Models and logics for process algebra
van der Zwaag, M.B.

Citation for published version (APA):
Timed Cones and Foci

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.

1. Introduction

Time often plays a crucial role in process behavior. For this reason, process algebras such as CCS [71], CSP [57] and ACP [15, 35] have been extended with some notion of time [73, 76, 3, 4]. 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 \( \mu \text{CRL} \) [52] with time [47, 77]. 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 [53] introduced a method to prove that the transition systems generated by two \( \mu \text{CRL} \) process equations—one called the implementation, the other the specification—are (untimed) branching bisimilar [44]. 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 for example [37, 50, 54, 78].

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 modelling timed behavior. In timed \( \mu \text{CRL} \), 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 [7] in case a discrete time domain is chosen. In case of a continuous time domain, our definition corresponds to the notion of timed branching bisimilarity in the setting of real time ACP with urgent actions [63]. 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 [62, 34], 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 [53] 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 $A_\tau$, and $u, v, \ldots$ to denote arbitrary elements of $T$.

A timed transition system is a triple $(S, T_r, U)$, where

(i) $S$ is a nonempty set of states,
(ii) $T_r \subseteq S \times A_\tau \times T \times S$ is a set of transitions, and
(iii) $U \subseteq S \times T$ is a delay relation, such that always

- if $u < v$ and $U(s, v)$, then $U(s, u)$, and
- if $T r(s, a, u, r)$, 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 generalized τ-step relation $\xrightarrow{u}$ as the reflexive transitive closure of the relation $\xrightarrow{u}$.
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 $sR_u r$ for $R(s, u, r)$.

**Definition 2.1.** A relation $R \subseteq S \times T \times S$ is a **timed branching bisimulation** over the timed transition system $(S, Tr, U)$, if whenever $sR_u r$, then also $rR_u s$, and the following conditions hold:

(i) If $s \xrightarrow{a} u s'$ for some $a$ and $s'$, then either
- $a = \tau$ and $s'R_u r$, or
- there are $r'$ and $r''$ such that $r \xrightarrow{u} r'' \xrightarrow{a} u r'$ and $sR_u r''$ and $s'R_u r'$.

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

The states $s$ and $r$ are timed branching bisimilar at $u$, if there exists a timed branching bisimulation $R$ with $sR_u r$. 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 for example [35, 44] 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 "$\tau$-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 $s_0 u_0 s_1 u_1 s_2 u_2 \ldots$ such that $s = s_0$, $u \leq u_0$, and, for all $i \geq 0$, $s_i \xrightarrow{u_i} s_{i+1}$ and $u_i \leq u_{i+1}$.

### 3. Process Structures

We introduce (timed) process structures, that are represented by timed $\mu$CRL linear process equations. We first fix the action names used, and auxiliary sets $D_a$, for the parameters of actions. Let $\delta \not\in A_\tau$ model a case of inaction, and let $Act$ be a collection of functions $a : D_a \rightarrow A$, where the $D_a$ are nonempty sets. We require that $\tau, \delta \not\in Act$ and that $range(a)$ and $range(b)$ are disjoint for all distinct $a, b \in Act$. We write $Act_\tau$ for the set $Act \cup \{\tau\}$, and $Act_{\delta \tau}$ for $Act \cup \{\tau, \delta\}$. 
Timed Cones and Foci

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, that is, 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. We define a process structure $\mathcal{P}$ over $\text{Act}$ as a tuple $(D, E, f, b, g)$, where

- $D$ is a nonempty set called the state space of $\mathcal{P}$,
- $E$ is a nonempty set of environments,
- $f$ is a collection of functions $f_a : D \times E \times T \rightarrow D_a$, one for every $a \in \text{Act}$,
- $b$ is a collection of relations $b_a \subseteq D \times E \times T$, one for every $a \in \text{Act}_\delta$,
- $g$ is a collection of functions $g_a : D \times E \times T \rightarrow D$, one for every $a \in \text{Act}_\tau$.

