A new equivalence for processes with timing

A New Equivalence for Processes with Timing
With an Application to Protocol Verification

J.C.M. Baeten[1], C.A. Middelburg[1,2], and M.A. Reniers[1]

[1] Computing Science Department, Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
[2] Department of Philosophy, Utrecht University,
P.O. Box 80126, 3508 TC Utrecht, the Netherlands
{josb,keesm,michelr}@win.tue.nl

Abstract. We propose a variant of the version of branching bisimulation equivalence for processes with discrete relative timing from Baeten, Bergstra, and Reniers. We show that this new equivalence allows for the functional correctness of the PAR protocol as well as its performance properties to be analyzed. The new equivalence still coincides with the original version of branching bisimulation equivalence from Glabbeek and Weijland in the case without timing.

Keywords: process algebra, discrete relative timing, branching bisimulation, PAR protocol, functional correctness, performance properties.

1998 CR Categories: C.2.2, D.1.3, D.2.4, F.1.2, F.3.1.

1 Introduction

The PAR (Positive Acknowledgement with Retransmission) protocol [1] is a communication protocol which is based on time-outs of positive acknowledgements. Timing is essential for the correct behaviour of the PAR protocol: the protocol behaves only correctly if the time-out time of the sender, i.e. the time after which it retransmits a datum for which it is still awaiting an acknowledgement, is longer than the time that a complete protocol cycle takes. There have been various attempts to describe this protocol using process algebra without timing. In most attempts, e.g. the attempt presented in [2], it is excluded that the sender times out too early by inhibiting it so long as it does not lead to deadlock. However, this boils down to assuming an oracle that informs the sender whenever a datum or an acknowledgement has got lost. We consider such attempts, which resort to artificial tricks, unsatisfactory. In other attempts, the premature time-out of the sender is not excluded at all. Those attempt are unquestionable unsatisfactory as well.

In this paper, we begin by using the version of process algebra with discrete relative timing and abstraction from [3] to describe the PAR protocol and to show that the protocol behaves correctly if the time-out time of the sender is longer than the time that a complete protocol cycle takes. In order to achieve this result, it is necessary to take the timing of actions into account in all calculations

that must be done before abstraction from the actions that should be regarded as internal can be applied. From the point where abstraction from those actions can be applied, the timing of actions is no longer relevant. Actually, many internal actions can only be removed after abstraction from the timing of actions.

It would facilitate performance analysis if most internal actions could be removed without preceding abstraction from the timing of actions. The axioms used for the removal of internal actions are based on the version of branching bisimulation equivalence for processes with discrete relative timing introduced in [4], known as rooted branching tail bisimulation equivalence. What is needed in the case of performance analysis, is an equivalence that is coarser than that version of branching bisimulation equivalence. Therefore, we propose a coarser equivalence, with a plausible motivation, which still coincides with the original version of branching bisimulation equivalence from [5] in the case without timing. We show that in the case of the PAR protocol all internal actions that hinder performance analysis can be removed without preceding abstraction from the timing of actions if we use the axioms based on the new equivalence.

The structure of this paper is as follows. First of all, a brief summary of the version of process algebra with discrete relative timing primarily used in this paper is given (Section 2). Next, we use this version to describe the PAR protocol (Section 3) and to analyze it (Sections 4 and 5). After that, we introduce a variant of the version of branching bisimulation equivalence for processes with discrete relative timing from [4] and the axioms corresponding with the new equivalence (Sections 6 and 7). Then, we continue the analysis using the new axioms (Section 8). Finally, some concluding remarks are made (Section 9).

2 Process Algebraic Preliminaries

The version of process algebra with discrete relative timing used in Sections 3, 4 and 5 is $\mathrm{ACP}^{\mathrm{drt}}$ [6] extended with abstraction and guarded recursion. This section gives a brief summary. For reference, the axioms are given in Appendix A. For a comprehensive overview of $\mathrm{ACP}^{\mathrm{drt}}$, and its extension with abstraction and guarded recursion, the reader is referred to [3].

$\mathbf{ACP}^{\mathbf{drt}}$

In the case of $\mathrm{ACP}^{\mathrm{drt}}$, timing is relative to the time at which the preceding action is performed and time is measured on a discrete time scale. Measuring time on a discrete time scale means that time is divided into time slices and timing of actions is done with respect to the time slices in which they are performed. Roughly speaking, $\mathrm{ACP}^{\mathrm{drt}}$ is ACP [7, 8] extended with three operators to deal with timing: relative delay, relative time-out, and relative initialization. The first operator is a basic one and the latter two operators are useful auxiliary ones. $\mathrm{ACP}^{\mathrm{drt}}$ has the following constants and operators:

- the *undelayable action* a, written $\underline{a}$, is the process that performs action a in the current time slice and then terminates successfully;

- the *undelayable deadlock*, written $\underline{\underline{\delta}}$, is the process that is neither capable of performing any action in the current time slice nor capable of idling till the next time slice;
- the *deadlocked process*, written δ, is the process that can be viewed as (a trace of) a process that has deadlocked before the current time slice;
- the *relative delay* of p for n time slices, written $\sigma^n_{\mathsf{rel}}(p)$, is the process that idles till the nth-next time slice and then behaves like p;
- the *alternative composition* of p_1 and p_2, written $p_1 + p_2$, is the process that behaves either like p_1 or like p_2, but not both;
- the *sequential composition* of p_1 and p_2, written $p_1 \cdot p_2$, is the process that first behaves like p_1, but when p_1 terminates successfully it continues by behaving like p_2;
- the *parallel composition* of p_1 and p_2, written $p_1 \parallel p_2$, is the process that proceeds with p_1 and p_2 in parallel;
- the *left merge* of p_1 and p_2, written $p_1 \lfloor\!\lfloor p_2$, is the same as $p_1 \parallel p_2$ except that $p_1 \lfloor\!\lfloor p_2$ starts with performing an action of p_1;
- the *communication merge* of p_1 and p_2, written $p_1 \mid p_2$, is the same as $p_1 \parallel p_2$ except that $p_1 \mid p_2$ starts with performing an action of p_1 and an action of p_2 synchronously;
- the *encapsulation* of p with respect to a set of actions H, written $\partial_H(p)$, keeps p from performing actions in H;
- the *relative time-out* of p after n time slices, written $\upsilon^n_{\mathsf{rel}}(p)$, keeps p from idling till the nth-next time slice;
- the *relative initialization* of p after n time slices, written $\overline{\upsilon}^n_{\mathsf{rel}}(p)$, keeps p from performing actions before the nth-next time slice.

We use the notations $\sum_{i\in\mathcal{I}} t_i$ and $\Vert_{i\in\mathcal{I}} t_i$, where $\mathcal{I} = \{i_1, \ldots, i_n\}$, for the alternative composition $t_{i_1} + \ldots + t_{i_n}$ and the parallel composition $t_{i_1} \parallel \ldots \parallel t_{i_n}$, respectively. We further use the convention that $\sum_{i\in\mathcal{I}} t_i$ and $\Vert_{i\in\mathcal{I}} t_i$ stand for δ if $\mathcal{I} = \emptyset$.

In most applications, communication is synchronous communication between two processes. This kind of communication is called *handshaking communication*. An important result is the following expansion theorem, which is useful in the elimination of parallel compositions in terms of $\mathrm{ACP}^{\mathrm{drt}}$ in the case where all communication is handshaking communication.

Theorem (Expansion Theorem). *In* $\mathrm{ACP}^{\mathrm{drt}}$ *with the axioms of standard concurrency and the handshaking axiom, the following equation is derivable for all* $n > 2$*:*

$$\begin{aligned} &x_1 \parallel \ldots \parallel x_n \\ &= \sum_{1\le i\le n} x_i \lfloor\!\lfloor \left(\Vert_{\substack{1\le j\le n,\\ j\ne i}} x_j \right) + \sum_{\substack{1\le i\le n,\\ i<j\le n}} (x_i \mid x_j) \lfloor\!\lfloor \left(\Vert_{\substack{1\le k\le n,\\ k\ne i, k\ne j}} x_k \right) . \end{aligned}$$

The axioms of standard concurrency and the handshaking axiom can be found in Appendix A.

We introduce some standardized terminology and notation for handshaking communication. Processes send, receive and communicate data at *ports*. If a port is used for communication between two processes, it is called *internal*. Otherwise, it is called *external*. We write:

- $s_i(d)$ for the action of sending datum d at port i;
- $r_i(d)$ for the action of receiving datum d at port i;
- $c_i(d)$ for the action of communicating datum d at port i.

The action $c_i(d)$ is the action that is left when $s_i(d)$ and $r_i(d)$ are performed synchronously.

Abstraction

In order to deal with abstraction from certain actions, we have an additional constant and an additional operator:

- the *undelayable silent step*, written $\underline{\tau}$, is the process that performs an internal action, i.e. an action that is considered to be unobservable, in the current time slice;
- the *abstraction* of p with respect to a set of actions I, written $\tau_I(p)$, renames each action of p that is contained in I into an internal action.

Guarded Recursion

In order to cover processes that may never terminate, it is essential to have an additional way of combining processes: guarded recursion.

A set of equations of the form $X = t$, where X is a variable and t is a term that contains only variables that are among the variables on the left-hand sides of the equations in the set, is called a recursive specification. A recursive specification E satisfies the guardedness criterion if all occurrences of variables in the right-hand sides of the equations in E are preceded by an action other than τ or a delay of at least one time slice. A recursive specification that satisfies the guardedness criterion, called a *guarded recursive specification*, has a unique solution.

