Termination in timed process algebra

Citation for published version (APA):

Document status and date:
Published: 01/01/2000

Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 19. Oct. 2020
Termination in Timed Process Algebra

by

J.C.M. Baeten and M.Á. Reniers

00/13

ISSN 0926-4515

All rights reserved
editors: prof.dr. J.C.M. Baeten
prof.dr. P.A.J. Hilbers

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Reports 00/13
Eindhoven, June 2000
Termination in Timed Process Algebra

J.C.M. Baeten and M.A. Reniers
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
{J.C.M.Baeten,M.A.Reniers}@tue.nl

Abstract

We investigate different forms of termination in timed process algebras. The integrated framework of discrete and dense time, relative and absolute time process algebras is extended with forms of successful and unsuccessful termination. The different algebras are interrelated by embeddings and conservative extensions.

1 Introduction

Process algebras that incorporate some form of timing, enabling quantitative analysis of time performance, have been studied extensively by now (see e.g. [BB91, HR95, MT90, NS94, QdFA93]), and successfully applied in a number of case studies (see e.g. [BR97, SDJ+91]).

For different purposes and applications different versions of describing timing have been presented in the literature. The following fundamental choices have to be made:

1. The nature of the time domain.
   (a) Discrete time versus dense time. This choice is with respect to which type of time domain, timing is described. Usually, two types of time domains are distinguished. These are discrete time, where the time domain is of a discrete nature, and dense time, where the time domain is of a continuous nature, i.e. between every two moments in time there is another moment in time.
   (b) Linear time versus branching time. In a linear time domain each two moments in time are ordered by some total ordering ≤, in a branching time domain this is only a partial ordering.
   (c) Least element available or not.

For example in timed μCRL [Gro97], the user of the language can specify his own time domain by means of algebraic specifications as long as it has a least element and a total order. In this process algebra both discrete and dense time domains can be used.
2. The way time is described syntactically.

(a) Time-stamped description versus two-phase description. If the description of time is attached to the atomic actions, we speak of a time-stamping mechanism. If, on the other hand, time delay is decoupled from action execution, we have a two-phase approach.

3. The way time is incorporated semantically.

(a) Absolute time versus relative timing. Sometimes, it is convenient to describe the passage of time with respect to a global clock, sometimes, it is convenient to describe passage of time relative to the previous action. The first is called absolute timing, the second relative timing. The combination of these two notions is called parametric timing [BB96, BB97]

(b) Time-determinism versus time-nondeterminism. A choice to be made with far reaching consequences is whether the passage of time may determine a choice. In the literature three versions are encountered. Firstly, if the passage of time by itself may not determine any choice we speak of strong time-determinism. Secondly, if the passage of $t$ time by itself cannot determine a choice between alternatives that allow passage of $t$ time we speak of weak time-determinism. Finally, if the passage of time can determine a choice we speak of time-nondeterminism. Strong time-determinism is found in ATP [NS94], weak time-determinism in many ACP-like timed process algebras, and time-nondeterminism in $ACP_{\tau}$ [Gro91]. In the literature, time-determinism is also referred to as time-factorisation.

(c) Duration of actions. One can either assume that actions have no duration, and hence are instantaneous, or that they do have a duration. In the latter case, the duration can be specified or unspecified.

(d) Urgent actions versus multi-actions. If actions with the same time can be ordered these are called urgent actions, otherwise these are called multi-actions.

In [BM99], several process algebras are treated in a common framework, and related by embeddings and conservative extension relations. These process algebras, $ACP_{\text{sat}}$, $ACP_{\text{art}}$, $ACP_{\text{dat}}$ and $ACP_{\text{drt}}$, allow the execution of two or more actions consecutively at the same point in time, separate the execution of actions from the passage of time, adhere to the principle of time-determinism, and consider actions to have no duration. The process algebra $ACP_{\text{sat}}$ is a real-time process algebra with absolute time, $ACP_{\text{art}}$ is a real-time process algebra with relative time. Similarly, $ACP_{\text{dat}}$ and $ACP_{\text{drt}}$ are discrete-time process algebras with absolute time and relative time respectively. In these process algebras, considerable attention was given to the inaction constant $\delta$, standing for unsuccessful termination or deadlock and the different roles it plays in the untimed theory.

In this paper, we extend the framework of [BM99] with a constant representing successful termination for the relative-time, dense-time case. This means that we develop a linear-time, dense-time process algebra with urgent actions which have no duration. Furthermore, we will adopt the principle of time-determinism. In the introduction of this constant guidance comes from discussing the different roles of such a constant in untimed process algebra. First, however, we discuss the roles of unsuccessful termination in the untimed theory BPA$_{\delta}$ and the
relative-time, dense-time process algebra BPA\textsuperscript{sr}. The reason for this detour is that similar motivations and reasonings apply to the introduction of successful termination.

In the untimed Basic Process Algebra BPA\textsubscript{\textbf{d}}, with encapsulation, deadlock is represented by the constant $\delta$. This constant has two roles. On the one hand, deadlock is the neutral element for alternative composition (very useful in algebraic calculations and extensions with generalised sum over a data type), on the other hand it represents a blocked (or encapsulated) atomic action.

The two roles of $\delta$ in an untimed setting diverge when timing is added. The preferred interpretation of an untimed atomic action $a$ in a timed setting is an action that happens at some unspecified moment in time. This interpretation captures the intuition that describing a process without explicitly mentioning timing should be read as not restricting the process described in time. As a consequence the blocked action $\delta$ can let time pass, but can execute no action. In a timed setting, however, some processes cannot let time pass indefinitely, as they must execute some action by some specified time. Thus, $\delta$ cannot be the neutral element of alternative composition as this would give that the passage of time can determine a choice.

In [BM99], these different roles of deadlock are separated as follows: the encapsulation of an atomic action $a$ is (still) represented by the constant $\delta$, and the neutral element of alternative composition is represented by the new constant $\tilde{\delta}$. This constant $\tilde{\delta}$ is usually called \textit{immediate deadlock}, but in this paper we call it the \textit{deadlocked process}. Besides these two deadlock constants, Baeten and Middelburg also introduce the \textit{undelayable deadlock} constant $\tilde{s}$. The undelayable deadlock constant $\tilde{s}$ does not allow the passage of time, just as the deadlocked process $\tilde{\delta}$, but it does allow the execution of urgent actions in a parallel context, in contrast with $\delta$.

The empty process $\epsilon$ denoting successful termination or skip has not been studied nearly as well as the unsuccessful termination constant. The untimed theory was investigated in [KV85, BG87, Vra97]. In the context of ACP-like process algebras the empty process in a timed setting is mentioned in [Gro91, Ver97, BV97]. In [Ver97, BV97] a relative-time, discrete-time process algebra has been extended with both a non-delayable and a delayable successful termination constant.