The functions $f_a, g_a$ may be partial, but must be defined on the elements of $b_a$. Below we give an example of a process structure.

Process structures are the objects that are represented by timed $\mu$CRL linear process equations; we have abstracted from the $\mu$CRL 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 $\mu$CRL 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 $\mathcal{P}$ written as above. With $\mathcal{P}$ we associate a transition system as follows.

**Definition 3.1.** The timed transition system $\text{ttt}(\mathcal{P})$ is given by

$$\text{ttt}(\mathcal{P}) = (D, \text{Tr}, U),$$

where $\text{Tr}$ and $U$ are the smallest sets such that, for all $d \in D$, $a \in \text{Act}_\delta$, $e \in E$ and $u, v \in T$, the following hold:

1. If $b_a(d, e, u)$ and $a \neq \delta$, then $\text{Tr}(d, a, u, g_a(d, e, u))$, where $a = \tau$, if $a = \tau$, and $a = a(f_a(d, e, u))$ otherwise.
2. If $b_a(d, e, u)$ and $v \leq u$, then $U(d, v)$.

Observe that the environments may be used to describe nondeterminism: it may be that $b_a(d, e_1, u)$ and $b_a(d, e_2, u)$ for environments $e_1$ and $e_2$, while $f_a(d, e_1, u) = f_a(d, e_2, u)$ and $g_a(d, e_1, u) \neq g_a(d, e_2, u)$.

The relation $b_\delta$ 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 behavior as follows: if $b_\delta(d, e, u)$, then $U(d, u)$, that is, 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 3.2. The delay condition \( DC_P \subseteq D \times T \) of process structure \( P \) is defined as follows: \( DC_P(d, u) \) if and only if \( b_a(d, e, v) \) and \( u \leq v \) for some \( a \in \text{Act}_\delta, e \in E \) and \( v \in T \).

Observe that \( DC_P(d, u) \) if and only if \( U(d, u) \) in \( tts(P) \). So, if \( DC_P(d, u) \), then in state \( d \) time may pass at least until time \( u \).

Definition 3.3. The focus condition \( FC_P \subseteq D \times T \times T \) of process structure \( P \) is defined as follows: \( FC_P(d, u, v) \) 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 \( \tau \)-steps between \( u \) and \( v \) in \( tts(P) \). An untimed focus point is simply a state without outgoing \( \tau \)-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 3.4. A relation \( I \subseteq D \times T \) is an invariant of \( P \), if whenever \( I(d, u) \) and \( b_a(d, e, v) \) and \( u \leq u' \leq v \), then \( I(d, u') \) and, if \( a \neq \delta \), also \( I(g_a(d, e, v), v) \).

If \( I \) is an invariant of \( P \) with \( I(d, u) \), then \( I \) will remain true in all states that can be reached by action steps or by the passage of time: we find 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 \( I(d, v) \).

Example: Buffers. We give a process structure that models the behavior of a buffer with capacity one. Between the reading and the sending of a message, there is a fixed time delay \( \Delta \). Let \( M \) be a nonempty set of messages.

Let \( \text{Act} = \{ r, s \} \) and \( D_r = D_s = M \). An action \( s(m) \) models the sending of message \( m \), and \( r(m) \) models the receiving of message \( m \).

A buffer \( P \) is the process structure \((D, E, f, b, g)\) over \( \text{Act} \) with state space

\[
D = \{ \lambda \} \cup (M \times T),
\]

\[
E = M, \text{ and } f, b, g \text{ defined as follows:}
\]

\[
f_s((m, v), e, u) = m,
\]

\[
f_\tau(\lambda, e, u) = e,
\]

\[
b_s(d, e, u) \iff d = (m, u - \Delta),
\]

\[
b_\tau(d, e, u) \iff d = \lambda,
\]

\[
b_a = \emptyset \quad \text{if } a \in \{ \delta, \tau \},
\]