For each guarded recursive specification E and each variable X that occurs on the left-hand side of an equation in E, we introduce a constant $\langle X|E\rangle$ which is interpreted as the unique solution for X of E. We usually write X for $\langle X|E\rangle$ if E is clear from the context. In those cases, it should also be clear from the context that we use X as a constant.

The axioms for recursion are known as RDP (Recursive Definition Principle) and RSP (Recursive Specification Principle). For a fixed recursive specification E, RDP expresses that the constants $\langle X|E\rangle$ make up a solution of E and RSP expresses that this solution is the only one.

Let t be a term and E be a guarded recursive specification. Then we use the notation $\langle t|E\rangle$ for t with, for each variable X that occurs on the left-hand side of an equation in E, all occurrences of X in t replaced by $\langle X|E\rangle$.

A system is often described by a term of the form $\partial_H(\langle X_1|E_1\rangle \| \ldots \| \langle X_n|E_n\rangle)$, i.e. as the encapsulated parallel composition of a number of processes that are described by guarded recursive specifications. The first step in analyzing such a system is usually the extraction of a guarded recursive specification of the system from the term describing it. This involves mere algebraic calculations, in which the expansion theorem plays an important part, and finally application of RSP. In case the functional correctness is analyzed, we can proceed by extracting a guarded recursive specification from a term of the form $\tau_I(\langle X|E\rangle)$ or the form $\tau_I(\pi_{\mathsf{tf}}(\langle X|E\rangle))$, where E is the result of the first step and π_{tf} is the time free projection operator described below. This involves again application of RSP.

Other extensions

The *time free projection* of p, written $\pi_{\mathsf{tf}}(p)$, is the process p made time free. For any time free process, the following holds: it is always capable of idling till a future time slice and it is never bound to idle till a future time slice. Thus, with the time free projection operator, we abstract from the timing of actions.

Let P be an expression, possibly containing variable i, such that $P[n/i]$ (P with n substituted for i) represents a process for all $n \in \mathbb{N}$; and let $K \subseteq \mathbb{N}$. Then the *summation* $\sum_{i\in K} P$ is the process that behaves like one of the processes $P[n/i]$ for $n \in K$. Hence, summation is a form of alternative composition over a set of alternatives that may be countably infinite.

3 Description of the PAR Protocol

We consider a simple version of the communication protocol known as the *PAR* protocol. The sender waits for a positive acknowledgement before a new datum is transmitted. If an acknowledgement is not received within a complete protocol cycle, the old datum is retransmitted. In order to avoid duplicates due to retransmission, data are labeled with an alternating bit from $B = \{0, 1\}$. The configuration of the PAR protocol is shown in Fig. 1 by means of a connection diagram.

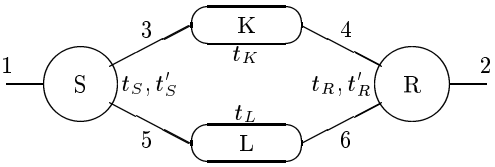

Fig. 1. Connection diagram for PAR protocol

We have a sender process S, a receiver process R, and two channels K and L. The process S waits until a datum d is offered at an external port (port 1). When a datum is offered at this port, S consumes it, packs it with an alternating bit b

in a frame (d,b), and then delivers the frame at an internal port used for sending (port 3). Next, S waits until an acknowledgement ack is offered at an internal port used for receiving (port 5). When the acknowledgement does not arrive within a certain time period, S delivers the same frame again and goes back to waiting for an acknowledgement. When the acknowledgement arrives within that time period, S goes back to waiting for a datum. The process R waits until a frame (d,b) is offered at an internal port used for receiving (port 4). When a frame is offered at this port, R consumes it, unpacks it, and then delivers the datum d at an external port (port 2) if the alternating bit b is the right one and in any case an acknowledgement ack at an internal port used for sending (port 6). After that, R goes back to waiting for a frame, but the right bit changes to $(1-b)$ if the alternating bit was the right one. The processes K and L pass on frames from an internal port of S to an internal port of R and acknowledgements from an internal port of R to an internal port of S, respectively. Because the channels are supposed to be unreliable, they may produce an error instead of passing on frames or acknowledgements. The times t_S, t_R, t_K, t_L are the times that it takes the different processes to pack and deliver, to unpack and deliver or simply to deliver what they consume. The time t'_S is the time-out time of the sender, i.e., the time after which it retransmits a datum in the case where it is still waiting for an acknowledgement. The time t'_R is the time that it takes the receiver to produce and deliver an acknowledgement.

We assume that the times $t_S, t_R, t_K, t_L, t'_S, t'_R$ are non-zero. We also assume a finite set of data D. Let $F = D \times B$ be the set of frames. For $d \in D$ and $b \in B$, we write d,b for the frame (d,b). We use the standardized notation for handshaking communication introduced in Section 2. The recursive specification of the sender consists of the following equations:

$$S = S_0 \ ,$$

$$S_b = \sum_{d \in D} \underline{\underline{r_1(d)}} \cdot \sigma_{\mathsf{rel}}^{t_S}(SF_{d,b}) + \sigma_{\mathsf{rel}}^{1}(S_b)$$

(for every $b \in B$),

$$SF_{d,b} = \underline{\underline{s_3(d,b)}} \cdot \left(\sum_{k<t'_S} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{r_5(ack)}}) \cdot S_{1-b} + \sigma_{\mathsf{rel}}^{t'_S}(SF_{d,b}) \right)$$

(for every $d \in D$ and $b \in B$).

The recursive specification of the receiver consists of the following equations:

$$R = R_0 \ ,$$

$$R_b = \sum_{d \in D} \underline{\underline{r_4(d,b)}} \cdot \sigma_{\mathsf{rel}}^{t_R}(\underline{\underline{s_2(d)}}) \cdot \sigma_{\mathsf{rel}}^{t'_R}(\underline{\underline{s_6(ack)}}) \cdot R_{1-b} + \sum_{d \in D} \underline{\underline{r_4(d,1-b)}} \cdot \sigma_{\mathsf{rel}}^{t'_R}(\underline{\underline{s_6(ack)}}) \cdot R_b + \sigma_{\mathsf{rel}}^{1}(R_b)$$

(for every $b \in B$).

Each of the channels is recursively defined by a single equation:

$$K = \sum_{f \in F} \underline{\underline{r_3(f)}} \cdot \left(\sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{s_4(f)}}) + \sum_{k \leq t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \right) \cdot K + \sigma_{\mathsf{rel}}^{1}(K) \ ,$$

$$L = \underline{\underline{r_6(ack)}} \cdot \left(\sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{s_5(ack)}}) + \sum_{k \leq t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \right) \cdot L + \sigma_{\mathsf{rel}}^{1}(L) \ .$$

The whole system is described by the following term:

$$\partial_H(S \parallel K \parallel L \parallel R) \ ,$$

where

$$H = \{s_i(f), r_i(f) \mid i \in \{3,4\}, f \in F\} \cup \{s_i(ack), r_i(ack) \mid i \in \{5,6\}\} \ .$$

This protocol is only correct if the time-out time t'_S is longer than a complete protocol cycle, i.e., if $t'_S > t_K + t_R + t'_R + t_L$. If the time-out time is shorter than a complete protocol cycle, the time-out is called premature. In that case, while an acknowledgement is still on the way, the sender will retransmit the current frame. When the acknowledgement finally arrives, the sender will treat this acknowledgement as an acknowledgement of the retransmitted frame. However, an acknowledgement of the retransmitted frame may be on the way. If the next frame transmitted gets lost and the latter acknowledgement arrives, no retransmission of that frame will follow and the protocol will fail.

4 Analysis of the PAR Protocol: Expansion

We now start to analyze the PAR protocol described in Section 3. We will use a technique similar to the basic one used in the case without timing.

We first gave guarded recursive specifications of the sender process S, the receiver process R and the channel processes K and L, and then described the whole PAR protocol by the term $\partial_H(S \parallel K \parallel L \parallel R)$. Because all communication is handshaking communication, the expansion theorem for $\mathrm{ACP}^{\mathrm{drt}}$ (see Section 2) is applicable. By using this expansion theorem and RSP, we are able to give a guarded recursive specification of the whole PAR protocol.