As is the case for deadlock in the untimed theory, also the empty process has more roles. On the one hand, it serves as the neutral element for sequential composition (and parallel composition), on the other hand, it stands for the process that executes no actions but terminates at some unspecified time. Assuming that we want the embedding of untimed process algebra into timed process algebra where atomic actions and deadlock are delayable timed constants, it is impossible to use only one timed successful termination constant for both roles. This is explained as follows. Suppose that we want to treat the untimed successful termination constant as being nondelayable in the timed setting. Then, the timed interpretation of the untimed identity $\epsilon + \delta = \epsilon$ is not valid anymore as the left-hand side of the identity is delayable and the right-hand side is not! Thus, the interpretation of $\epsilon$ must be a delayable constant. Such a delayable constant cannot act as a neutral element for sequential composition: $\epsilon \cdot \tilde{s} \neq \tilde{s}$. Hence, with timing, if $\epsilon$ represents the successful termination constant that allows passage of time, we introduce a new constant $\tilde{\epsilon}$, called the \textit{terminated process}, that is the neutral element for sequential and parallel composition.
The process \( \hat{e} \) denotes a terminated process: termination has taken place, so no parallel activity can precede the termination. With this constant, we finally have a complete interpretation of the constant process \( a \) in a timed setting: upon executing the action, what remains is \( \hat{e} \).

Instead of extending the relative-time, dense-time process algebra \( \text{BPA}^{\text{strt}} \), we could equally well have chosen to extend one of the other process algebras from the framework presented by Baeten and Middelburg. The reason that we choose for the extension of \( \text{BPA}^{\text{strt}} \) is that the definition of the operational semantics and the equivalence relation (strong bisimilarity) are complex in this case. This is due to the intricate interplay between termination, time-determinism, and sequential composition. Another reason for choosing for the relative-time case is that the discrete-time version has been investigated already in [Ver97].

Thus, we have extended the integrated framework of process algebras with timing from [BM99] with a relative-time, dense-time process algebra with the empty process. We will show that the various subtheories are still related by means of embeddings and conservative extensions. This extension with the empty process, in itself needed for a clear understanding of termination in timed process algebra, is also needed in order to give semantics for programming languages and specification languages that involve skip: examples are CSP (see [Hoa85, SDJ+91]), \( \chi \) (see [BK00]) and MSC (see [Ren99]).

This article is structured as follows. In Section 2, we first present the process algebra \( \text{BPA}^{\text{strt}} \) from [BM99]. Then, we present the process algebra \( \text{BPA}^{\text{ert}} \) which is basically \( \text{BPA}^{\text{strt}} \) extended with a constant for the terminated process. We present an operational semantics for this process algebra and prove completeness of the axiomatisation with respect to timed strong bisimilarity on this operational semantics. In Section 3, we extend the process algebra \( \text{BPA}^{\text{ert}} \) with operators for parallelism and communication. In Section 4, we consider the relation of the process algebra \( \text{BPA}^{\text{ert}} \) with other process algebras by means of embeddings and conservative extensions.

Acknowledgments We like to thank Sjouke Mauw, Kees Middelburg and Tim Willemse for their various useful suggestions.

2 The basic process algebra

2.1 The process algebra \( \text{BPA}^{\text{strt}} \)

In this section we give a short overview of the process algebra \( \text{BPA}^{\text{strt}} \) as it the process algebra without successful termination constants that is closest to the process algebra that we introduce.

The process algebra \( \text{BPA}^{\text{strt}} \) is parametrised by a set \( A \) of actions. For each action \( a \in A \), the signature of \( \text{BPA}^{\text{strt}} \) contains an atomic action (constant) \( \hat{a} \) which represents the immediate and urgent execution of action \( a \). Furthermore, the signature of \( \text{BPA}^{\text{strt}} \) contains the constants \( \hat{d} \) for the deadlocked process and \( \hat{\delta} \) for undelayable deadlock. More complex processes are constructed from these constants using the alternative composition operator \( + \), the sequential
Table 1: Axioms of $\text{BPA}^{\text{st}}$ ($a \in A, p, q \geq 0, r > 0$)

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>$x + y = y + x$</td>
</tr>
<tr>
<td>A2</td>
<td>$(x + y) + z = x + (y + z)$</td>
</tr>
<tr>
<td>A3</td>
<td>$x + x = x$</td>
</tr>
<tr>
<td>A4</td>
<td>$(x + y) \cdot z = x \cdot z + y \cdot z$</td>
</tr>
<tr>
<td>A5</td>
<td>$\frac{a}{\delta} + \frac{b}{\eta} = \frac{a}{\epsilon}$</td>
</tr>
<tr>
<td>A6</td>
<td>$\sigma^{\text{rel}}_{a}(x) = x$</td>
</tr>
<tr>
<td>A7</td>
<td>$\sigma^{\text{rel}}<em>{a}(x) + \sigma^{\text{rel}}</em>{b}(y) = \sigma^{\text{rel}}_{a+y}$</td>
</tr>
</tbody>
</table>

composition operator $\cdot$, and the relative delay operator $\sigma^{\text{rel}}_{a}$ which specifies a delay of $p$ time. In [BM99], also a time-out operator $v^{\text{rel}}_{a}$ and an initialisation operator $\mu^{\text{rel}}_{a}$ are introduced, but these are omitted here. The axioms of $\text{BPA}^{\text{st}}$ are presented in Table 1.

2.2 Axiomatisation

We start with the presentation of Basic Process Algebra with dense relative timing and neutral elements for alternative and sequential composition, called $\text{BPA}^{\text{st}}$. This process algebra is based on the algebra $\text{BPA}^{\text{st}}$ of [BM99] and extends this by the constant $\epsilon$. Using $\epsilon$ the delay operators of [BM99] can be replaced by delay constants. This gives rise to less axioms and a clearer separation between describing actions and describing timing of these. This is in accordance with the process algebras $\text{ACP}^{\text{st}}$ [Gro91] which contains the constants $t$ and $\delta$ representing a discrete unit delay and an arbitrary discrete delay respectively, Timed CSP [RR88] which has a construct "WAIT $t$", and LOTOS-T [MFV93] which has a construct "exit $\{t\}$".

We have the following syntax and informal interpretation:

- **alternative composition** or sum, written $+$. The choice is resolved by the execution of an action, not by the mere passage of time (see axiom SRT3, Table 2).
- **sequential composition** or product, written $\cdot$. Relative timing means that time is measured from the execution of the previous action if any, and otherwise from the time of initialisation of the process.
- **atomic actions** $a$, where $a \in A$, a given set of atoms. The process $\hat{a}$ denotes immediate and urgent execution of the action $a$, at the current moment of time, followed by immediate and successful termination, i.e. execution of action takes no time.
- **undelayable deadlock** $\overline{a}$. Time cannot progress beyond the current moment of time, and no successful termination can take place. However, the current moment of time is a consistent state (i.e. the process is still 'alive').
Table 2: Axioms of $\text{BPA}\text{\_rt}$ ($a \in A_{\delta}, p, q \geq 0, r > 0$)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Axiom</th>
<th>Axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + y = y + x$</td>
<td>A1</td>
<td>$x + \delta = x$</td>
</tr>
<tr>
<td>$(x + y) + z = x + (y + z)$</td>
<td>A2</td>
<td>$\delta \cdot x = \delta$</td>
</tr>
<tr>
<td>$x + x = x$</td>
<td>A3</td>
<td>$\delta \cdot x = x$</td>
</tr>
<tr>
<td>$(x + y) \cdot z = x \cdot z + y \cdot z$</td>
<td>A4</td>
<td>$x \cdot \delta = x$</td>
</tr>
<tr>
<td>$(x \cdot y) \cdot z = x \cdot (y \cdot z)$</td>
<td>A5</td>
<td>$\delta \cdot x = \delta$</td>
</tr>
<tr>
<td>$\overline{\frac{a}{b}} + \delta = \overline{\delta}$</td>
<td>A6a</td>
<td>$\overline{\frac{a^0}{\delta}} = \delta$</td>
</tr>
<tr>
<td>$\overline{\frac{a^p + \delta}{\delta}} = \overline{\delta}$</td>
<td>A6b</td>
<td>$\overline{\frac{a^p \cdot \delta^q}{\delta}} = \overline{a^{p+q}}$</td>
</tr>
<tr>
<td>$\overline{\frac{a^p}{\delta}} \cdot \overline{\frac{x + y}{\delta}}$</td>
<td>A7SR</td>
<td>$\overline{\frac{a^p}{\delta}} \cdot \overline{\frac{x + y}{\delta}} = \overline{a^p \cdot \delta \cdot \overline{\frac{x + y}{\delta}}}$</td>
</tr>
</tbody>
</table>

- the **deadlocked process** $\delta$. The process $\delta$ denotes a process that has already deadlocked. The current moment of time cannot be reached by just idling. This is the neutral element of alternative composition (see axiom A6ID in Table 2).