\[
g_s(d, e, u) = \lambda,
\]

\[
g_\tau(d, e, u) = (e, u).
\]
A buffer in state $\lambda$ is empty and ready to read any message at any time; this is true because $b_\tau(\lambda, m, u)$ 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 $m$, $tts(P)$ has transitions

$$
\lambda \xrightarrow{t(m)}_u (m, u) \xrightarrow{s(m)}_{u+\Delta} \lambda.
$$

Also observe that $DC_P(\lambda, u)$ for all $u$, and $DC_P((m, v), u)$ for all $u \leq v + \Delta$.

4. Cones and Foci

In the untimed technique, a focus point is a state that has no outgoing $\tau$-transitions. The idea is that, in convergent transition systems,\footnote{In [53] also an extended technique is presented that deals with $\tau$-divergence using the fairness principle CFAR.} every state of the implementation must, after a number of $\tau$-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 $\tau$-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 visualization 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 a set of action declarations that are written as before, and let

$$
P = (D, E, f, b, g) \quad \text{and} \quad Q = (D', E, f', b', g')
$$

be process structures over $Act$ with $b'_\tau = \emptyset$; so the transition system $tts(Q)$ does not have $\tau$-transitions. 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 $a \in Act, e \in E$ and $v \in T$, the following conditions hold.

1. The state $d$ is convergent at $u$ in $tts(P)$.
2. If $b_\tau(d, e, u)$, then $h(d) = h(g(d, e, u))$ and $DC_Q(h(d), u)$.
   If a state can do a $\tau$-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 $b_a(d, e, u)$, then $b'_a(h(d), e, u)$.
   If a state has an $a$-step at time $u$, then its image also has some $a$-step at time $u$.\footnote{In [53] also an extended technique is presented that deals with $\tau$-divergence using the fairness principle CFAR.}
(4) If \( b'_{\alpha}(h(d), e, v) \) and \( u \leq v \) and \( FC_{P}(d, u, v) \), then \( b_{\alpha}(d, e, v) \).

If the image of \( d \) has an \( a \)-step at some time \( v \) later than \( u \), and \( d \) is a focus point between \( u \) and \( v \), then \( d \) also has some \( a \)-step at time \( v \).

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

If a state can do some \( a \)-action at time \( u \) for some \( e \), then its image can do the same action at time \( u \).

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

If a state has an \( a \)-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 \( b_{\beta}(d, e, u) \), 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 so-called general equality theorem of [53].

**Theorem 4.1.** Let \( P \) and \( Q \) be written as above. If \( I \) is an invariant of \( P \) and \( h : D \rightarrow D' \) is a mapping such that \( I(d, u) \) implies \( C_{h}(d, u) \) for all \( d \) and \( u \), then \( d_{0} \) and \( h(d_{0}) \) are timed branching bisimilar at \( u_{0} \) for any \( d_{0} \) and \( u_{0} \) with \( I(d_{0}, u_{0}) \).

**Proof.** Let \( I \) be an invariant of \( P \), and let \( h \) be a state mapping that satisfies the matching criteria for all \( d \) and \( u \) with \( I(d, u) \).

Assume, without loss of generality, that \( D \) and \( D' \) are disjoint. So the union of \( tts(P) \) and \( tts(Q) \) is \( (D'' \times T \times U) \), 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 \( xR_{u}y \); by definition of \( R \) either \( x = h(y) \) or \( y = h(x) \), and in both cases also \( yR_{u}x \).

**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 \( C_{h}(x, u) \).
• If $a = \tau$, then $b_\tau(x, e, u)$ and $x' = g_\tau(x, e, u)$, for some $e$, by Definition 3.1. By criterion (2) we have $h(x) = h(x')$, so $x' R_u y$, by definition of $R$, as required.