First, we rewrite the recursive specifications of S, R, K and L, using their equations and the axioms of $\mathrm{ACP}^{\mathrm{drt}}$, to ones in a form that is better suited to expansion. We refrain from mentioning for each equation schema that there is an instance for every $d \in D$ and/or $b \in B$.

$$\begin{aligned}
S &= S_0 \ , \\
S_b &= \sum_{d \in D} \underline{\underline{r_1(d)}} \cdot S'_{d,b} + \sigma_{\mathsf{rel}}^{1}(S_b) \ , \\
S'_{d,b} &= \sigma_{\mathsf{rel}}^{t_S}(\underline{\underline{s_3(d,b)}}) \cdot S''_{d,b} \ , \\
S''_{d,b} &= \sum_{k < t'_S} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{r_5(ack)}}) \cdot S_{1-b} + \sigma_{\mathsf{rel}}^{t'_S}(\underline{\underline{s_3(d,b)}}) \cdot S''_{d,b} \ ,
\end{aligned}$$

$$
\begin{aligned}
R &= R_0 \ , \\
R_b &= \sum_{d\in D} \underline{\underline{r_4(d,b)}} \cdot R'_{d,b} + \sum_{d\in D} \underline{\underline{r_4(d,1-b)}} \cdot R''_b + \sigma^1_{\mathsf{rel}}(R_b) \ , \\
R'_{d,b} &= \sigma^{t_R}_{\mathsf{rel}}(\underline{\underline{s_2(d)}}) \cdot R''_{1-b} \ , \\
R''_b &= \sigma^{t'_R}_{\mathsf{rel}}(\underline{\underline{s_6(ack)}}) \cdot R_b \ , \\
\\
K &= \sum_{(d,b)\in D\times B} \underline{\underline{r_3(d,b)}} \cdot K'_{d,b} + \sigma^1_{\mathsf{rel}}(K) \ , \\
K'_{d,b} &= \sigma^{t_K}_{\mathsf{rel}}(\underline{\underline{s_4(d,b)}}) \cdot K + \sum_{k\le t_K} \sigma^k_{\mathsf{rel}}(\underline{\underline{error}}) \cdot K \ , \\
\\
L &= \underline{\underline{r_6(ack)}} \cdot L' + \sigma^1_{\mathsf{rel}}(L) \ , \\
L' &= \sigma^{t_L}_{\mathsf{rel}}(\underline{\underline{s_5(ack)}}) \cdot L + \sum_{k\le t_L} \sigma^k_{\mathsf{rel}}(\underline{\underline{error}}) \cdot L \ .
\end{aligned}
$$

Secondly, we expand the term $\partial_H(S_b \parallel K \parallel L \parallel R_b)$ by repeated application of the expansion theorem. Except in the first step, we expand a subterm of the right-hand side of a previous equation. The subterms concerned are indicated by putting them in square brackets. We remove in each step immediately those alternatives that are known to be equal to $\sigma^n_{\mathsf{rel}}(\underline{\underline{\delta}})$ (for some $n \ge 0$) because of incapability to communicate, encapsulation or timing conflict, provided the removal is justified by the fact that $\sigma^m_{\mathsf{rel}}(t) + \sigma^n_{\mathsf{rel}}(\underline{\underline{\delta}}) = \sigma^m_{\mathsf{rel}}(t)$ is derivable for all closed terms $t \neq \dot{\delta}$ and for all $m, n \ge 0$ such that $m \ge n$. In the expansion, we will use the following abbreviation for every $d \in D$, $b \in B$ and $t > 0$:

$$
S''_{d,b,t} \text{ for } \sum_{k<t} \sigma^k_{\mathsf{rel}}(\underline{\underline{r_5(ack)}}) \cdot S_{1-b} + \sigma^t_{\mathsf{rel}}(\underline{\underline{s_3(d,b)}}) \cdot S''_{d,b} \ .
$$

Again, we refrain from mentioning after each equation schema that there is an instance for every $d \in D$ and/or $b \in B$.

$$
\begin{aligned}
&\partial_H(S_b \parallel K \parallel L \parallel R_b) \\
&= \sum_{d\in D} \underline{\underline{r_1(d)}} \cdot [\partial_H(S'_{d,b} \parallel K \parallel L \parallel R_b)] + \sigma^1_{\mathsf{rel}}([\partial_H(S_b \parallel K \parallel L \parallel R_b)]) \ , \\
&[\partial_H(S'_{d,b} \parallel K \parallel L \parallel R_b)] = \sigma^{t_S}_{\mathsf{rel}}(\underline{\underline{c_3(d,b)}}) \cdot [\partial_H(S''_{d,b} \parallel K'_{d,b} \parallel L \parallel R_b)] \ , \\
\\
&[\partial_H(S''_{d,b} \parallel K'_{d,b} \parallel L \parallel R_b)] \\
&= \sigma^{t_K}_{\mathsf{rel}}(\underline{\underline{c_4(d,b)}}) \cdot [\partial_H(S''_{d,b,t'_S-t_K} \parallel K \parallel L \parallel R'_{d,b})] \\
&\quad + \sum_{k\le t_K} \sigma^k_{\mathsf{rel}}(\underline{\underline{error}}) \cdot [\partial_H(S''_{d,b,t'_S-k} \parallel K \parallel L \parallel R_b)] \ , \\
&[\partial_H(S''_{d,b,t} \parallel K \parallel L \parallel R'_{d,b})] = \sigma^{t_R}_{\mathsf{rel}}(\underline{\underline{s_2(d)}}) \cdot [\partial_H(S''_{d,b,t-t_R} \parallel K \parallel L \parallel R''_{1-b})] \\
&\text{(for every } t > t_R),
\end{aligned}
$$

$$[\partial_H(S''_{d,b,t} \parallel K \parallel L \parallel R''_{1-b})]$$
$$= \sigma_{\mathsf{rel}}^{t'_R}(\underline{\underline{c_6(ack)}}) \cdot [\partial_H(S''_{d,b,t-t'_R} \parallel K \parallel L' \parallel R_{1-b})]$$
(for every $t > t'_R$),

$$[\partial_H(S''_{d,b,t} \parallel K \parallel L' \parallel R_{1-b})]$$
$$= \sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{c_5(ack)}}) \cdot [\partial_H(S_{1-b} \parallel K \parallel L \parallel R_{1-b})]$$
$$+ \sum_{k \leq t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot [\partial_H(S''_{d,b,t-k} \parallel K \parallel L \parallel R_{1-b})]$$
(for every $t > t_L$),

$$[\partial_H(S''_{d,b,t} \parallel K \parallel L \parallel R_b)] = \sigma_{\mathsf{rel}}^{t}(\underline{\underline{c_3(d,b)}}) \cdot [\partial_H(S''_{d,b} \parallel K'_{d,b} \parallel L \parallel R_b)]$$
(for every $t > 0$),

$$[\partial_H(S''_{d,b,t} \parallel K \parallel L \parallel R_{1-b})]$$
$$= \sigma_{\mathsf{rel}}^{t}(\underline{\underline{c_3(d,b)}}) \cdot [\partial_H(S''_{d,b} \parallel K'_{d,b} \parallel L \parallel R_{1-b})]$$
(for every $t > 0$),

$$[\partial_H(S''_{d,b} \parallel K'_{d,b} \parallel L \parallel R_{1-b})]$$
$$= \sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{c_4(d,b)}}) \cdot [\partial_H(S''_{d,b,t'_S-t_K} \parallel K \parallel L \parallel R''_{1-b})]$$
$$+ \sum_{k \leq t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot [\partial_H(S''_{d,b,t'_S-k} \parallel K \parallel L \parallel R_{1-b})] .$$

If the terms on the left-hand sides of these equations include all unexpanded terms on the right-hand sides, we have that the terms on the left-hand sides make up a solution of the guarded recursive specification obtained by replacing all occurrences of these terms in the equations by occurrences of corresponding variables. It is easy to see that this is the case iff $t'_S > t_K + t_R + t'_R + t_L$. Hence, we derive from RSP that $\partial_H(S_b \parallel K \parallel L \parallel R_b)$ is the solution for its corresponding variable of this guarded recursive specification. The guarded recursive specification concerned can easily be rewritten, using its equations and the axioms of $\mathrm{ACP}^{\mathrm{drt}}$, to the following sender-oriented guarded recursive specification:

$$X_b = \sum_{d \in D} \underline{\underline{r_1(d)}} \cdot \sigma_{\mathsf{rel}}^{t_S}(Y_{d,b}) + \sigma_{\mathsf{rel}}^{1}(X_b) ,$$
$$Y_{d,b} = \underline{\underline{c_3(d,b)}} \cdot \Bigg(\sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{c_4(d,b)}}) \cdot \sigma_{\mathsf{rel}}^{t_R}(\underline{\underline{s_2(d)}}) \cdot \sigma_{\mathsf{rel}}^{t'_R}(\underline{\underline{c_6(ack)}}) \cdot Z_{d,b}$$
$$+ \sum_{k \leq t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot \sigma_{\mathsf{rel}}^{t'_S-k}(Y_{d,b}) \Bigg) ,$$
$$Z_{d,b} = \sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{c_5(ack)}}) \cdot X_{1-b} + \sum_{k \leq t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot \sigma_{\mathsf{rel}}^{t'_S-(t_K+t_R+t'_R+k)}(U_{d,b}) ,$$

$$U_{d,b} = \underline{\underline{c_3(d,b)}} \cdot \Bigg(\sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{c_4(d,b)}}) \cdot \sigma_{\mathsf{rel}}^{t'_R}(\underline{\underline{c_6(ack)}}) \cdot V_{d,b} + \sum_{k \leq t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot \sigma_{\mathsf{rel}}^{t'_S - k}(U_{d,b}) \Bigg) ,$$

$$V_{d,b} = \sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{c_5(ack)}}) \cdot X_{1-b} + \sum_{k \leq t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{error}}) \cdot \sigma_{\mathsf{rel}}^{t'_S - (t_K + t'_R + k)}(U_{d,b}) .$$

From this recursive specification we can conclude informally that, if we abstract from all actions other than the send and receive actions at the external ports 1 and 2 and in addition from the timing of actions, the whole PAR protocol behaves as a buffer with capacity one.

