(d, e, u)) = g'_{\mathtt{a}}(h(d), e, u))$.

   If a state has an $\mathtt{a}$-step at time u for some e, then the resulting state should be mapped to the result of executing the same action in its image.

7. If $b_\delta(d, e, u)$, then $DC_Q(h(d), u)$.

   If a state has a time deadlock at time u, then its image should be able to let time pass until u.

8. If $b'_\delta(h(d), e, v)$ and $u < v$ and $FC_P(d, u, v)$, then $DC_P(d, v)$.

   If $h(d)$ has a time deadlock at some time v strictly after u, and d is a focus point between u and v, then d can let time pass until v.

The first 6 criteria are the adaptations of the criteria for the untimed case. The last two had to be added in order to deal with explicit time deadlocks, that do not exist in the setting without time.

In general, it will not be possible to find a state mapping that satisfies the matching criteria for all states and all times. Using an invariant, we can limit ourselves to the part of $D \times T$ that satisfies the invariant. This is stated in the next theorem. This theorem is the timed counterpart of the *general equality theorem* of [13].

Theorem 4.1 Let P and Q be written as above. If I is an invariant of P and $h : D \to D'$ is a mapping such that $I(d, u)$ implies $C_h(d, u)$ for all d and u, then, for any d_0 and u_0 with $I(d_0, u_0)$, it holds that d_0 and $h(d_0)$ are timed branching bisimilar at u_0.

Proof. Let I be an invariant of P, and let h be a state mapping that satisfies they matching criteria for all d and u with $I(d, u)$. We write mc_i to refer to the ith matching criterium.

Assume, without loss of generality, that D and D' are disjoint. So the union of $tts(P)$ and $tts(Q)$ is (D'', Tr, U), where D'' is the union of D and D', Tr is the union of the transitions of $tts(P)$ and $tts(Q)$, and U is the union of the delay relations of $tts(P)$ and $tts(Q)$. It is easily seen that if a state is convergent at time u in $tts(P)$, then it is also convergent at u in this union.

Let $R \subseteq D'' \times T \times D''$ be the smallest set such that whenever $I(d, u)$, then $R(d, u, h(d))$ and $R(h(d), u, d)$. We show that R is a timed branching bisimulation over (D'', Tr, U).

Take any x, y and u with xR_uy; by definition of R either $x = h(y)$ or $y = h(x)$, and in both cases also yR_ux.

Action step. Suppose that $x \xrightarrow{a}_u x'$. This step must be matched in the right way by y. First consider the case where $y = h(x)$. By definition of R we know $I(x, u)$, so by assumption also $C_h(x, u)$.

1. If $a = \tau$, then $b_\tau(x, e, u)$ and $x' = g_\tau(x, e, u)$, for some e, by Definition 3.3. By mc_2 we have $h(x) = h(x')$, so $x'R_uy$ by definition of R, qed.

2. If $a \neq \tau$, then $b_{\mathtt{a}}(x, e, u)$ and $x' = g_{\mathtt{a}}(x, e, u)$ and $a = \mathtt{a}(f_{\mathtt{a}}(x, e, u))$, for some $\mathtt{a}$ in Act, e in E, by Definition 3.3. It holds by mc_3 that $b'_{\mathtt{a}}(h(x), e, u)$, by mc_5 that $\mathtt{a}(f'_{\mathtt{a}}(h(x), e, u)) = a$, and by mc_6 that $h(x') = g'_{\mathtt{a}}(h(x), e, u)$. So we know by Definition 3.3 that $h(x) \xrightarrow{a}_u h(x')$ and by definition of R we have $x'R_uh(x')$, qed.

Now consider the case where $x = h(y)$. By the assumption that $b'_\tau = \emptyset$, we see that $a \neq \tau$. So, for some $\mathtt{a}$ in Act and e in E, it holds that $b'_\mathtt{a}(x, e, u)$ and $x' = g'_\mathtt{a}(x, e, u)$ and $a = \mathtt{a}(f'_\mathtt{a}(x, e, u))$. Now consider y. By definition of R, we know $I(y, u)$; so also $C_h(y, u)$. By mc_1, there is a y' such that $y \Rightarrow_u y'$ and there is no τ-step from y' at u; so $FC_P(y', u, u)$. As the invariant and hence the matching criteria hold for all states on this τ-path, we can repeatedly apply mc_2 and Definition 3.3 to get $h(y') = h(y) = x$.