- the **terminated process** $\delta$. The terminated process is the neutral element of sequential composition (see axioms A8IE, A9IE) and the remainder of $a$ after action execution.

- the **relative delay constant** $\overline{\frac{a^p}{\delta}}$. This constant describes a delay of an amount of time $p$ where $p$ is a non-negative real number. When $p = 0$, nothing is left (see axiom IE).

Sequential composition binds stronger than alternative composition. In the remainder of this paper we will write $A_{\delta}$ for $A \cup \{\delta\}$. The set of all closed process terms constructed using the above syntax is denoted $P$. The axioms of $\text{BPA}\text{\_rt}$ are given in Table 2.

The axioms A1-A5 describe straightforward properties of alternative and sequential composition. Alternative composition is commutative, associative, and idempotent, sequential composition distributes over alternative composition from the right, and sequential composition is associative.

The axioms A6ID-A7ID explain that the deadlocked process is a neutral element for alternative composition and a left-zero element for sequential composition. Intuitively these state that the choice for the deadlocked process is only made if it cannot be avoided and that after a deadlocked process nothing happens anymore. The axioms A8IE-A9IE explain that the terminated process is a neutral element for sequential composition.

The axioms A6a, A6b, and A7SR can be used to prove that for closed terms the constant undelayable deadlock is a quasi-neutral element for alternative composition: this means that $x + \delta = x$ provided $x \neq \delta$ and $x \neq \delta$.

Axiom SRT2 describes that two sequentially composed relative delays can be combined into one by adding the reals indicating the delays. This axiom expresses that we are using relative time. Axiom SRT3 describes that the passage of time by itself cannot determine a choice. This principle is often referred to as *time-determinism*. It describes that the passage of time is
really different from the execution of an atomic action. Axiom IE expresses that the constant \( \sigma^0 \) is equivalent to the terminated process \( \dot{\epsilon} \).

If we compare the axioms of BPA\(_{\text{st}}\) with the axioms of BPA\(_{\text{st}}\) [BM99] we find that there are not many differences. The axioms A1-A5, A6ID-A7ID, A6a, and A7SR are identical. The axioms SRT2-SRT3 and A6b are only different in the sense that we now have the constant \( \sigma^p \) instead of the operator \( o-:el \).

Because of the strong relationship between the unary operator \( \sigma^p_{\text{rel}} \) from BPA\(_{\text{st}}\) and the constant \( \sigma^p \) from BPA\(_{\text{st}}\), and the wish to maintain \( \sigma^0_{\text{rel}}(x) = x \), we strive for \( \sigma^0 \cdot x = x \). This is achieved by considering \( \sigma^0 \) and \( \dot{\epsilon} \) equivalent. This is expressed by axiom IE. Then, axiom SRT1 of BPA\(_{\text{st}}\) \( (\sigma^0_{\text{rel}}(x) = x) \) is in the setting of BPA\(_{\text{st}}\) derivable from axioms IE and A8IE. Axiom SRT4 of BPA\(_{\text{st}}\) \( (\sigma_{\text{rel}}(x \cdot y) = \sigma_{\text{rel}}(x) \cdot y) \) is not necessary anymore as it is part of axiom A5, again due to the fact that we have a delay constant instead of an operator.

As we have just explained, the terminated process \( \dot{\epsilon} \) is expressible in terms of another constant. It is, however, impossible to express the delay constant in terms of the operator without using something like the terminated process.

The reader might be surprised that we have not introduced a constant \( \bar{\delta} \), standing for undelayable termination. We have not done this as it can be expressed in terms of undelayable deadlock and the terminated process: \( \bar{\delta} = \delta + \dot{\epsilon} \).

### 2.3 Operational semantics

The definition of an operational semantics by means of SOS deduction rules is standard. By means of these rules, we define binary relations \( \xrightarrow{a} \) and \( \xrightarrow{t} \) and unary relations \( \downarrow \) and \( \uparrow \) on closed terms from \( P \) (for \( a \in A \) and \( t > 0 \)). Intuitively, they have the following meaning:

1. \( x \xrightarrow{a} x' \) means that \( x \) evolves into \( x' \) by executing atomic action \( a \);
2. \( x \xrightarrow{t} x' \) means that \( x \) evolves into \( x' \) by waiting for time \( t \);
3. \( x \downarrow \) means that \( x \) has an option to terminate successfully, it holds for all processes which contain \( \dot{\epsilon} \) as an alternative; and
4. \( x \uparrow \) means that \( x \) has terminated.

The predicate \( \uparrow \) discriminates between \( \bar{\delta} \) and \( \dot{\delta} \) and the predicate \( \downarrow \) discriminates between \( \dot{\delta} \) and \( \dot{\epsilon} \).

For BPA\(_{\text{st}}\) similar predicates and relations have been used. There are two differences. First, we replace the immediate deadlock predicate ID by the predicate terminated \( \uparrow \). Their meaning is the same however if we only consider the part of the signature that the two process algebras have in common. Second, the introduction of the empty process also has a positive side for the description of the operational semantics of the process algebra as it allows the separation
of the action step relation and the termination predicate. This means that, instead of the relation \( \rightarrow \) and the predicate \( \Rightarrow \), we now have the relation \( \rightarrow \) and the predicate \( \downarrow \).

In the literature two ways can be found to enforce time-determinism. The first way is to formulate the deduction rules in such a way that the following property holds for all closed terms \( x \): if \( x \xrightarrow{r} x' \) and \( x \xrightarrow{s} x'' \) for some closed process terms \( x' \) and \( x'' \) and positive real number \( t \), then \( x' \equiv x'' \). In other words, for each amount of time, there is at most one transition. The second way is to enforce time-determinism in the notion of equivalence that is used. This means that the deduction rules do not have to guarantee the above property and that processes with different time transitions can still be considered equivalent.

In BPA\( ^{st} \) time-determinism is enforced in the term deduction system explicitly by adopting the following deduction rules for \( + \):

\[
\frac{x \xrightarrow{r} x' \quad y \xrightarrow{s} y'}{x + y \xrightarrow{r+s} x' + y'} \quad \frac{x \xrightarrow{r} x' \quad y \not\xrightarrow{r}}{x + y \xrightarrow{r} x'} \quad \frac{y \not\xrightarrow{s}}{x + y \xrightarrow{r} y'}
\]

For BPA\( _2^{st} \) similar deduction rules for \( + \) can be defined, but due to the existence of a neutral element for sequential composition we get an intricate interplay between time transitions and termination of sequentially composed processes. Three cases can occur to conclude that a process of the form \( x \cdot y \) can let time pass for an amount \( t \):

1. \( x \) can let the full amount of time pass on its own.
2. \( x \) can terminate immediately and \( y \) can let the full amount of time pass.
3. \( x \) can let a portion of the amount of time pass, immediately followed by termination, and \( y \) can let the remainder of the amount of time pass.