5 Analysis of the PAR Protocol: Abstraction

We want to abstract from all actions other than the send and receive actions at the external ports 1 and 2, i.e. from the actions in the set

$$I = \{c_i(d,b) \mid i \in \{3,4\}, d \in D, b \in \{0,1\}\} \cup \{c_5(ack), c_6(ack), error\} .$$

We can proceed in different ways. First of all, we can focus on functional correctness. This means that we abstract from all timing of actions using the time free projection operator described in Section 2 before we abstract from the actions in the set I. In that case, we can apply the abstraction operator in the theory without timing. Starting from the specification of $\partial_H(S_b \parallel K \parallel L \parallel R_b)$ at the end of Section 4, we can easily calculate that $\pi_{\mathsf{tf}}(\partial_H(S_b \parallel K \parallel L \parallel R_b))$ is the solution of the guarded recursive specification that consists of the following equations:

$$X'_b = \sum_{d \in D} r_1(d) \cdot Y'_{d,b} ,$$

$$Y'_{d,b} = c_3(d,b) \cdot (c_4(d,b) \cdot s_2(d) \cdot c_6(ack) \cdot Z'_{d,b} + error \cdot Y'_{d,b}) ,$$

$$Z'_{d,b} = c_5(ack) \cdot X'_{1-b} + error \cdot U'_{d,b} ,$$

$$U'_{d,b} = c_3(d,b) \cdot (c_4(d,b) \cdot c_6(ack) \cdot V'_{d,b} + error \cdot U'_{d,b}) ,$$

$$V'_{d,b} = c_5(ack) \cdot X'_{1-b} + error \cdot U'_{d,b} .$$

Starting from this specification, we can calculate, using axiom B2 from the theory without timing, together with CFAR (Cluster Fair Abstraction Rule) to remove cycles of silent steps, that $\tau_I(\pi_{\mathsf{tf}}(\partial_H(S_b \parallel K \parallel L \parallel R_b)))$ is the solution of the guarded recursive specification that consists of the following equation:

$$B = \sum_{d \in D} r_1(d) \cdot s_2(d) \cdot B .$$

For more information on the silent step in the theory without timing, including details about CFAR, see [8]. We have obtained a guarded recursive specification

of a buffer with capacity one. Thus, we see that the PAR protocol is functionally correct. We want to stress that, in order to achieve this result, it was necessary to calculate first the time-dependent behavior of the whole protocol, because the PAR protocol is only correct if the timing parameters are set correctly. A complete verification in process algebra without timing is not possible without resorting to artificial tricks such as excluding the premature time-out of an acknowledgement by inhibiting a time-out so long as it does not lead to deadlock (see e.g. [2]).

Next, we can have a look at the performance properties. Starting from the specification of $\partial_H(S_b \,\|\, K \,\|\, L \,\|\, R_b)$ at the end of Section 4, we can easily calculate that $\tau_I(\partial_H(S_b \,\|\, K \,\|\, L \,\|\, R_b))$ is the solution of the guarded recursive specification that consists of the following equations:

$$
\begin{aligned}
X'' &= \sum_{d \in D} \underline{\underline{r_1(d)}} \cdot \sigma_{\mathsf{rel}}^{t_S}(Y''_d) + \sigma_{\mathsf{rel}}^{1}(X'') \;, \\
Y''_d &= \sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t_R}(\underline{\underline{s_2(d)}} \cdot \sigma_{\mathsf{rel}}^{t'_R}(Z''))) + \sum_{k \le t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t'_S - k}(Y''_d)) \;, \\
Z'' &= \sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{\tau}} \cdot X'') + \sum_{k \le t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t'_S - (t_K + t_R + t'_R + k)}(U'')) \;, \\
U'' &= \sigma_{\mathsf{rel}}^{t_K}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t'_R}(V'')) + \sum_{k \le t_K} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t'_S - k}(U'')) \;, \\
V'' &= \sigma_{\mathsf{rel}}^{t_L}(\underline{\underline{\tau}} \cdot X'') + \sum_{k \le t_L} \sigma_{\mathsf{rel}}^{k}(\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{t'_S - (t_K + t'_R + k)}(U'')) \;.
\end{aligned}
$$

Not many simplifications have been achieved. This is mainly because we cannot leave out silent steps that occur in between delays. In effect, all internal choices made, e.g. whether or not a channel forwards a datum correctly, remain visible. Some initial observations concerning this matter, as well as an analysis of a slightly different version of the PAR protocol, were made in [9]. In any case, from this specification, we can conclude informally that the protocol takes at least $t_S + t_K + t_R$ time slices between consumption and delivery of a datum, and in general, between consumption and delivery we have $t_S + t_K + t_R + i \cdot t'_S$ time slices, where $i \geq 0$. After delivery, at least $t'_R + t_L$ time slices must pass before the next datum can be consumed, and in general, we have $t'_R + t_L$ or $t'_R + t_L + j \cdot t'_S - t_R$ time slices, where $j > 0$.

6 Towards a Coarser Equivalence

The experience with analyzing the PAR protocol (and other protocols) triggered a quest for a new equivalence. We now informally introduce an equivalence that is coarser than rooted branching tail bisimulation equivalence, the version of branching bisimulation equivalence for processes with discrete relative timing introduced in [4] and used in [3] to construct a model for the axioms of $\mathrm{ACP}_\tau^{\mathrm{drt}}$ ($\mathrm{ACP}^{\mathrm{drt}}$ extended with abstraction, see Appendix A). The coarser equivalence treats an internal action always as redundant if it is followed by a process that is only capable of idling till the next time slice.

Our motivation for this equivalence is that it requires unlikely means to perceive differences between spending time in not performing any action and spending time in performing actions that are considered to be unobservable. Consider a process p of which one of the options is to idle till the next time slice and then to behave as process q. In addition, consider a process p' that behaves as p except that in the above-mentioned option idling till the next time slice is preceded by performing an internal action in the current time slice. In the case of p, the capabilities of the process change by idling till the next time slice. In the case of p', the capabilities of the process change by performing an internal action, but this change is only observable after the subsequent idling till the next time slice. In either case, the capabilities after the idling are the same. It is hard to imagine a realistic experiment that can distinguish between the processes p and p'.

In Section 7, we will define the coarser equivalence, called ab-bisimulation equivalence,[1] in terms of rooted branching tail bisimulation equivalence. For a better understanding, we first introduce the transition relations used in [3] for the operational semantics of $\mathrm{ACP}_\tau^{\mathrm{drt}}$: a binary *action step* relation $\_ \xrightarrow{a} \_$ and a unary *action termination* relation $\_ \xrightarrow{a} \surd$ for each $a \in \mathsf{A} \cup \{\tau\}$,[2] a binary *time step* relation $\_ \overset{m}{\longmapsto} \_$ for each $m \in \mathbb{N} - \{0\}$, and a unary *deadlocked* relation $\_\!\uparrow$. The transition relations can be explained as follows:

$t \xrightarrow{a} t'$: process t is capable of first performing action a in the current time slice and then proceeding as process t';
$t \xrightarrow{a} \surd$: process t is capable of first performing action a in the current time slice and then terminating successfully;
$t \overset{m}{\longmapsto} t'$: process t is capable of first idling till the mth-next time slice and then proceeding as process t';
$t\!\uparrow$: process t has deadlocked before the current time slice.

An action step $t_1 \xrightarrow{a} t_2$ is called a *silent step* if $a \equiv \tau$. The time step relations are defined such that $t \overset{m}{\longmapsto} t'$ and $t \overset{m}{\longmapsto} t''$ only if t' and t'' are the same. In other words, the time steps of a process are combined in the operational semantics of $\mathrm{ACP}_\tau^{\mathrm{drt}}$. This is no longer the case after the saturation of the transition relations as described below.

Informally, two closed terms t, t' are ab-bisimulation equivalent if t, t' are rooted branching tail bisimulation equivalent after saturation of the transition relations used for the operational semantics in the following two ways:

- whenever there is a silent step with thereafter only time steps, we add corresponding direct time steps; and conversely;
- whenever there are time steps of equal length to different processes, we add a time step of the same length to the alternative composition of those processes; and conversely.

[1] Here ab-bisimulation stands for adapted branching.
[2] The set A of actions is a parameter of $\mathrm{ACP}_\tau^{\mathrm{drt}}$, see further Appendix A.

The first kind of additions corresponds with the idea to treat internal actions always as redundant if they are followed by a process that is only capable of idling till the next time slice. The second kind of additions is needed because it is now impossible to combine in all cases in advance the time steps of a process.

Under ab-bisimulation equivalence, performing an internal action is in certain cases considered to be the same as not performing any action. That is why, in the case of some operators, the internal action must be distinguished from the other actions in the rules for the operational semantics. The necessary changes in the transition rules lead to corresponding changes in the axioms. In Section 7, we give the new rules needed for the operational semantics (Table 3) as well as the new axioms (Table 4). The distinction needs to be made between the internal action and the other actions in the case of the following operators: relative time-out, relative initialization, parallel composition, left merge, and communication merge. If we restrict ourselves to relative delay, alternative composition, sequential composition, encapsulation and abstraction, the additional identifications made by ab-bisimulation equivalence are completely characterized by the axiom $\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\underline{\delta}}) = \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\underline{\delta}})$ (axiom DRTB4). If we do not restrict ourselves to these operators, each of the axioms DRTO5, DRI5, CM3DRID and CM7DR (see Appendix A) has to be replaced by new axioms that are equivalent to the original axiom, except that they exclude the cases where the operator on the left-hand side has an argument of the form $\underline{\underline{\tau}} \cdot \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\underline{\delta}})$ – i.e. the cases where axiom DRTB4 can be used.

Notice that rooted branching tail bisimulation equivalence is the variant of rooted branching bisimulation equivalence for processes with discrete relative timing needed in the presence of the deadlocked process $\dot{\delta}$ (see e.g. [4]). In cases where rooted branching bisimulation equivalence suffices, the same saturation based approach can be used to define an equivalence that treats an internal action always as redundant if it is followed by a process that is only capable of idling till the next time slice.