We have $b_\mathtt{a}(y', e, u)$ by mc_4, $a = \mathtt{a}(f_\mathtt{a}(y', e, u))$ by mc_5 and $h(g_\mathtt{a}(y', e, u)) = x'$ by mc_6.

By Definition 3.3, we have $y \Rightarrow_u y' \xrightarrow{a}_u g_\mathtt{a}(y', e, u)$, and by definition of R we have that $y' R_u x$ and $g_\mathtt{a}(y', e, u) R_u x'$, qed.

Delay behavior. Suppose that $u < v$, $U(x, v)$. This delay behavior must be matched in the right way by y. First consider the case where $y = h(x)$. By definition of R, we know $I(x, u)$; so by assumption also $C_h(x, u)$.

From Definition 3.3, we know that $b_\mathtt{a}(x, e, v')$ for some $\mathtt{a}$, e and $v' \geq v$. So $I(x, v)$ and $I(x, v')$, and therefore $C_h(x, v')$. Case distinction: if $\mathtt{a} = \tau$, then $DC_Q(y, v')$ by mc_2; if $\mathtt{a} = \delta$, then $DC_Q(y, v')$ by mc_7; else $DC_Q(y, v')$ by mc_3. So $DC_Q(y, v')$. By Definition 3.3, we know that $U(y, v)$, and by definition of R we have $x R_v y$, qed.

Now consider the case where $x = h(y)$. By Definition 3.3, we have that $b'_\mathtt{a}(x, e, v')$ and $v \leq v'$ for some $\mathtt{a}$, e and v'. Now consider y. It holds that $I(y, u)$, and hence $C_h(y, u)$, by definition of R. By mc_1, it holds that for some $n \geq 0$, there are y_i, u_i such that, $u = u_0$, $y = y_0$, $y_i \Rightarrow_{u_i} y_{i+1}$, for all $i \leq n$, and, for all $i < n$, $u_i < u_{i+1}$, such that $FC_P(y_{n+1}, u_n, v')$ and $u_n \leq v'$.

We see that the invariant holds for all intermediate states on this τ-idle-path. Therefore we can by repeatedly applying mc_2 and Definition 3.3 derive that $h(y_i) = h(y) = x$ for all $i \leq n + 1$. Also it follows, by definition of R, that $y_i R_{u_i} x$ and $y_{i+1} R_{u_i} x$ for all $i \leq n$.

If $u_n \geq v$, then there is an $i < n$, such that $U(y_{i+1}, v)$, qed.

If $u_n < v$, then, if $\mathtt{a} \neq \delta$, it follows from mc_4 that $b_\mathtt{a}(y_{n+1}, e, v')$ and hence $U(y_{n+1}, v')$ by Definition 3.3. If $\mathtt{a} = \delta$, then it follows from mc_8 that $DC_P(y_{n+1}, v')$ and hence $U(y_{n+1}, v')$. So $U(y_{n+1}, v')$. We see that also $U(y_{n+1}, v)$, qed.

We conclude that R is a timed branching bisimulation over (D'', Tr, U). It holds by definition of R and the assumption $I(d_0, u_0)$ that $d_0 R_{u_0} h(d_0)$. Therefore d_0 and $h(d_0)$ are timed branching bisimilar at u_0. □

5 Example: two serial buffers

Consider the buffers introduced in Section 3.1. We now 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 modelled by a τ action. After the communication of the message, the first buffer returns to the empty state. The second buffer outputs the message at time $u + 2\Delta$.

The implementation The action declarations are as in Section 3.1. To simplify the example, we assume that the set M of messages is a singleton; we abstract from the identity of messages. Consequently, we can represent the set $\{\varepsilon\} \cup (M \times T)$ (the state space of single buffers) by the set $T_\varepsilon = T \cup \{\varepsilon\}$.