These three cases are reflected in Table 3 by deduction rules [21], [22], and [23]. Observe that these deduction rules are not pairwise disjoint. As a consequence it is possible to derive more than one time transition \( \xrightarrow{t} \) for some processes. In the case that we wish to enforce time-determinism in the deduction system, we have to reformulate these deduction rules in such a way that at most one time transition \( \xrightarrow{t} \) can be deduced for each \( t \).

This however is in general impossible. For, in order to determine the process term \( q \) a term of the form \( x \cdot y \) evolves into after waiting for time \( t \) \( (x \cdot y \xrightarrow{t} q) \), we have to consider all pairs of positive reals \( r \) and \( s \) such that \( r + s = t \), and see if there are terms \( x' \) and \( y' \) such that \( x \xrightarrow{r} x' \), \( x' \xrightarrow{s} \), and \( y \xrightarrow{s} y' \). All possible \( y' \) must be summands of the term \( q \). In discrete time with empty process, the number of possibilities is finite and case distinction works [BV97].

In Section 4.3 we introduce initial delay constants \( \bar{\sigma}^{(0,p)} \). Such a constant describes a process that terminates successfully after the passage of an arbitrary amount of time from the interval \([0,p)\). Thus, potentially, this gives rise to an infinite number of time transitions. Consider for example the process term \( \bar{\sigma}^{(0,p)} \cdot (\bar{\sigma}^{(0,q)} \cdot \bar{a}) \). For \( 0 \leq t < p + q \) the following time transitions are possible (amongst others): \( \bar{\sigma}^{(0,p)} \cdot (\bar{\sigma}^{(0,q)} \cdot \bar{a}) \xrightarrow{t} \bar{\sigma}^{(0,q-t_2)} \cdot \bar{a} \) for all \( t_2 \) such that \( 0 \leq t - t_2 < p \) and \( 0 \leq t_2 < q \).
Table 3: Deduction rules for BPA with relative real time (\(a \in A, r, s > 0, p \geq 0\))

\[
\begin{align*}
&\text{[1]} \ \bar{a} \xrightarrow{a} \bar{e} \\
&\text{[2]} \ \bar{e} \downarrow \\
&\text{[3]} \ \bar{0} \downarrow \\
&\text{[4]} \ \bar{e} \uparrow \\
&\text{[5]} \ \bar{e} \uparrow \\
&\text{[6]} \ \bar{0} \uparrow \\
&\text{[7]} \ \bar{p} \xrightarrow{p+r} \bar{p} \\
&\text{[8]} \ \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \\
&\text{[9]} \ \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'} \\
&\text{[10]} \ \frac{x \downarrow}{x + y \downarrow} \\
&\text{[11]} \ \frac{y \downarrow}{x + y \downarrow} \\
&\text{[12]} \ \frac{x \uparrow, y \uparrow}{x + y \uparrow} \\
&\text{[13]} \ \frac{x \xrightarrow{r} x'}{x + y \xrightarrow{r} x'} \\
&\text{[14]} \ \frac{x \xrightarrow{r} y'}{x + y \xrightarrow{r} y'} \\
&\text{[15]} \ \frac{x \xrightarrow{a} x'}{x \xrightarrow{y} x' \cdot y} \\
&\text{[16]} \ \frac{x \xrightarrow{y} y'}{x \cdot y \xrightarrow{y} y'} \\
&\text{[17]} \ \frac{x \uparrow, y \downarrow}{x \cdot y \uparrow} \\
&\text{[18]} \ \frac{x \uparrow, y \downarrow}{x \cdot y \uparrow} \\
&\text{[19]} \ \frac{x \downarrow, y \downarrow}{x \cdot y \downarrow} \\
&\text{[20]} \ \frac{x \xrightarrow{r} x'}{x \cdot y \xrightarrow{r} x' \cdot y} \\
&\text{[21]} \ \frac{x \xrightarrow{r} y'}{x \cdot y \xrightarrow{r} y'} \\
&\text{[22]} \ \frac{x \xrightarrow{r} x', x' \downarrow, y \xrightarrow{r} y'}{x \cdot y \xrightarrow{r} x' \cdot y}
\end{align*}
\]

Therefore, we will not attempt to explicitly incorporate time-determinism in the deduction system. Rather, we formulate the time-determinism principle indirectly in the definition of equivalence (bisimilarity). This cannot be achieved by adopting the normal definition of strong bisimilarity. This can be seen as follows. Consider the processes \(\bar{x} \cdot (a + b)\) and \(\bar{a} \cdot \bar{x} \cdot \bar{b}\).

The first has only one possibility for delaying for 1 time unit: \(\bar{x} \cdot (a + b) \xrightarrow{1} \bar{x} \cdot (\bar{a} + \bar{b})\).

The second has two such possibilities: \(\bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b} \xrightarrow{1} \bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b}\) and \(\bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b} \xrightarrow{1} \bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b}\).

Neither one of these resulting processes is strongly bisimilar with \(\bar{x} \cdot (a + b)\).

The addition of a deduction rule combining the time transitions such as deduction rule [33] is not sufficient as the transitions \(\bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b} \xrightarrow{1} \bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b}\) and \(\bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b} \xrightarrow{1} \bar{x} \cdot \bar{a} \cdot \bar{x} \cdot \bar{b}\) are still present and can still not be matched.

Hence, we consider another definition of equivalence. For this purpose we can have a look at the notion of \(\sigma\)-bisimilarity as it was used in the Algebra of Timed Frames [BFM96]. There, time transitions are defined separately and time-determinism is enforced in the definition of \(\sigma\)-bisimilarity by requiring that for any pair of related processes \(s\) and \(t\): if a process \(s\) can do a time transition, then the set of all such time transitions from \(s\) must be related to the set of all such time transitions from the process \(t\). We present the deduction rules in Table 3 and we define \(\sigma\)-bisimilarity for BPA\(_{\text{rt}}^\varepsilon\).

**Definition 1 (\(\sigma\)-bisimilarity)** A symmetric, binary relation \(R \subseteq \mathcal{P}(P) \times \mathcal{P}(P)\) is called a \(\sigma\)-bisimulation relation if and only if for all \(s, t \subseteq P\) with \((s, t) \in R\) we have

- for all \(s \in S\): if \(s \downarrow\) then there exists \(t \in T\) such that \(t \downarrow\);
• if \( s \uparrow \) for all \( s \in S \), then \( t \uparrow \) for all \( t \in T \);

• for all \( s \in S \) and \( a \in A \): if \( s \xrightarrow{a} s' \) for some \( s' \in P \), then there exists \( t \in T \) such that \( t \xrightarrow{a} t' \) for some \( t' \in P \) and \( (\{s'\}, \{t'\}) \in R \);

• for all \( s \in S \) and \( r > 0 \): if \( s \xrightarrow{r} s' \) for some \( s' \in P \), then there exists \( t \in T \) such that \( t \xrightarrow{r} t' \) for some \( t' \in P \) and \( (\{s', \exists s \in S \xrightarrow{r} s\}', \{t', \exists t \in T \xrightarrow{r} t'\}) \in R \);

Two processes \( p \) and \( q \) are \( \sigma \)-bisimilar, notation \( p \leftrightarrow_{\sigma} q \), if there exists a \( \sigma \)-bisimulation relation \( R \) such that \( (\{p\}, \{q\}) \in R \).