7 The New Equivalence and Corresponding Axioms

In this section, we define ab-bisimulation equivalence, we give the new transition rules needed for the operational semantics of some operators, and we give the new axioms.

The definition of ab-bisimulation equivalence given below resembles the informal explanation given in Section 6. That is, it is defined as rooted branching tail bisimulation equivalence with respect to saturated transition relations. Our attempts to define ab-bisimulation equivalence more directly all resulted in definitions that were less elegant. The second way of saturation used in the definition, i.e. saturation according to the transition rules given in Table 2, originates from [10].

For the description of the saturation by means of additional transition rules, two auxiliary transition relations are introduced: a unary *idling* relation $\mathcal{I}(\_)$ and a binary *summand* relation $\_ \preceq \_$. They can be explained as follows:

$\mathcal{I}(t)$: process t is only capable of idling till the next time slice;
$t \preceq t'$: process t' is amongst other things able to behave as process t.

In consequence of the additional transition relations, the definitions of branching tail bisimulation equivalence and rooted branching tail bisimulation equivalence from [3, 4] have to be extended to take these transition relations into account. In the definitions, we denote the reflexive and transitive closure of $\_ \xrightarrow{\tau} \_$ by $\_ \twoheadrightarrow \_$. So $t \twoheadrightarrow t'$ indicates that t' is reachable from t by performing zero or more silent steps. Moreover, we write $t \xrightarrow{(a)} t'$ to indicate that either $t \xrightarrow{a} t'$ or $a \equiv \tau$ and $t = t'$.

A *branching tail bisimulation* is a symmetric binary relation B on closed terms such that for all closed terms t_1, t_1' with $B(t_1, t_1')$ the following conditions hold:

1. whenever $t_1 \xrightarrow{a} t_2$, then there are closed terms t_1^*, t_2' such that $t_1' \twoheadrightarrow t_1^* \xrightarrow{(a)} t_2'$ and $B(t_1, t_1^*)$ and $B(t_2, t_2')$;
2. whenever $t_1 \xrightarrow{a} \surd$, then there is a closed term t_1^* such that $t_1' \twoheadrightarrow t_1^* \xrightarrow{a} \surd$ and $B(t_1, t_1^*)$;
3. whenever $t_1 \overset{m}{\longmapsto} t_2$, then there are closed terms t_1^*, t_2' such that $t_1' \twoheadrightarrow t_1^* \overset{m}{\longmapsto} t_2'$ and $B(t_1, t_1^*)$ and $B(t_2, t_2')$;
4. whenever $t_1 \uparrow$, then $t_1' \uparrow$;
5. whenever $\mathcal{I}(t_1)$, then there is a closed term t_1^* such that $t_1' \twoheadrightarrow t_1^*$ and $\mathcal{I}(t_1^*)$ and $B(t_1, t_1^*)$;
6. whenever $t_2 \preceq t_1$, then there are closed terms t_1^*, t_2' such that $t_1' \twoheadrightarrow t_1^*$ and $t_2' \preceq t_1^*$ and $B(t_1, t_1^*)$ and $B(t_2, t_2')$.

Two closed terms t and t' are *branching tail bisimulation equivalent*, written $t \leftrightarroweq_{\mathrm{bt}} t'$, if there exists a branching tail bisimulation B such that $B(t, t')$.

If B is a branching tail bisimulation, then we say that the pair (t_1, t_1') satisfies the *root condition* in B if the following holds:

1. whenever $t_1 \xrightarrow{a} t_2$, then there is a closed term t_2' such that $t_1' \xrightarrow{a} t_2'$ and $B(t_2, t_2')$;
2. whenever $t_1 \xrightarrow{a} \surd$, then $t_1' \xrightarrow{a} \surd$;
3. whenever $t_1 \overset{m}{\longmapsto} t_2$, then there is a closed term t_2' such that $t_1' \overset{m}{\longmapsto} t_2'$ and $B(t_2, t_2')$;
4. whenever $\mathcal{I}(t_1)$, then $\mathcal{I}(t_1')$;
5. whenever $t_2 \preceq t_1$, then there is a closed term t_2' such that $t_2' \preceq t_1'$ and $B(t_2, t_2')$.

Two closed terms t and t' are *rooted branching tail bisimulation equivalent*, written $t \leftrightarroweq_{\mathrm{rbt}} t'$, if there exists a branching tail bisimulation B such that the pair (t, t'), and all pairs (t_1, t_1') with $t \overset{m}{\longmapsto} t_1$ for some $m > 0$ and $B(t_1, t_1')$, satisfy the root condition in B.

Two closed terms t and t' are *ab-bisimulation equivalent*, written $t \leftrightarroweq_{\mathrm{ab}} t'$, if $t \leftrightarroweq_{\mathrm{rbt}} t'$ after saturation with silent steps and time steps according to the

Table 1. Rules for first way of saturation ($m > 0$, $n \geq 0$)

$$\frac{x \xrightarrow{\tau} x',\ x' \overset{m}{\longmapsto} x'',\ \mathcal{I}(x')}{x \overset{m}{\longmapsto} x''} \qquad \frac{x \overset{m}{\longmapsto} x'}{x \xrightarrow{\tau} \overline{\upsilon}^1_{\mathsf{rel}}(x)}$$

$$\frac{x \not\downarrow}{\mathcal{I}(\sigma^{n+1}_{\mathsf{rel}}(x))} \qquad \frac{\mathcal{I}(x),\ \mathcal{I}(y)}{\mathcal{I}(x + y)} \qquad \frac{\mathcal{I}(x)}{\mathcal{I}(x \cdot y)} \qquad \frac{\mathcal{I}(x)}{\mathcal{I}(\upsilon^{n+2}_{\mathsf{rel}}(x))} \qquad \frac{x \not\downarrow}{\mathcal{I}(\overline{\upsilon}^{n+1}_{\mathsf{rel}}(x))}$$

$$\frac{\mathcal{I}(x),\ \mathcal{I}(y)}{\mathcal{I}(x \parallel y)} \qquad \frac{\mathcal{I}(x)}{\mathcal{I}(x \between y)} \qquad \frac{\mathcal{I}(x),\ \mathcal{I}(y)}{\mathcal{I}(x \mid y)} \qquad \frac{\mathcal{I}(x)}{\mathcal{I}(\partial_H(x))} \qquad \frac{\mathcal{I}(x)}{\mathcal{I}(\tau_I(x))}$$

$$\frac{\mathcal{I}(\langle t_X \,|\, E\rangle)}{\mathcal{I}(\langle X \,|\, E\rangle)} \text{ for each equation } X = t_X \in E \qquad \frac{\{\mathcal{I}(F(i)) \mid i \in K\}}{\mathcal{I}(\sum_{i \in K} F(i))} \qquad \frac{x \xrightarrow{\tau} x',\ \mathcal{I}(x')}{\mathcal{I}(x)}$$

Table 2. Rules for second way of saturation ($m > 0$, $n \geq 0$)

$$\frac{x \overset{m}{\longmapsto} x',\ x \overset{m}{\longmapsto} x''}{x \overset{m}{\longmapsto} x' + x''} \qquad \frac{x \overset{m}{\longmapsto} x',\ x'' \preceq x'}{x \overset{m}{\longmapsto} x''}$$

$$\frac{}{x \preceq x} \qquad \frac{}{\underline{\delta} \preceq x} \qquad \frac{x \not\downarrow}{\underline{\delta} \preceq x} \qquad \frac{x \preceq y}{\sigma^n_{\mathsf{rel}}(x) \preceq \sigma^n_{\mathsf{rel}}(y)} \qquad \frac{x' \preceq x}{x' \preceq x + y} \qquad \frac{y' \preceq y}{x' \preceq x + y}$$

$$\frac{x' \preceq x,\ y' \preceq y}{x' + y' \preceq x + y} \qquad \frac{x \preceq y}{x \cdot z \preceq y \cdot z} \qquad \frac{x \preceq y}{\upsilon^n_{\mathsf{rel}}(x) \preceq \upsilon^n_{\mathsf{rel}}(y)} \qquad \frac{x \preceq y}{\overline{\upsilon}^n_{\mathsf{rel}}(x) \preceq \overline{\upsilon}^n_{\mathsf{rel}}(y)}$$

$$\frac{x' \preceq x}{x' \between y \preceq x \parallel y} \qquad \frac{y' \preceq y}{y' \between x \preceq x \parallel y} \qquad \frac{x' \preceq x}{x' \mid y \preceq x \parallel y} \qquad \frac{y' \preceq y}{y' \mid x \preceq x \parallel y}$$

$$\frac{x \preceq y}{x \between z \preceq y \between z} \qquad \frac{x \preceq y}{x \mid z \preceq y \mid z} \qquad \frac{y \preceq z}{x \mid y \preceq x \mid z} \qquad \frac{x \preceq y}{\partial_H(x) \preceq \partial_H(y)} \qquad \frac{x \preceq y}{\tau_I(x) \preceq \tau_I(y)}$$

$$\frac{x \preceq \langle t_X \,|\, E\rangle}{x \preceq \langle X \,|\, E\rangle} \text{ for each equation } X = t_X \in E \qquad \frac{\{F(i) \preceq G(i) \mid i \in K\}}{\sum_{i \in K} F(i) \preceq \sum_{i \in L} G(i)}\ K \subseteq L$$

transition rules given in Table 1[3] and saturation with time steps according to the transition rules given in Table 2.[4]

As mentioned before in Section 6, under ab-bisimulation equivalence, the internal action must be distinguished from the other actions in the transition rules for relative time-out, relative initialization, parallel composition, left merge, and communication merge. This leads to the following changes in the transition rules describing the operational semantics of $\mathrm{ACP}^{\mathrm{drt}}_{\tau}$+Rec ($\mathrm{ACP}^{\mathrm{drt}}$ extended with abstraction and guarded recursion, see Appendix A):

- the four transition rules in the upper row of Table 3 are restricted to cover only $a \in \mathsf{A}$;
- the four transition rules in the lower row of Table 3 are added.