• If $a \neq \tau$, then we find, by Definition 3.1, that $b_a(x, e, u)$, that $x' = g_a(x, e, u)$, and that $a = a(f_a'(x, e, u))$, for some $a$ in $Act$ and $e$ in $E$. It follows from criterion (3) that $b_a'(h(x), e, u)$, from criterion (5) that $a(f_a'(h(x), e, u)) = a$, and from criterion (6) that $h(x) = g_a'(h(x), e, u)$. So we know by Definition 3.1 that $h(x) \xrightarrow{a} u h(x')$ and by definition of $R$ we have $x' R_u h(x')$, which was to be shown.

Second, consider the case where $x = h(y)$. By the assumption that $b_\tau = \emptyset$, we see that $a \neq \tau$. So, for some $a$ in $Act$ and $e$ in $E$, we have that $b_a(x, e, u)$ and $x' = g_a(x, e, u)$ and $a = a(f_a'(x, e, u))$. Now consider $y$. By definition of $R$, we know $I(y, u)$; so also $C_h(y, u)$. By criterion (1) there is a $y'$ such that $y \xrightarrow{u} y'$ and there is no $\tau$-step from $y'$ at $u$; so $FC\rho(y', u, u)$. As the invariant and hence the matching criteria hold for all states on this $\tau$-path, we can repeatedly apply criterion (2) and Definition 3.1 to get $h(y') = h(y) = x$. We have $b_a(y', e, u)$ by criterion (4), $a = a(f_a(y', e, u))$ by criterion (5), and by criterion (6) that $h(g_a(y', e, u)) = x'$. By Definition 3.1, we have $y \xrightarrow{u} y' \xrightarrow{a} u g_a(y', e, u)$, and by definition of $R$ we find the required $y' R_u x$ and $g_a(y', e, u) R_u x'$.

**Delay behavior:** Suppose that $u < v$ and $U(x, v)$ for some $v$. 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 3.1, we know that $b_a(x, e, v')$ for some $a$, $e$ and $v' \geq v$. So $I(x, v)$ and $I(x, v')$, and therefore $C_h(x, v')$. Case distinction: if $a = \tau$, then $DC\rho(y, v')$ by criterion (2); if $a = \delta$, then $DC\rho(y, v')$ by criterion (7); else $DC\rho(y, v')$ by criterion (3). So $DC\rho(y, v')$. By Definition 3.1, we know that $U(y, v)$, and by definition of $R$ we find that $x R_u y$, as was to be shown.

Second, consider the case with $x = h(y)$. By Definition 3.1, we find $b_a'(x, e, v')$ and $v \leq v'$ for some $a$, $e$ and $v'$. Now consider $y$. It holds that $I(y, u)$, and hence $C_h(y, u)$, by definition of $R$. By criterion (1) there are, for some $n \geq 0$, $y_i, u_i$ with $u = u_0$, $y = y_0$, $y_i \xrightarrow{u_i} y_{i+1}$ for all $i \leq n$, and, $u_i < u_{i+1}$ for all $i < n$, such that $FC\rho(y_{n+1}, u_n, v')$ and $u_n \leq v'$. We see that the invariant holds for all intermediate states on this $\tau$-idle-path. Therefore we can by repeatedly applying criterion (2) and Definition 3.1 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 $i < n$ with $U(y_{i+1}, v)$, which was to be demonstrated.
5. Example: Two Serial Buffers

If $u_n < v$, then, if $a \neq \delta$, we find that $b_a(y_{n+1}, e, v')$ using criterion (3), and hence $U(y_{n+1}, v')$ by Definition 3.1. If $a = \delta$, then it follows from criterion (8) that $DCP(y_{n+1}, v')$ and hence $U(y_{n+1}, v')$. We see that also $U(y_{n+1}, v)$, which was to be demonstrated.