For the example processes \( (a+b) \) and \( (a+b) \), the time transitions are the ones described before. This time however, the results of the time transitions do not have to be related in isolation. Instead, the sets \( \{a \cdot (a+b)\} \) and \( \{a \cdot a, a \cdot b\} \) have to be related.

Although the above definition of \( \sigma \)-bisimilarity serves our purposes in identifying "equivalent" processes, we are not satisfied with it for several reasons. First, in \( \sigma \)-bisimulations not only processes have to be related, but also sets of processes. This is counterintuitive. Second, the definition of \( \sigma \)-bisimilarity does not allow to use the meta-theory for obtaining congruence and completeness results presented in [BV95]. With this formulation of equivalence of processes the proof that it is a congruence for the operators of the process algebra and the proof that the axiomatisation of \( \sigma \)-bisimilarity is complete are long and tedious. In the process algebra BPA \( \sigma \) there was no need for \( \sigma \)-bisimilarity; there strong bisimilarity (in the terminology of [BV95]) could be used.

Because of these drawbacks we looked for a different solution. Instead of bundling all outgoing time transitions into one time transition to the sum of the terms, we can also get bisimilar systems by saturating the transition systems, by putting in extra time transitions. The idea of saturating transition systems dates back to Bergstra and Klop [BK88] and Van Glabbeek [Gla87] where saturation with silent steps was used to turn weakly bisimilar transition systems into saturated strongly bisimilar systems. Here, we adapt this idea to turn \( \sigma \)-bisimilar systems into saturated strongly bisimilar systems. We just define time transitions without regard to the existence of other time transitions. Next, we saturate in two ways: whenever we have from one process term two time transitions to two different terms, we add a time transition to the sum of the terms (deduction rule [33]), and whenever we have a time transition to a term that has different summands, we add a time transition to each of its summands (deduction rule [34]). On the resulting, saturated transition systems we can use strong bisimulation.

We extend the operational semantics with a relation \( \preceq \) which expresses that a process term is a summand of another process term (see Table 4). We claim that this relation is a partial order. From these deduction rules we can derive that \( \sigma^1 \cdot (a+b) \xrightarrow{1} \sigma^0 \cdot a \) as follows. First, from the deduction rules [8] and [20] we obtain \( \sigma^1 \cdot (a+b) \xrightarrow{1} \sigma^0 \cdot (a+b) \). Second, from the deduction rules [23] and [27] we obtain \( a \preceq a+b \). Finally, combining these results using deduction rule [33] gives the desired result.

**Definition 2 (Strong bisimilarity)** A symmetric, binary relation \( R \subseteq P \times P \) is called a strong bisimulation relation if for all \( s, t \in P \) such that \( (s, t) \in R \) we have...
Table 4: Deduction rules for BPA with relative real time ($a \in A, r, s > 0, p \geq 0$)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>$x \preceq x$</td>
</tr>
<tr>
<td>24</td>
<td>$\delta \preceq x$</td>
</tr>
<tr>
<td>25</td>
<td>$\frac{x \gamma}{\delta} \preceq x$</td>
</tr>
<tr>
<td>26</td>
<td>$\frac{x \downarrow}{\varepsilon} \preceq x$</td>
</tr>
<tr>
<td>27</td>
<td>$x' \preceq x$ $x' \preceq x + y$</td>
</tr>
<tr>
<td>28</td>
<td>$\frac{y' \preceq y}{y'} \preceq x + y$</td>
</tr>
<tr>
<td>29</td>
<td>$\frac{x' \preceq x}{x' + y'} \leq x + y$ $\frac{y' \preceq y}{x' + y'} \leq x + y$</td>
</tr>
<tr>
<td>30</td>
<td>$\frac{x \preceq y}{x \cdot z \preceq y \cdot z}$</td>
</tr>
<tr>
<td>31</td>
<td>$\frac{x \downarrow y \preceq z}{y \preceq x \cdot z}$</td>
</tr>
<tr>
<td>32</td>
<td>$\frac{x \preceq y}{\frac{y}{\sigma \cdot x} \preceq \frac{y}{\sigma \cdot y}}$</td>
</tr>
<tr>
<td>33</td>
<td>$\frac{x \uparrow x' x \uparrow x''}{x \uparrow x' x \uparrow x''}$</td>
</tr>
<tr>
<td>34</td>
<td>$\frac{x \uparrow x' x'' \preceq x'}{x \uparrow x''}$</td>
</tr>
</tbody>
</table>

- if $s \downarrow$ then $t \downarrow$;
- if $s \uparrow$ then $t \uparrow$;
- for all $a \in A$ and $s' \in P$: if $s \Rightarrow s'$, then there exists $t' \in P$ such that $t \Rightarrow t'$ and $(s', t') \in R$;
- for all $r \in R \geq 0$ and $s' \in P$: if $s \vdash r s'$, then there exists $t' \in P$ such that $t \vdash r t'$ and $(s', t') \in R$;
- for all $s' \in P$: if $s' \preceq s$, then there exists $t' \in P$ such that $t' \preceq t$ and $(s', t') \in R$.

Two processes $p$ and $q$ are strongly bisimilar, notation $p \equiv q$, if there exists a strong bisimulation relation $R$ such that $(p, q) \in R$. If a relation $R$ is given that witnesses the strong bisimilarity of processes $p$ and $q$, then we write $R : p \equiv q$.

From the deduction rules we obtain the following time transitions with delay 1 for the process $\sigma^{1} \cdot (\bar{a} + \bar{b})$:

\[
\begin{align*}
\sigma^{1} \cdot (\bar{a} + \bar{b}) & \Rightarrow \sigma^{0} \cdot (\bar{a} + \bar{b}), \\
\sigma^{1} \cdot (\bar{a} + \bar{b}) & \Rightarrow \sigma^{0} \cdot \bar{a}, \\
\sigma^{1} \cdot (\bar{a} + \bar{b}) & \Rightarrow \sigma^{0} \cdot \bar{b}, \\
\sigma^{1} \cdot (\bar{a} + \bar{b}) & \Rightarrow \sigma^{0} \cdot \delta,
\end{align*}
\]

and many more syntactically different, but semantically equivalent transitions. Similarly, for the process $\sigma^{1} \cdot \bar{a} + \sigma^{1} \cdot \bar{b}$ we obtain the following time transitions with delay 1:

\[
\begin{align*}
\sigma^{1} \cdot \bar{a} + \sigma^{1} \cdot \bar{b} & \Rightarrow \sigma^{0} \cdot \bar{a}, \\
\sigma^{1} \cdot \bar{a} + \sigma^{1} \cdot \bar{b} & \Rightarrow \sigma^{0} \cdot \bar{b}, \\
\sigma^{1} \cdot \bar{a} + \sigma^{1} \cdot \bar{b} & \Rightarrow \sigma^{0} \cdot \delta,
\end{align*}
\]
and many more syntactically different, but semantically equivalent transitions. Now, the time transitions of the processes do match as required by the definition of strong bisimilarity.

By saturating the transition systems, we have obtained that the time determinism axiom SRT3 is valid on our transition systems. Thus, we can use the strong bisimulation relation induced by the relations and predicates of our term deduction systems, and all the standard results on term deduction systems (such as the congruence property, and conservative extension results) become easily available. We feel this gives a definite advantage over the cumbersome to use notion of $\sigma$-bisimilarity.