The implementation P is the process structure (D, M, f, b, g) with state space $D = T_\varepsilon \times T_\varepsilon$, and f, b, g defined below. Now that there is only one message, we do not write the second function

argument "e". Also note that f is defined trivially.

$$\begin{array}{rcl} b_{\mathtt{s}}((d_1,d_2),u) & \Leftrightarrow & d_2 = u - \Delta \text{ and } \beta_1(u) \\ b_{\mathtt{r}}((d_1,d_2),u) & \Leftrightarrow & d_1 = \varepsilon \text{ and } \beta_2(u) \\ b_{\tau}((d_1,d_2),u) & \Leftrightarrow & d_1 = u - \Delta \text{ and } d_2 = \varepsilon \\ b_{\delta} & = & \emptyset \\ g_{\mathtt{s}}((d_1,d_2),u) & = & (d_1,\varepsilon) \\ g_{\mathtt{r}}((d_1,d_2),u) & = & (u,d_2) \\ g_{\tau}((d_1,d_2),u) & = & (\varepsilon,u) \end{array}$$

The conditions $\beta_i(u)$, with $i \in \{1,2\}$, abbreviate ($d_i = \varepsilon$ or $u \leq d_i + \Delta$). These conditions have to be added in order to avoid timing inconsistencies.

The specification The specification Q is the process structure (D, M, f', b', g') with f', b', and g' defined below. It has the same state space as the implementation, but the rôles of the constituents of states are different. In a state (d_f, d_s), the d_f is the time the *first* contained message was received, and d_s is the time of the second. If the system is empty, then $d_f = d_s = \varepsilon$. An invariant of Q is that $d_f = \varepsilon$ implies $d_s = \varepsilon$.

$$\begin{array}{rcl} b'_{\mathtt{s}}((d_f,d_s),u) & \Leftrightarrow & d_f = u - 2\Delta \\ b'_{\mathtt{r}}((d_f,d_s),u) & \Leftrightarrow & d_f \neq \varepsilon \text{ implies } (d_s = \varepsilon \text{ and } d_f + \Delta \leq u \leq d_f + 2\Delta) \\ b'_{\tau} & = & \emptyset \\ b'_{\delta} & = & \emptyset \\ g'_{\mathtt{s}}((d_f,d_s),u) & = & (d_s,\varepsilon) \\ g'_{\mathtt{r}}((d_f,d_s),u) & = & \begin{cases} (u,\varepsilon) & \text{if } d_f = \varepsilon \\ (d_f,u) & \text{otherwise} \end{cases} \end{array}$$

The verification We define the state mapping $h : D \to D$ by

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

The invariant I of the implementation is defined as follows. Let $I((d_1,d_2),u)$ be the conjunction of

$$\begin{array}{ll} I_1: & \text{if } d_1 \neq \varepsilon, \text{ then } u \leq d_1 + \Delta, \\ I_2: & \text{if } d_2 \neq \varepsilon, \text{ then } d_2 \leq u, \\ I_3: & \text{if } d_1 \neq \varepsilon \text{ and } d_2 \neq \varepsilon, \text{ then } d_2 \leq d_1. \end{array}$$

It is straightforward to check that I is indeed an invariant of P.

Lemma 5.1 $I(d,u)$ implies $C_h(d,u)$ for all $d \in D$ and $u \in T$.

Proof. Take any d and u such that $I(d,u)$. We show that $C_h(d,u)$ by checking the matching criteria for any $\mathtt{a} \in Act$ and $v \in T$. Let $d = (d_1,d_2)$ and $h(d) = (d_f,d_s)$. The criteria 7 and 8 hold trivially, since $b_\delta = \emptyset$. The first six criteria are shown as follows.

1. Clearly the implementation is convergent: every τ-step leads to a state where no further τ-step is enabled.

2. Suppose that $b_\tau(d,u)$. We show that $h(d) = h(g_\tau(d,u))$ and $DC_Q(h(d),u)$.

   By definition of b_τ, we see that $d_1 = u - \Delta$ and $d_2 = \varepsilon$, and hence $h(d) = d$ by definition of h. Also $h(g_\tau(d,u)) = h(\varepsilon,u) = (u-\Delta,\varepsilon) = d$.