We conclude that $R$ is a timed branching bisimulation over the transition system $(D'', Tr, U)$. From the definition of $R$ and the assumption $I(d_0, u_0)$, it follows 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 the example in Section 3. 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 $\tau$ 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. 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 $\{\lambda\} \cup (M \times T)$ (the state space of single buffers) by the set $T_\lambda = T \cup \{\lambda\}$.

The implementation is the process structure given by

$$P = (D, M, f, b, g),$$

with state space $D = T_\lambda \times T_\lambda$, 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. The $b$ relations are defined by

$$b_s((d_1, d_2), u) \Leftrightarrow d_2 = u - \Delta \text{ and } \beta_1(u),$$

$$b_\tau((d_1, d_2), u) \Leftrightarrow d_1 = \lambda \text{ and } \beta_2(u),$$

$$b_\tau((d_1, d_2), u) \Leftrightarrow d_1 = u - \Delta \text{ and } d_2 = \lambda,$$

$$b_s = \emptyset,$$

and the $g$ functions by

$$g_s((d_1, d_2), u) = (d_1, \lambda),$$

$$g_\tau((d_1, d_2), u) = (u, d_2),$$

$$g_\tau((d_1, d_2), u) = (\lambda, u).$$

The conditions $\beta_i(u)$, with $i \in \{1, 2\}$, abbreviate $(d_i = \lambda \text{ or } u \leq d_i + \Delta)$. These conditions have to be added in order to avoid timing inconsistencies.
The Specification. The specification is the process structure given by

\[ Q = (D, M, f', b', g') \]

with \( b' \) defined by,

\[ b'_s((d_f, d_s), u) \iff d_f = u - 2\Delta, \]
\[ b'_t((d_f, d_s), u) \iff d_f \neq \lambda \] implies \((d_s = \lambda \text{ and } d_f + \Delta \leq u \leq d_f + 2\Delta)\),

and \( b'_t = b'_s = \emptyset \), and \( g' \) defined by

\[ g'_s((d_f, d_s), u) = (d_s, \lambda), \]
\[ g'_t((d_f, d_s), u) = \begin{cases} (u, \lambda) & \text{if } d_f = \lambda, \\ (d_f, u) & \text{otherwise}. \end{cases} \]

The specification has the same state space as the implementation, but the roles of the constituents of states are different. In a state \((d_f, d_s)\), 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 \( d_f = d_s = \lambda \). An invariant of \( Q \) is that \( d_f = \lambda \) implies \( d_s = \lambda \).

The Verification. We define the state mapping \( h : D \to D \) by

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

The invariant \( I \) of the implementation is defined as follows:

\[ I((d_1, d_2), u) = I_1 \land I_2 \land I_3, \]

where

\[ I_1 : \text{if } d_1 \neq \lambda, \text{ then } u \leq d_1 + \Delta, \]
\[ I_2 : \text{if } d_2 \neq \lambda, \text{ then } d_2 \leq u, \]
\[ I_3 : \text{if } d_1 \neq \lambda \text{ and } d_2 \neq \lambda, \text{ then } d_2 \leq d_1. \]

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 \( a \in Act \) 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 \( b_5 = \emptyset \). The first six criteria are shown as follows.

1. Clearly the implementation is convergent: every \( \tau \)-step leads to a state where no further \( \tau \)-step is enabled.
(2) Suppose that $b_t(d, u)$. We show that $h(d)$ equals $h(g_t(d, u))$ and that $DC_Q(h(d), u)$.

By definition of $b_t$, we see that $d_1 = u - \Delta$ and $d_2 = \lambda$, and hence $h(d) = d$ by definition of $h$. Also $h(g_t(d, u)) = h(\lambda, u) = (u - \Delta, \lambda) = d$. From $b'_s(h(d), d_1 + 2\Delta)$, it follows that $DC_Q(h(d), u)$.

(3) Suppose that $b_\lambda(d, u)$. We show that $b'_n(h(d), u)$.