A disadvantage of using two predicates and three relations in the term deduction system and the definition of strong bisimilarity is that proving soundness of the axioms becomes a laborious task. The following two lemmata show that the predicates $\downarrow$ and $\uparrow$ need not be considered in such proofs. This does not mean that these predicates can be removed altogether from the term deduction system. They are used in defining the $\preceq$-relation.

**Lemma 3** For all closed $\text{BPA}^\text{rt}_i$-terms $p$ we have that $p \downarrow$ if and only if $\hat{p} \preceq p$.

**Proof** The "only if" part follows from deduction rule [26]. The "if" part is easily proven by induction on the depth of the derivation of $\hat{p} \preceq p$.

**Lemma 4** For all closed $\text{BPA}^\text{rt}_i$-terms $p$ we have that $p \uparrow$ if and only if $\hat{p} \preceq p$.

**Proof** The "only if" part is proven as follows. Suppose that $\hat{p} \preceq p$. Suppose that $p \not\gamma$. By deduction rule [25] we then have $\hat{p} \preceq p$. This leads to a contradiction and hence we have $p \uparrow$. For the "if" part we have the following reasoning. Suppose that $p \uparrow$. Suppose that $\hat{p} \preceq p$. By induction on the depth of the proof of $\hat{p} \preceq p$ we easily find a contradiction in all cases. Hence, we conclude that $\hat{p} \not\preceq p$.

Now that we have introduced different notions of equivalence, we feel the need to express the relation between them. The next theorem states that strong bisimilarity implies $\sigma$-bisimilarity. The inverse also holds.

**Theorem 5** For all closed $\text{BPA}^\text{rt}_\sigma$-terms $p$ and $q$ we have that $p \preceq q$ implies $p \preceq \sigma q$.

**Proof** Assume that $R : p \preceq q$. Define $R_\sigma$ as follows:

$$R_\sigma = \{(S, T) \mid \forall s \in S \exists t \in T (s, t) \in R \land \forall t \in T \exists s \in S (s, t) \in R\}.$$ 

We prove that $R_\sigma$ is a $\sigma$-bisimulation relation. Consider an arbitrary pair $(S, T) \in R_\sigma$. Suppose $s \downarrow$ for some $s \in S$. Then, obviously there exists $t \in T$ such that $(s, t) \in R$. Hence, $t \downarrow$. Suppose $s \uparrow$ for all $s \in S$. Then, obviously $t \uparrow$ for all $t \in T$. Suppose $s \overset{\sigma}{\rightarrow} s'$ for some $s \in S$. Then there exists $t \in T$ such that $(s, t) \in R$. Hence $t \overset{\sigma}{\rightarrow} t'$ for some $t'$. Furthermore we must prove that $S' = \{s' \mid \exists s \in S s \overset{\sigma}{\rightarrow} s'\}$ and $T' = \{t' \mid \exists t \in T t \overset{\sigma}{\rightarrow} t'\}$ are related by $R_\sigma$. This is obviously the case.
Theorem 6  For all closed \( \text{BPA}^{\text{st}} \)-terms \( p \) and \( q \) we have \( p \equiv \sigma q \) implies \( p \equiv q \).

Proof Assume \( R_\sigma : \{ p \} \equiv \sigma \{ q \} \). For each set \( S \) of closed process terms, we define the set \( S^{\text{sat}} \) as the smallest set that satisfies: \( S \subseteq S^{\text{sat}} \), if \( s_1, s_2 \in S^{\text{sat}} \), then \( s_1 + s_2 \in S^{\text{sat}} \), and if \( s \in S^{\text{sat}} \) and \( s' \leq s \), then \( s' \in S^{\text{sat}} \). Define the relation \( R_\sigma^{\text{sat}} \) as follows: if \((S, T) \in R_\sigma \), then \((S^{\text{sat}}, T^{\text{sat}}) \in R_\sigma^{\text{sat}} \). Now it is our claim that \( R_\sigma^{\text{sat}} \) is also a \( \sigma \)-bisimulation relation. The saturation of \( S \) (and \( T \)) to \( S^{\text{sat}} \) (and \( T^{\text{sat}} \)) corresponds exactly to the saturation with time transitions performed in the deduction system. This relation \( R_\sigma^{\text{sat}} \) has the property that whenever \((S^{\text{sat}}, T^{\text{sat}}) \in R_\sigma^{\text{sat}} \) and \( s \in S^{\text{sat}} \), there exists \( t \in T^{\text{sat}} \) such that \( s \equiv \sigma t \).

Thus we can even further extend the relation \( R_\sigma^{\text{sat}} \) by adding the pairs \( \{ \{ s \}, \{ t \} \} \). Now define the relation \( R \) as follows: \( (s, t) \in R \) if and only if \( \{ \{ s \}, \{ t \} \} \in R_\sigma^{\text{sat}} \).

The notion of strong bisimilarity on closed \( \text{BPA}^{\text{st}} \)-terms is a congruence for all the operators of the process algebra \( \text{BPA}^{\text{st}} \). First, however, we prove that strong bisimilarity is an equivalence.

Theorem 7 (Equivalence)  Strong bisimilarity is an equivalence relation.

Proof We have to show that strong bisimilarity is reflexive, symmetric, and transitive. The first and second are trivial and therefore omitted. We prove that strong bisimilarity is transitive. Suppose that \( R_1 : x \leftrightarrow y \) and \( R_2 : y \leftrightarrow z \). Let \( R = R_1 \circ R_2 \). We prove that \( R \) is a strong bisimulation. Consider an arbitrary pair \( (p, r) \in R \). As \( (p, r) \in R \) we have the existence of \( q \) such that \((p, q) \in R_1 \) and \((q, r) \in R_2 \).

- Suppose \( p \xrightarrow{\delta} p' \). As \((p, q) \in R_1 \) we have the existence of \( q' \) such that \( q \xrightarrow{\delta} q' \) and \((p', q') \in R_1 \). As \((q, r) \in R_2 \) we have the existence of \( r' \) such that \( r \xrightarrow{\delta} r' \) and \((q', r') \in R_2 \).

Hence, we also have the existence of \( r' \) such that \( r \xrightarrow{\delta} r' \) and \((p', r') \in R \).

- Suppose \( p \xrightarrow{\sigma} p' \). Similar to the previous case.

Hence, the relation \( R \) satisfies all conditions of a strong bisimulation relation as given in Definition 2.

Definition 8  We define the norm \(|.|\) on closed terms as follows: for closed terms \( x \) and \( y \), \( a \in A_\delta \), and \( p \in R_0 \):

\[
|\delta| = |\sigma| = |\delta| = |\varepsilon| = 1, \quad |x \cdot y| = |x| \times |y| + 1, \\
|\sigma p| = p + 1, \quad |x + y| = |x| + |y|.
\]

Theorem 9 (Congruence)  Strong bisimilarity is a congruence for the operators of the process algebra \( \text{BPA}^{\text{st}} \).