   From $b'_{\mathtt{s}}(h(d), d_1 + 2\Delta)$, it follows that $DC_Q(h(d),u)$.

3. Suppose that $b_{\mathtt{a}}(d,u)$. We show that $b'_{\mathtt{a}}(h(d),u)$.

   • If $\mathtt{a} = \mathtt{s}$, then $d_2 = u - \Delta$ by definition of $b_{\mathtt{s}}$. We must show that $d_f = u - 2\Delta$.

   From $d_2 \neq \varepsilon$, we see by definition of h that $d_f = d_2 - \Delta$. With $d_2 = u - \Delta$, we get the required $d_f = u - 2\Delta$.

   • If $\mathtt{a} = \mathtt{r}$, then $d_1 = \varepsilon$ and $\beta_2(u)$ by definition of $b_{\mathtt{r}}$.

   If $d_2 = \varepsilon$, then $d_f = d_1 = \varepsilon$ by definition of h, and hence $b'_{\mathtt{r}}((d_f,d_s),u)$.

   If $d_2 \neq \varepsilon$, then $d_f = d_2 - \Delta$ and $d_s = d_1 = \varepsilon$ by definition of h. We see that $b'_{\mathtt{r}}(h(d),u)$, if $d_2 \leq u \leq d_2 + \Delta$. The first inequality follows from $I_2(d,t)$, and the second from $\beta_2(u)$.

4. Suppose that $b'_{\mathtt{a}}(h(d),v)$ and $u \leq v$ and $FC_P(d,u,v)$. We show that $b_{\mathtt{a}}(d,v)$.

   • If $\mathtt{a} = \mathtt{s}$, then $d_f = v - 2\Delta$ by definition of $b'_{\mathtt{s}}$.

   If $d_s = \varepsilon$, then by definition of h either

   – $d_2 = d_s = \varepsilon$ and $d_1 = d_f = v - 2\Delta$. Since $b_\tau(d, d_1 + \Delta)$ and $I_1(d,u)$, we see that this case violates assumption $FC_P(d,u,v)$; or

   – $d_1 = d_s = \varepsilon$ and $d_f = v - 2\Delta = d_2 - \Delta$. Then $v = d_2 + \Delta$, so $b_{\mathtt{s}}(d,v)$.

   If $d_s \neq \varepsilon$, then $d_s = d_1$ and $d_f = d_2 - \Delta$ by definition of h. Since also $d_f = v - 2\Delta$, we have $v = d_2 + \Delta$. The required $b_{\mathtt{s}}(d,v)$ holds if $\beta_1(v)$ which holds if $v \leq d_1 + \Delta$. Since $v = d_2 + \Delta$, we must show that $d_2 \leq d_1$. This holds by $I_3(d,u)$.

   • If $\mathtt{a} = \mathtt{r}$, then $d_f = \varepsilon$ implies, by definition of h, that $d_1 = d_2 = \varepsilon$, and hence $b_{\mathtt{r}}(d,v)$.

   If $d_f \neq \varepsilon$, then $d_s = \varepsilon$ and $d_f + \Delta \leq v \leq d_f + 2\Delta$ by definition of $b'_{\mathtt{r}}$. By definition of h, we know that either

   – $d_1 = \varepsilon$ and $d_2 \neq \varepsilon$ and $d_f = d_2 - \Delta$. We must show that $\beta_2(v)$, which holds if $v \leq d_2 + \Delta$. This follows from $d_f = d_2 - \Delta$ and $v \leq d_f + 2\Delta$; or

   – $d_2 = \varepsilon$ and $d_f = d_1$. From $d_f + \Delta \leq v$, it follows that $d_1 + \Delta \leq v$. This case contradicts the assumption $FC_P(d,u,v)$, since $b_\tau(d, d_1 + \Delta)$ and, by $I_1(d,u)$, $u \leq d_1 + \Delta$.