First, if $a = s$, then $d_2 = u - \Delta$ by definition of $b_s$. We must show that $d_f = u - 2\Delta$. From $d_2 \neq \lambda$, we see by definition of $h$ that $d_f = d_2 - \Delta$. With $d_2 = u - \Delta$, we get the required $d_f = u - 2\Delta$.

Second, consider the case with $a = r$. Observe that $d_1 = \lambda$ and $\beta_2(u)$ by definition of $b_r$. If $d_2 = \lambda$, then $d_f = d_1 = \lambda$ by definition of $h$, and hence $b'_t((d_f, d_s), u)$. Else, if $d_2 \neq \lambda$, then $d_f = d_2 - \Delta$ and $d_s = d_1 = \lambda$ by definition of $h$. We see that $b'_t(h(d), u)$, 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 $b'_d(h(d), v)$ and $u \leq v$ and $FC_P(d, u, v)$. We must show that $b_a(d, v)$.

First, we look at the case with $a = s$. We find $d_f = v - 2\Delta$ by definition of $b'_s$.

If $d_s = \lambda$, then by definition of $h$ we see that one of the following cases applies.

- $d_2 = d_s = \lambda$ and $d_1 = d_f = v - 2\Delta$. Since $b_t(d, d_1 + \Delta)$ and $I_1(d, u)$, we see that this case violates assumption $FC_P(d, u, v)$.
- $d_1 = d_s = \lambda$ and $d_f = v - 2\Delta = d_2 - \Delta$. Then $v = d_2 + \Delta$, so indeed $b_s(d, v)$.

If $d_s \neq \lambda$, 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 $b_s(d, v)$ follows from $\beta_1(v)$ 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 $I_3(d, u)$.

This finishes the case with $a = s$.

Now assume that $a = r$. We must show that $b_t(d, v)$.

If $d_f = \lambda$ then $d_1 = d_2 = \lambda$ by definition of $h$, and hence $b_t(d, v)$.

Next, if $d_f \neq \lambda$, then $d_s = \lambda$ and $d_f + \Delta \leq v \leq d_f + 2\Delta$, by definition of $b'_t$. By definition of $h$, we know that one of the following two cases applies.

- $d_1 = \lambda$ and $d_2 \neq \lambda$ and $d_f = d_2 - \Delta$. Observe that it follows from $d_f = d_2 - \Delta$ and $v \leq d_f + 2\Delta$, that $v \leq d_2 + \Delta$, and hence $\beta_2(v)$, which implies the required $b_t(d, v)$.
- $d_2 = \lambda$ 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 $b_t(d, d_1 + \Delta)$ and, by $I_1(d, u)$, $u \leq d_1 + \Delta$.

(5) Trivial, since $M$ is a singleton set.

(6) Suppose that $b_a(d, u)$. We show that $h(g_a(d, u)) = g'_a(h(d), u)$.
If $a = s$, then $d_2 \neq \lambda$, and
\[ h(g_s(d, u)) = h(d_1, \lambda) = (d_1, \lambda) \]
\[ = g'_s((d_2 - \Delta, d_1), u) = g'_s(h(d), u). \]

If $a = r$, then $d_1 = \lambda$. If $d_2 = \lambda$, then
\[ h(g_r((\lambda, \lambda), u)) = h(u, \lambda) = (u, \lambda) \]
\[ = g'_r((\lambda, \lambda), u) = g'_r(h(\lambda, \lambda), u) = g'_r(h(d), u). \]

If $d_2 \neq \lambda$, then
\[ h(g_r(d, u)) = h(u, d_2) = (d_2 - \Delta, u) \]
\[ = g'_r((d_2 - \Delta, \lambda), u) = g'_r(h(\lambda, d_2), u) = g'_r(h(d), u). \]

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 for example the start state $d = (\lambda, \lambda)$. Then also $h(d) = (\lambda, \lambda)$. It is easily seen that $I(d, u)$ for all time elements $u$, so $d$ and $h(d)$ are timed branching bisimilar at any $u$. 