The coarser equivalence and these changes in the transition rules lead to the following changes in the axioms of $\mathrm{ACP}^{\mathrm{drt}}_{\tau}$+Rec:

- axiom DRTB4 given in Table 4 is added;

[3] Recall that $\overline{\upsilon}^1_{\mathsf{rel}}(x)$ differs from x only in not being capable of performing actions in the current time slice. Hence, we also have that $\overline{\upsilon}^1_{\mathsf{rel}}(x) \overset{m}{\longmapsto} x'$ if $x \overset{m}{\longmapsto} x'$.

[4] In Tables 1 and 2, $x \not\downarrow$ stands for "not $x \downarrow$".

Table 3. New transition rules ($a \in \mathsf{A}$, $n \geq 0$)

$$\frac{x \xrightarrow{a} x'}{\upsilon_{\mathsf{rel}}^{n+1}(x) \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x'}{\overline{\upsilon}_{\mathsf{rel}}^{0}(x) \xrightarrow{a} x'} \qquad \frac{x \xrightarrow{a} x',\ y\not\xrightarrow{}}{x \parallel y \xrightarrow{a} x' \parallel y} \qquad \frac{x \xrightarrow{a} x',\ y\not\xrightarrow{}}{x \between y \xrightarrow{a} x' \parallel y}$$

$$\frac{x \xrightarrow{\tau} x',\ \neg\mathcal{I}(x')}{\upsilon_{\mathsf{rel}}^{n+1}(x) \xrightarrow{\tau} x'} \qquad \frac{x \xrightarrow{\tau} x',\ \neg\mathcal{I}(x')}{\overline{\upsilon}_{\mathsf{rel}}^{0}(x) \xrightarrow{\tau} x'} \qquad \frac{x \xrightarrow{\tau} x',\ \neg\mathcal{I}(x'),\ y\not\xrightarrow{}}{x \parallel y \xrightarrow{\tau} x' \parallel y} \qquad \frac{x \xrightarrow{\tau} x',\ \neg\mathcal{I}(x'),\ y\not\xrightarrow{}}{x \between y \xrightarrow{\tau} x' \parallel y}$$

Table 4. New axioms ($a, b \in \mathsf{A} \cup \{\tau, \delta\}$, $m, n \geq 0$, $\mathsf{A}_\delta = \mathsf{A} \cup \{\delta\}$)

$\underline{\tau} \cdot \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\delta}) = \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\delta})$		DRTB4
$\upsilon_{\mathsf{rel}}^{n}(\underline{a} \cdot x) = \upsilon_{\mathsf{rel}}^{n}(\underline{a}) \cdot x$	if $a \in \mathsf{A}_\delta$	DRTO5a
$\upsilon_{\mathsf{rel}}^{n}(\underline{\tau} \cdot \upsilon_{\mathsf{rel}}^{1}(x)) = \upsilon_{\mathsf{rel}}^{n}(\underline{\tau}) \cdot \upsilon_{\mathsf{rel}}^{1}(x)$		DRTO5b
$\upsilon_{\mathsf{rel}}^{n}(\underline{\tau} \cdot (\underline{a} \cdot x + y)) = \upsilon_{\mathsf{rel}}^{n}(\underline{\tau}) \cdot (\underline{a} \cdot x + y)$	if $a \in \mathsf{A}$	DRTO5c
$\overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{a} \cdot x) = \overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{a}) \cdot x$	if $a \in \mathsf{A}_\delta$	DRI5a
$\overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{\tau} \cdot \upsilon_{\mathsf{rel}}^{1}(x)) = \overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{\tau}) \cdot \upsilon_{\mathsf{rel}}^{1}(x)$		DRI5b
$\overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{\tau} \cdot (\underline{a} \cdot x + y)) = \overline{\upsilon}_{\mathsf{rel}}^{n}(\underline{\tau}) \cdot (\underline{a} \cdot x + y)$	if $a \in \mathsf{A}$	DRI5c
$\underline{a} \cdot x \between (y + \underline{\delta}) = \underline{a} \cdot (x \parallel (y + \underline{\delta}))$	if $a \in \mathsf{A}_\delta$	CM3DRIDa
$\underline{\tau} \cdot \upsilon_{\mathsf{rel}}^{1}(x) \between (y + \underline{\delta}) = \underline{\tau} \cdot (\upsilon_{\mathsf{rel}}^{1}(x) \parallel (y + \underline{\delta}))$		CM3DRIDb
$\underline{\tau} \cdot (\underline{a} \cdot x + y) \between (z + \underline{\delta}) = \underline{\tau} \cdot ((\underline{a} \cdot x + y) \parallel (z + \underline{\delta}))$	if $a \in \mathsf{A}$	CM3DRIDc
$\underline{a} \cdot x \mid \underline{b} \cdot y = (\underline{a} \mid \underline{b}) \cdot (x \parallel y)$	if $a \in \mathsf{A}_\delta$ or $b \in \mathsf{A}_\delta$	CM7DRa
$\underline{\tau} \cdot \upsilon_{\mathsf{rel}}^{1}(x) \mid \underline{\tau} \cdot y = \underline{\delta}$		CM7DRb
$\underline{\tau} \cdot (\underline{a} \cdot x + y) \mid \underline{\tau} \cdot z = \underline{\delta}$	if $a \in \mathsf{A}$	CM7DRc
$\underline{\tau} \cdot x \mid \underline{\tau} \cdot \upsilon_{\mathsf{rel}}^{1}(y) = \underline{\delta}$		CM7DRd
$\underline{\tau} \cdot x \mid \underline{\tau} \cdot (\underline{a} \cdot y + z) = \underline{\delta}$	if $a \in \mathsf{A}$	CM7DRe

– axioms DRTO5, DRI5, CM3DRID and CM7DR are replaced by axioms DRTO5a–DRTO5c, DRI5a–DRI5c, CM3DRIDa–CM3DRIDc and CM7DRa–CM7DRe given in Table 4.

Axiom DRTB4 is the axiom that characterizes the additional identifications made by ab-bisimulation equivalence. The sets of axioms DRTO5a–DRTO5c, DRI5a–DRI5c, CM3DRIDa–CM3DRIDc and CM7DRa–CM7DRe are equivalent to the axioms DRTO5, DRI5, CM3DRID and CM7DR, respectively, except that they exclude the cases where the operator on the left-hand side has an argument of the form $\underline{\tau} \cdot \sigma_{\mathsf{rel}}^{n+1}(x + \underline{\delta})$.

Of the latter axioms, we have until now only used CM3DRID and CM7DR in the analysis of the PAR protocol. Moreover, we have not used them in cases excluded by the corresponding new axioms. Therefore, we seem to have used the new axioms from the start.

If we assume that with the adapted operational semantics ab-bisimulation equivalence is a congruence, it is straightforward to show that the adapted axiomatization is sound for ab-bisimulation equivalence. We claim that with the adapted operational semantics ab-bisimulation equivalence is a congruence. We also claim that the adapted axiomatization is complete for ab-bisimulation equiv-

alence. In ongoing work, the authors are constructing the full proofs of these results.

8 Analysis of the PAR Protocol Revisited

We take up the analysis of the PAR protocol again, but now using the additional axiom DRTB4.

In this case, we abstract from the actions in I without preceding abstraction from the timing of actions. Starting from the specification of $\partial_H(S_b \parallel K \parallel L \parallel R_b)$ at the end of Section 4, we can now calculate that $\tau_I(\partial_H(S_b \parallel K \parallel L \parallel R_b))$ is the solution of the guarded recursive specification that consists of the following equations:

$$\begin{aligned}
X^* &= \sum_{d \in D} \underline{\underline{r_1(d)}} \cdot \sigma_{\mathsf{rel}}^{t_S}(Y_d^*) + \sigma_{\mathsf{rel}}^1(X^*) \;, \\
Y_d^* &= \sigma_{\mathsf{rel}}^{t_K + t_R}(\underline{\underline{s_2(d)}}) \cdot (\sigma_{\mathsf{rel}}^{t'_R + t_L}(\underline{\underline{\tau}} \cdot X^*) + \sigma_{\mathsf{rel}}^{t'_S - (t_K + t_R)}(Z^*)) + \sigma_{\mathsf{rel}}^{t'_S}(Y_d^*) \;, \\
Z^* &= \sigma_{\mathsf{rel}}^{t_K + t'_R + t_L}(\underline{\underline{\tau}} \cdot X^*) + \sigma_{\mathsf{rel}}^{t'_S}(Z^*) \;.
\end{aligned}$$

Many simplifications have been achieved by using axiom DRTB4. In particular, all internal actions that hinder performance analysis could be removed. Moreover, it is now possible to show that the PAR protocol is functionally correct by abstracting from the timing of actions next. That is, we can calculate that $\pi_{\mathsf{tf}}(\tau_I(\partial_H(S_b \parallel K \parallel L \parallel R_b)))$ is the solution of the guarded recursive specification of a buffer with capacity one. Surprisingly, CFAR is not needed in this case.