Proof  In order to prove that strong bisimilarity is a congruence we claim that the term deduction system presented is isomorphic with the term deduction system obtained
by omitting the delay constants and adding unary delay operators $\sigma^p_{rel}$ with the following deduction rules:

$$
\frac{\vdash x}{\sigma^0_{rel}(x) \downarrow} \quad \frac{\vdash x}{\sigma^0_{rel}(x) \uparrow} \quad \frac{x \xrightarrow{\sigma^0_{rel}(x)} x'}{\sigma^p_{rel}(x) \xrightarrow{a} x'} \quad \frac{\sigma^p_{rel}(x) \xrightarrow{\vdash} \sigma^p_{rel}(x)}{\sigma^p_{rel}(x) \xrightarrow{\vdash} \sigma^p_{rel}(x)}
$$

Actually these deduction rules are obtained from the deduction rules of the original term deduction system by considering $\sigma^p_{rel}(x)$ and $\sigma^p \cdot x$ equivalent. The term deduction system constructed in this way is stratifiable and in path format. Then, as a consequence we find that strong bisimilarity is a congruence [Ver95]. The stratification is defined as follows: for closed terms $x$ and $x'$, $\alpha \in \mathcal{A}$, and $r > 0$

$$
S(x \downarrow) = |x|, \\
S(x \uparrow) = |x|, \\
S(x \xrightarrow{\alpha} x') = |x|, \\
S(x \xrightarrow{\vdash} x') = 2\omega + |x|, \\
S(x \preceq x') = \omega + |x'|.
$$

For each deduction rule of the term deduction system we have to prove that each of it's positive hypotheses is not more complex than the conclusion and that each of it's negative hypothesis is strictly less complex than the conclusion. As an example thereof, for deduction rule [25] from Table 4 we have to prove that $S(x \uparrow) < S(\delta \preceq x)$ for arbitrary closed term $x$. As $S(x \uparrow) = |x|$ and $S(\delta \preceq x) = \omega + |x|$ this is the case.

As another example, for deduction rule [34] from Table 4 we have to prove that $S(x \xrightarrow{\vdash} x') \preceq S(x \xrightarrow{\vdash} x''')$ and $S(x'' \preceq x') \preceq S(x \xrightarrow{\vdash} x')$. This is obviously the case.

### 2.4 Soundness and completeness

In this section, we establish that the structure of transition systems modulo strong bisimilarity is a model for our axioms, or, put differently, that our axioms are sound with respect to the set of closed terms modulo strong bisimilarity. We also prove that the axiomatisation is complete.

**Theorem 10 (Soundness)** The process algebra $BPA_{rt}$ is a sound axiomatisation of strong bisimilarity on closed $BPA_{rt}$-terms.

First, we define a notion of basic terms. These are useful in the proofs to come. It turns out that every closed term is derivably equal to a basic term. This provides us with an easier to use means of induction: instead of proving properties by induction on the structure of closed terms, it suffices to prove these properties by induction on the structure of basic terms. The proof that the axiomatisation is complete with respect to the set of closed terms modulo strong bisimilarity uses this induction principle.
**Definition 11** The set of all basic terms is the smallest set $B$ that satisfies

1. $\delta \in B$;
2. $\epsilon \in B$;
3. for $a \in A_\delta$ and $x \in B$: $\tilde{a} \cdot x \in B$;
4. for $p > 0$ and $x \in B$: $\tilde{\sigma}^p \cdot x \in B$;
5. for $x, y \in B$: $x + y \in B$.

The following theorem states that every closed term is derivably equal to a basic term. The virtue of this theorem is that in future proofs we are allowed to apply induction on the structure of basic terms instead of on the structure of closed terms, thereby limiting the number of cases to be considered. An example of a proof in which induction on the structure of basic terms is used, is the proof of the completeness theorem (Theorem 13).

**Theorem 12 (Elimination)** For every closed term $s$ there exists a basic term $t$ such that $s = t$.

**Proof** Easy; by induction on the structure of the closed term $s$. We used the following lemma: for every two basic terms $t_1$ and $t_2$ there exists a basic term $t_3$ such that $t_1 \cdot t_2 = t_3$. This lemma is easily proven with induction on basic term $t_1$.

As alternative composition is commutative and associative and a neutral element is available ($\delta$), we can define a generalised alternative composition operator $\sum$. For a finite index set $I$ and process terms $p_i$, we define $\sum_{i \in I} p_i$ as the alternative composition of the process terms $p_i$. If $I = \emptyset$ we define $\sum_{i \in I} p_i = \delta$. The operator $\sum$ binds stronger than $+$ and weaker than all other operators.

As a consequence of the above elimination theorem we also have that every closed term $x$ can be written as

$$\sum_{i \in I} \tilde{a}_i \cdot x_i + \tilde{\sigma}^u \cdot x' + \sum_{j \in J} \epsilon_j,$$

where $I$ and $J$ are finite index sets, $a_i \in A_\delta$ for all $i \in I$, $u \geq 0$, and $x_i$ and $x'$ are basic terms. This can easily be proven.

**Theorem 13 (Completeness)** The process algebra $\text{BPA}_t^{\text{frt}}$ is a complete axiomatisation of strong bisimilarity on closed $\text{BPA}_t^{\text{frt}}$-terms.

**Proof** For a proof of this theorem we refer to Appendix A.
3 Parallelism and communication

In this section, we define some auxiliary operators and extend the theory with parallel composition. The auxiliary operators help in the axiomatisations to come, but are also useful in their own right.

3.1 Encapsulating actions and time

First of all, we have the encapsulation operator $\partial_H$, well-known from standard process algebra. The operator $\partial_H$ will block all actions from the set of actions $H (H \subseteq A)$. Time can go forward, and actions outside $H$ can occur. The operator is used to enforce communication between parallel components. The “now” operator $\nu_{rel}$ will block initial passage of time, forcing a process to start with an action at the current moment of time. The $\nu_{rel}$ is a timeout operator: in $\nu_{rel}^p(x)$, the first action of $x$ must occur within $p$ time units from now.

**Definition 14** We extend the definition of $|.|$ on closed terms as follows: for closed terms $x$, $H \subseteq A$, and $p \in R^{\geq 0}$

$$|\partial_H(x)| = |\nu_{rel}^p(x)| = |\nu_{rel}(x)| = |x|.$$

**Theorem 15 (Congruence)** Strong bisimilarity is a congruence for the operators of the process algebra presented in this section.

**Proof** The term deduction is still stratifiable and in panth format. Therefore congruence follows immediately.

The axioms given in Table 6 are sound, the operators can be eliminated from every closed term.
Table 6: Axioms of the auxiliary operators \((a \in A, p, q \geq 0, r > 0)\)

\[
\begin{align*}
\partial_H(\delta) &= \delta \\
\partial_H(\epsilon) &= \epsilon \\
\partial_H(\alpha \cdot x) &= \alpha \cdot \partial_H(x) & \text{if } a \notin H \\
\partial_H(\alpha \cdot x) &= \delta & \text{if } a \in H \\
\partial_H(\alpha^r \cdot x) &= \alpha^r \cdot \partial_H(x) \\
\partial_H(x + y) &= \partial_H(x) + \partial_H(y)
\end{align*}
\]

\[
\begin{align*}
u_{rel}^p(\delta) &= \delta \\
u_{rel}(\delta) &= \delta \\
u_{rel}(\epsilon) &= \epsilon \\
u_{rel}(\epsilon) &= \epsilon \\
u_{rel}(\alpha \cdot x) &= \alpha \cdot x \\
u_{rel}(\alpha^r \cdot x) &= \alpha^r \cdot \partial_H(x) \\
u_{rel}(\alpha^r \cdot x) &= \delta \\
u_{rel}(x + y) &= \nu_{rel}(x) + \nu_{rel}(y) \\
u_{rel}(x + y) &= \nu_{rel}(x) + \nu_{rel}(y)
\end{align*}
\]

**Theorem 16 (Soundness)** The process algebra presented in this section is a sound axiomatisation of strong bisimilarity on closed terms.