5. Trivial, since M is a singleton set.

6. Suppose that $b_{\mathtt{a}}(d,u)$. We show that $h(g_{\mathtt{a}}(d,u)) = g'_{\mathtt{a}}(h(d),u)$.

   • If $\mathtt{a} = \mathtt{s}$, then $d_2 \neq \varepsilon$, and

   $$h(g_{\mathtt{s}}(d,u)) = h(d_1,\varepsilon) = (d_1,\varepsilon) = g'_{\mathtt{s}}((d_2 - \Delta, d_1),u) = g'_{\mathtt{s}}(h(d),u).$$

   • If $\mathtt{a} = \mathtt{r}$, then $d_1 = \varepsilon$. If $d_2 = \varepsilon$, then

   $$h(g_{\mathtt{r}}((\varepsilon,\varepsilon),u)) = h(u,\varepsilon) = (u,\varepsilon) = g'_{\mathtt{r}}((\varepsilon,\varepsilon),u) = g'_{\mathtt{r}}(h(\varepsilon,\varepsilon),u) = g'_{\mathtt{r}}(h(d),u).$$

   If $d_2 \neq \varepsilon$, then

   $$h(g_{\mathtt{r}}(d,u)) = h(u,d_2) = (d_2 - \Delta, u) = g'_{\mathtt{r}}((d_2 - \Delta,\varepsilon),u) = g'_{\mathtt{r}}(h(\varepsilon,d_2),u) = g'_{\mathtt{r}}(h(d),u).$$

□

Take any d and u such that $I(d,u)$. By Theorem 4.1 and Lemma 5.1 we find that d and $h(d)$ are timed branching bisimilar at u. Consider e.g. start state $d = (\varepsilon,\varepsilon)$. Then also $h(d) = (\varepsilon,\varepsilon)$. It is easily seen that $I(d,u)$ for all time elements u, so d and $h(d)$ are timed branching bisimilar at any u.

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 and J. A. Bergstra. Discrete time process algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.

[3] J. C. M. Baeten, J. A. Bergstra, and M. A. Reniers. Discrete time process algebra with silent step. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language, and Interaction: Essays in Honour of Robin Milner*, chapter 18, pages 535 – 569. MIT Press, 2000.

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

[5] W. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. Springer-Verlag, 2000.

[6] W. J. Fokkink. An axiomatization for regular processes in timed branching bisimulation. *Fundamenta Informaticae*, 32:329–340, 1997.

[7] L.-Å. Fredlund, J. F. Groote, and H. P. Korver. Formal verification of a leader election protocol in process algebra. *Theoretical Computer Science*, 177:459–486, 1997.

[8] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, May 1996.

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

[10] J. F. Groote, F. Monin, and J. C. van de Pol. Checking verifications of protocols and distributed systems by computer. In D. Sangiorgi and R. de Simone, editors, *Proceedings of CONCUR'98*, volume 1466 of *Lecture Notes in Computer Science*, pages 629–655. Springer-Verlag, 1998.

[11] J. F. Groote and A. Ponse. The syntax and semantics of μCRL. In A. Ponse, C. Verhoef, and S. F. M. van Vlijmen, editors, *Algebra of Communicating Processes '94*, Workshops in Computing Series, pages 26–62. Springer-Verlag, 1995.

[12] J. F. Groote, M. A. Reniers, J. van Wamel, and M. B. van der Zwaag. Completeness of timed μCRL. Technical Report SEN-R0034, CWI, 2000.

[13] J. F. Groote and J. Springintveld. Focus points and convergent process operators. A proof strategy for protocol verification. Logic Group Preprint Series 142, Department of Philosophy, Utrecht University, 1995.

[14] J. F. Groote and J. Springintveld. Algebraic verification of a distributed summation algorithm. Technical Report CS-R9640, CWI, Amsterdam, 1996.

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

[16] A. S. Klusener. The silent step in time. In W. R. Cleaveland, editor, *Proceedings of CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*, pages 421–435. Springer-Verlag, 1992.

[17] A. S. Klusener. *Models and axioms for a fragment of real time process algebra*. PhD thesis, Eindhoven University of Technology, 1993.

[18] R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.

[19] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J. C. M. Baeten and J. W. Klop, editors, *Proceedings of CONCUR'90*, volume 458 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 1990.

[20] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58(1/3):249–261, 1998.

[21] C. Shankland and M. B. van der Zwaag. The tree identify protocol of IEEE 1394 in μCRL. *Formal Aspects of Computing*, 10:509–531, 1998.