A more intelligible specification of $\tau_I(\partial_H(S_b \parallel K \parallel L \parallel R_b))$ can be obtained by using the summation operator described in Section 2. Let E be a guarded recursive specification that includes the equation $X = t + \sigma_{\mathsf{rel}}^n(X)$. Then the equation $\langle X|E\rangle = \sum_{i \geq 0} \sigma_{\mathsf{rel}}^{i \cdot n}(\langle t|E\rangle)$ is derivable. Using this result, we can easily rewrite the specification of $\tau_I(\partial_H(S_b \parallel K \parallel L \parallel R_b))$ given above to the following one:

$$\begin{aligned}
X^* = \sum_{d \in D} r_1(d) \cdot \sum_{i \geq 0} &\sigma_{\mathsf{rel}}^{t_S + t_K + t_R + i \cdot t'_S}(\underline{\underline{s_2(d)}}) \\
&\cdot (\sigma_{\mathsf{rel}}^{t'_R + t_L}(\underline{\underline{\tau}} \cdot X^*) + \sum_{j > 0} \sigma_{\mathsf{rel}}^{t'_R + t_L + j \cdot t'_S - t_R}(\underline{\underline{\tau}} \cdot X^*)) \;,
\end{aligned}$$

where we write $r_1(d)$ for the term $\langle Y|Y = \underline{\underline{r_1(d)}} + \sigma_{\mathsf{rel}}^1(Y)\rangle$. This specification clearly exhibits both the functional behaviour and the performance properties of the PAR protocol. It is manifest that the protocol behaves as a buffer with capacity one, that a datum is delivered $t_S + t_K + t_R + i \cdot t'_S$ time slices after its consumption (for some $i \geq 0$), and that the next datum can be consumed $t'_R + t_L$ or $t'_R + t_L + j \cdot t'_S - t_R$ time slices after the delivery (for some $j > 0$).

9 Concluding Remarks

We have described and analyzed the PAR protocol using the version of process algebra with discrete relative timing and abstraction from [3]. We were able to describe the protocol adequately. In addition, we were able to analyze its functional correctness thoroughly. That is, we could prove that the protocol behaves correctly if the time-out time of the sender is longer than the time that a complete protocol cycle takes. On the other hand, we were not able to analyze its performance properties as thoroughly. Therefore, we have proposed a variant of the version of branching bisimulation equivalence for processes with discrete relative timing used in [3], called ab-bisimulation equivalence, which still coincides with the original version of branching bisimulation equivalence from [5] in the case without timing. We believe that we have a plausible motivation for this coarser equivalence. In any case, we have shown that the axioms based on this equivalence permit thorough performance analysis in the case of the PAR protocol.

In [11], the PAR protocol is described and analyzed using a version of process algebra with continuous absolute timing. The description originates from an earlier description in [12]. In the case of the analysis in [11], handwavings are needed to come as far as in the case of our analysis. Other protocols that are described and analyzed in versions of process algebra with timing include the ABP (Alternating Bit Protocol) and the CABP (Concurrent Alternating Bit Protocol), both in [13], and Fischer's mutual exclusion protocol, in [14]. In virtually all cases, there is a need for a coarser equivalence. The equivalences suggested in [13, 14] are in the case without timing also coarser than the original version of branching bisimulation equivalence from [5]. We claim that, in the case of the above-mentioned protocols, there is no need for an equivalence that is coarser than ab-bisimulation equivalence.

References

1. A.S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood Cliffs, 1981.
2. F.W. Vaandrager. Two simple protocols. In J.C.M. Baeten, editor, *Applications of Process Algebra*, pages 23–44. Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press, Cambridge, 1990.
3. J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*. EATCS Monographs Series, Springer Verlag, Berlin, 2002.
4. J.C.M. Baeten, J.A. Bergstra, and M.A. Reniers. Discrete time process algebra with silent step. In G.D. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 535–569. MIT Press, Cambridge, MA, 2000.
5. R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.
6. J.C.M. Baeten and C.A. Middelburg. Process algebra with timing: Real time and discrete time. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 627–684. Elsevier, Amsterdam, 2001.

7. J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1–3):109–137, 1984.
8. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, Cambridge, 1990.
9. D. Bošnački. Specification and verification of the PAR protocol in discrete time process algebra with relative timing. Unpublished paper, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, 1997.
10. J.C.M. Baeten and M.A. Reniers. Termination in timed process algebra. Computing Science Report 00-13, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, June 2000.
11. A.S. Klusener. *Models and Axioms for a Fragment of Real Time Process Algebra*. PhD thesis, Department of Computing Science, Eindhoven University of Technology, Eindhoven, 1993.
12. J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.
13. J.A. Hillebrand. The ABP and CABP – a comparison of performances in real time process algebra. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes 1994*, pages 124–147. Workshop in Computing Series, Springer-Verlag, Berlin, 1995.
14. J.J. Vereijken. Fischer's protocol in timed process algebra. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes 1995*, pages 245–284. Report 95-14, Department of Computing Science, Eindhoven University of Technology, Eindhoven, 1995.
15. C.A. Middelburg. Variable binding operators in transition system specifications. *Journal of Logic and Algebraic Programming*, 47(1):15–45, 2001.

A Process Algebra with Discrete Relative Timing

For reference, the axioms of $\mathrm{ACP}^{\mathrm{drt}}$, and its extension with abstraction and guarded recursion, are given in this appendix.

$\mathrm{ACP}^{\mathrm{drt}}$ assumes that a fixed but arbitrary finite set A of *actions* and a fixed but arbitrary *communication function*, i.e. a partial commutative and associative function $\gamma : \mathsf{A} \times \mathsf{A} \to \mathsf{A}$, has been given. The function γ is regarded to give the result of synchronously performing any two actions for which this is possible, and to be undefined otherwise. The axiom system of $\mathrm{ACP}^{\mathrm{drt}}$ consists of the equations given in Tables 5 and 6.

In Table 7, some equations concerning parallel composition are given that are derivable for closed terms from the axioms of $\mathrm{ACP}^{\mathrm{drt}}$ and hold in the model of $\mathrm{ACP}^{\mathrm{drt}}$ presented in [3, 6]. These equations are called the *axioms of standard concurrency*. In most applications, $\gamma(\gamma(a,b),c)$ is undefined for all $a, b, c \in \mathsf{A}$. That case is called handshaking communication. Under the assumption of handshaking communication, the equation given in Table 8, called the *handshaking axiom*, is derivable for closed terms.

The extension of $\mathrm{ACP}^{\mathrm{drt}}$ with abstraction is called $\mathrm{ACP}^{\mathrm{drt}}_{\tau}$. This extension assumes that $\tau \notin \mathsf{A}$ and that $\gamma(a,\tau)$ is undefined for all $a \in \mathsf{A} \cup \{\tau\}$. The axiom system of $\mathrm{ACP}^{\mathrm{drt}}_{\tau}$ consists of the equations in Table 5, with the understanding

that the axiom schemas in addition cover the case that $a \equiv \tau$, the equations in Table 6, with the understanding that the axiom schemas in addition cover the cases that $a \equiv \tau$ and $b \equiv \tau$, and the equations given in Table 9.

The extension of $\mathrm{ACP}_\tau^{\mathrm{drt}}$ with guarded recursion is called $\mathrm{ACP}_\tau^{\mathrm{drt}}$+Rec. The axiom system of $\mathrm{ACP}_\tau^{\mathrm{drt}}$+Rec consists of the axioms of $\mathrm{ACP}_\tau^{\mathrm{drt}}$ and the equations given in Table 10.

The defining axioms of *time free projection* are given in Table 11. Useful axioms of the *summation* operator are given in Table 12. We use F and G as variables ranging over functions that map each $n \in \mathbb{N}$ to a process and can be represented by terms containing a designated free variable ranging over $\mathbb{N}$. For more information on such second-order variables, see e.g. [3, 15].

Table 5. Axioms of $\mathrm{BPA}^{\mathrm{drt}}$ ($a \in \mathsf{A} \cup \{\delta\}$, $m, n \geq 0$)

$x + y = y + x$	A1	$\sigma^0_{\mathsf{rel}}(x) = x$	DRT1
$(x + y) + z = x + (y + z)$	A2	$\sigma^m_{\mathsf{rel}}(\sigma^n_{\mathsf{rel}}(x)) = \sigma^{m+n}_{\mathsf{rel}}(x)$	DRT2
$x + x = x$	A3	$\sigma^n_{\mathsf{rel}}(x) + \sigma^n_{\mathsf{rel}}(y) = \sigma^n_{\mathsf{rel}}(x + y)$	DRT3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$\sigma^n_{\mathsf{rel}}(x) \cdot y = \sigma^n_{\mathsf{rel}}(x \cdot y)$	DRT4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$\sigma^1_{\mathsf{rel}}(\dot\delta) = \underline{\delta}$	DRT7
$x + \dot\delta = x$	A6ID	$\underline{a} + \underline{\delta} = \underline{a}$	A6DRa
$\dot\delta \cdot x = \dot\delta$	A7ID		
$\upsilon^n_{\mathsf{rel}}(\dot\delta) = \dot\delta$	DRTO0	$\overline{\upsilon}^n_{\mathsf{rel}}(\dot\delta) = \sigma^n_{\mathsf{rel}}(\dot\delta)$	DRI0
$\upsilon^0_{\mathsf{rel}}(x) = \dot\delta$	DRTO1	$\overline{\upsilon}^0_{\mathsf{rel}}(x) = x$	DRI1
$\upsilon^{n+1}_{\mathsf{rel}}(\underline{a}) = \underline{a}$	DRTO2	$\overline{\upsilon}^{n+1}_{\mathsf{rel}}(\underline{a}) = \sigma^n_{\mathsf{rel}}(\underline{\delta})$	DRI2
$\upsilon^{m+n}_{\mathsf{rel}}(\sigma^n_{\mathsf{rel}}(x)) = \sigma^n_{\mathsf{rel}}(\upsilon^m_{\mathsf{rel}}(x))$	DRTO3	$\overline{\upsilon}^{m+n}_{\mathsf{rel}}(\sigma^n_{\mathsf{rel}}(x)) = \sigma^n_{\mathsf{rel}}(\overline{\upsilon}^m_{\mathsf{rel}}(x))$	DRI3
$\upsilon^n_{\mathsf{rel}}(x + y) = \upsilon^n_{\mathsf{rel}}(x) + \upsilon^n_{\mathsf{rel}}(y)$	DRTO4	$\overline{\upsilon}^n_{\mathsf{rel}}(x + y) = \overline{\upsilon}^n_{\mathsf{rel}}(x) + \overline{\upsilon}^n_{\mathsf{rel}}(y)$	DRI4
$\upsilon^n_{\mathsf{rel}}(x \cdot y) = \upsilon^n_{\mathsf{rel}}(x) \cdot y$	DRTO5	$\overline{\upsilon}^n_{\mathsf{rel}}(x \cdot y) = \overline{\upsilon}^n_{\mathsf{rel}}(x) \cdot y$	DRI5

Table 6. Additional axioms for $\mathrm{ACP}^{\mathrm{drt}}$ ($a, b \in \mathsf{A} \cup \{\delta\}$, $c \in \mathsf{A}$, $n \geq 0$)

$x \parallel y = (x \between y + y \between x) + x \mid y$	CM1	$\dot\delta \mid x = \dot\delta$	CMID3
$\dot\delta \between x = \dot\delta$	CMID1	$x \mid \dot\delta = \dot\delta$	CMID4
$x \between \dot\delta = \dot\delta$	CMID2	$\underline{a} \cdot x \mid \underline{b} = (\underline{a} \mid \underline{b}) \cdot x$	CM5DR
$\underline{a} \between (x + \underline{\delta}) = \underline{a} \cdot (x + \underline{\delta})$	CM2DRID	$\underline{a} \mid \underline{b} \cdot x = (\underline{a} \mid \underline{b}) \cdot x$	CM6DR
$\underline{a} \cdot x \between (y + \underline{\delta}) = \underline{a} \cdot (x \between (y + \underline{\delta}))$	CM3DRID	$\underline{a} \cdot x \mid \underline{b} \cdot y = (\underline{a} \mid \underline{b}) \cdot (x \parallel y)$	CM7DR
$\sigma^n_{\mathsf{rel}}(x) \between (\upsilon^n_{\mathsf{rel}}(y) + \sigma^n_{\mathsf{rel}}(z)) = \sigma^n_{\mathsf{rel}}(x \between z)$	DRCM2	$(\upsilon^1_{\mathsf{rel}}(x) + \underline{\delta}) \mid \sigma^{n+1}_{\mathsf{rel}}(y) = \underline{\delta}$	DRCM3ID
$(x + y) \between z = x \between z + y \between z$	CM4	$\sigma^{n+1}_{\mathsf{rel}}(x) \mid (\upsilon^1_{\mathsf{rel}}(y) + \underline{\delta}) = \underline{\delta}$	DRCM4ID
$\partial_H(\dot\delta) = \dot\delta$	D0	$\sigma^n_{\mathsf{rel}}(x) \mid \sigma^n_{\mathsf{rel}}(y) = \sigma^n_{\mathsf{rel}}(x \mid y)$	DRCM5
$\partial_H(\underline{a}) = \underline{a}$ if $a \notin H$	D1DR	$(x + y) \mid z = x \mid z + y \mid z$	CM8
$\partial_H(\underline{a}) = \underline{\delta}$ if $a \in H$	D2DR	$x \mid (y + z) = x \mid y + x \mid z$	CM9
$\partial_H(\sigma^n_{\mathsf{rel}}(x)) = \sigma^n_{\mathsf{rel}}(\partial_H(x))$	DRD	$\underline{a} \mid \underline{b} = \underline{c}$ if $\gamma(a, b) = c$	CF1DR
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	D3	$\underline{a} \mid \underline{b} = \underline{\delta}$ if $\gamma(a, b)$ undefined	CF2DR
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	D4		

Table 7. Axioms of standard concurrency

$x \parallel y = y \parallel x$	$x \mid y = y \mid x$
$(x \parallel y) \parallel z = x \parallel (y \parallel z)$	$(x \mid y) \mid z = x \mid (y \mid z)$
$(x \between y) \between z = x \between (y \parallel z)$	$x \mid (y \between z) = (x \mid y) \between z$

Table 8. Handshaking axiom

$\upsilon^1_{\mathsf{rel}}(x \mid y \mid z) + \underline{\underline{\delta}} = \underline{\underline{\delta}}$

Table 9. Additional axioms for $\mathrm{ACP}_\tau^{\mathrm{drt}}$ ($a \in \mathsf{A} \cup \{\delta, \tau\}$, $n \geq 0$)

$x \cdot (\underline{\underline{\tau}} \cdot (\upsilon^1_{\mathsf{rel}}(y) + z + \underline{\underline{\delta}}) + \upsilon^1_{\mathsf{rel}}(y)) = x \cdot (\upsilon^1_{\mathsf{rel}}(y) + z + \underline{\underline{\delta}})$	DRTB1
$x \cdot (\underline{\underline{\tau}} \cdot (\upsilon^1_{\mathsf{rel}}(y) + z + \underline{\underline{\delta}}) + z) = x \cdot (\upsilon^1_{\mathsf{rel}}(y) + z + \underline{\underline{\delta}})$	DRTB2
$x \cdot (\sigma^1_{\mathsf{rel}}(\underline{\underline{\tau}} \cdot (y + \underline{\underline{\delta}})) + \upsilon^1_{\mathsf{rel}}(z)) = x \cdot (\sigma^1_{\mathsf{rel}}(y + \underline{\underline{\delta}}) + \upsilon^1_{\mathsf{rel}}(z))$	DRTB3
$\tau_I(\underline{\underline{\delta}}) = \underline{\underline{\delta}}$	TI0
$\tau_I(\underline{\underline{a}}) = \underline{\underline{a}}$ if $a \notin I$	TI1DR
$\tau_I(\underline{\underline{a}}) = \underline{\underline{\tau}}$ if $a \in I$	TI2DR
$\tau_I(\sigma^n_{\mathsf{rel}}(x)) = \sigma^n_{\mathsf{rel}}(\tau_I(x))$	DRTI
$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$	TI3
$\tau_I(x \cdot y) = \tau_I(x) \cdot \tau_I(y)$	TI4

Table 10. Additional axioms for $\mathrm{ACP}_\tau^{\mathrm{drt}}$+Rec

$\langle X \mid E \rangle = \langle t_X \mid E \rangle$	for each $X = t_X \in E$	RDP
$E \Rightarrow X = \langle X \mid E \rangle$	for each $X \in \mathrm{V}(E)$	RSP

Table 11. Axioms for time free projection ($a \in \mathsf{A} \cup \{\delta, \tau\}$, $n \geq 0$)

$\pi_{\mathsf{tf}}(\underline{\underline{\delta}}) = \langle X \mid X = \underline{\underline{\delta}} + \sigma^1_{\mathsf{rel}}(X) \rangle$	DRTFP0
$\pi_{\mathsf{tf}}(\underline{\underline{a}}) = \langle X \mid X = \underline{\underline{a}} + \sigma^1_{\mathsf{rel}}(X) \rangle$	DRTFP1
$\pi_{\mathsf{tf}}(\sigma^n_{\mathsf{rel}}(x)) = \pi_{\mathsf{tf}}(x)$	DRTFP2
$\pi_{\mathsf{tf}}(x + y) = \pi_{\mathsf{tf}}(x) + \pi_{\mathsf{tf}}(y)$	DRTFP3
$\pi_{\mathsf{tf}}(x \cdot y) = \pi_{\mathsf{tf}}(x) \cdot \pi_{\mathsf{tf}}(y)$	DRTFP4

Table 12. Axioms for summation ($n \geq 0$)

$\sum_{i \in K} F(i) = \sum_{j \in K} F(j)$	SUM1	$(\forall i \in K \bullet F(i) = G(i)) \Rightarrow$	
$\sum_{i \in \emptyset} F(i) = \mathring{\delta}$	SUM2	$\sum_{i \in K} F(i) = \sum_{i \in K} G(i)$	SUM6
$\sum_{i \in \{n\}} F(i) = F(n)$	SUM3	$\sum_{i \in K} \sigma^n_{\mathsf{rel}}(F(i)) = \sigma^n_{\mathsf{rel}}(\sum_{i \in K} F(i))$	SUM7
$\sum_{i \in K \cup L} F(i) =$		$\sum_{i \in K} (F(i) + G(i)) =$	
$\sum_{i \in K} F(i) + \sum_{i \in L} F(i)$	SUM4	$\sum_{i \in K} F(i) + \sum_{i \in K} G(i)$	SUM8
$K \neq \emptyset \Rightarrow \sum_{i \in K} x = x$	SUM5	$\sum_{i \in K} (F(i) \cdot x) = (\sum_{i \in K} F(i)) \cdot x$	SUM9