**Theorem 17 (Elimination)** For every closed term \(s\) there exists a basic term \(t\) such that \(s = t\).

**Proof** It is easy to show that for every basic term \(t'\) there exists a basic term \(t\) such that \(f(t') = t\), where \(f\) is one of the operators \(\partial_H, \nu_{rel}, \) or \(\nu_{rel}\). The proof is by induction on the structure of basic term \(t'\). Then, these encapsulation operators can be eliminated from every closed term by applying this result repeatedly to an innermost occurrence of one of the encapsulation operators until no such operator remains. It is important to realize that any subterm without encapsulation operators is a closed BPA\(\epsilon\)-term and hence is equal to a basic term using Theorem 12. This also applies to the term resulting after the last occurrence of an encapsulation operator has been replaced.

**Theorem 18 (Completeness)** The process algebra presented in this section is a complete axiomatisation of strong bisimilarity on closed terms.

**Proof** This theorem is proven using the meta-theory of [BV95]. Thereto, we first prove that the process algebra is an operationally conservative extension of the process algebra BPA\(\epsilon\). Using this, and some facts already proven, we obtain that it is also an equationally conservative extension. Completeness is then obtained by also using the elimination property.

First, we prove that the process algebra is an operationally conservative extension of the process algebra BPA\(\epsilon\). Both the term deduction systems of BPA\(\epsilon\) and the process algebra
are pure, well-founded, and in panth format. This is sufficient to establish that the process algebra is an operationally conservative extension of \( \text{BPA}_{\text{rt}} \).

Then, suppose that \( s \equiv t \) for some closed \( \text{BPA}_{\text{rt}} \)-terms \( s \) and \( t \). Since the axioms are sound we have \( s \equiv t \). By the fact that the process algebra is an operationally conservative extension we have \( \text{BPA}_{\text{rt}} \vdash s \equiv t \). Completeness of \( \text{BPA}_{\text{rt}} \) then gives \( \text{BPA}_{\text{rt}} \vdash s = t \). As the axioms of \( \text{BPA}_{\text{rt}} \) are contained in the axioms of the process algebra we have \( s = t \). Hence we have proven that the process algebra is an equationally conservative extension of \( \text{BPA}_{\text{rt}} \).

Finally, to prove the completeness theorem, suppose that \( s \equiv t \) for arbitrary closed terms \( s \) and \( t \). By the elimination property we have the existence of basic terms \( \hat{s} \) and \( t' \) such that \( s = \hat{s} \) and \( t = t' \). Since \( \hat{s} \) and \( t' \) are closed \( \text{BPA}_{\text{rt}} \)-terms, if \( \hat{s} = \hat{t} \) is an equivalence we also have \( s \equiv t' \). Hence, since \( \equiv \) is complete on closed \( \text{BPA}_{\text{rt}} \)-terms we obtain \( \text{BPA}_{\text{rt}} \vdash \hat{s} = \hat{t} \). Combined with \( s = \hat{s} \) and \( t = t' \) this gives \( s = t \).

There is an interesting relationship between the now operator and the time-out operators. \[ \text{Theorem 19} \quad \text{For } p \geq 0, r > 0 \text{ and closed terms } x \text{ we have } \]

1. \( \nu^p_{\text{rel}}(x) + x = x \);
2. \( \nu_{\text{rel}}(x) + \nu^r_{\text{rel}}(x) = \nu^r_{\text{rel}}(x) \);
3. \( \nu_{\text{rel}}(x) + \nu^0_{\text{rel}}(x) = \nu_{\text{rel}}(x) \);
4. \( x + \nu_{\text{rel}}(x) = x \).

\[ \text{Proof} \]

1. For \( p = 0 \) this is trivial: \( \nu^p_{\text{rel}}(x) + x = \nu^0_{\text{rel}}(x) + x = \hat{\delta} + x = x \). For \( p > 0 \) the theorem is proven by induction on the structure of basic term \( x \).

2. Easy, by induction on the structure of basic term \( x \).
   - \( x \equiv \hat{\epsilon} \). Trivial as \( \nu_{\text{rel}}(x) + \nu^r_{\text{rel}}(x) = \nu_{\text{rel}}(\hat{\epsilon}) + \nu^r_{\text{rel}}(\hat{\epsilon}) = \hat{\epsilon} + \hat{\epsilon} = \hat{\epsilon} = \nu^r_{\text{rel}}(\hat{\epsilon}) = \nu^r_{\text{rel}}(x) \).
   - \( x \equiv \hat{\delta} \). Trivial as \( \nu_{\text{rel}}(x) + \nu^r_{\text{rel}}(x) = \nu_{\text{rel}}(\hat{\delta}) + \nu^r_{\text{rel}}(x) = \hat{\delta} + \nu^r_{\text{rel}}(x) = \nu^r_{\text{rel}}(x) \).
   - \( x \equiv \hat{a} \cdot x' \) for some \( a \in A \) and basic term \( x' \). Then \( \nu_{\text{rel}}(x) + \nu^r_{\text{rel}}(x) = \nu_{\text{rel}}(\hat{a} \cdot x') + \nu^r_{\text{rel}}(\hat{a} \cdot x') = \hat{a} \cdot x' + \hat{a} \cdot x' = \nu_{\text{rel}}(\hat{a} \cdot x') + \nu^r_{\text{rel}}(\hat{a} \cdot x') = \nu^r_{\text{rel}}(x) \).
   - \( x \equiv \bar{\nu}^q \cdot x' \) for some \( q > 0 \) and basic term \( x' \). If \( r > q \) then \( \nu_{\text{rel}}(x) + \nu^r_{\text{rel}}(x) = \nu_{\text{rel}}(\bar{\nu}^q \cdot x') + \nu^r_{\text{rel}}(\bar{\nu}^q \cdot x') = \bar{\nu}^q \cdot \nu_{\text{rel}}(\bar{\nu}^q \cdot x') + \bar{\nu}^q \cdot \nu^r_{\text{rel}}(\bar{\nu}^q \cdot x') = (\bar{\delta} + \bar{\nu}^q) \cdot \nu^r_{\text{rel}}(\bar{\nu}^q \cdot x') = \bar{\nu}^q \cdot \nu_{\text{rel}}(\bar{\nu}^q \cdot x') \).

There is an interesting relationship between the now operator and the time-out operators.
• \( x \equiv x_1 + x_2 \) for some basic terms \( x_1, x_2 \). By induction we have \( \nu_{rel}(x_1) + \nu^r_{rel}(x_1) = \nu^f_{rel}(x_1) \) and \( \nu_{rel}(x_2) + \nu^r_{rel}(x_2) = \nu^f_{rel}(x_2) \). Hence \( \nu_{rel}(x) + \nu^r_{rel}(x) = \nu_{rel}(x_1 + x_2) + \nu^r_{rel}(x_1 + x_2) = \nu_{rel}(x_1) + \nu_{rel}(x_2) + \nu_{rel}(x_1) + \nu_{rel}(x_2) = \nu_{rel}(x_1 + x_2) = \nu^f_{rel}(x) \).

3. \( \nu_{rel}(x) + \nu^0_{rel}(x) = \nu_{rel}(x) + \delta = \nu_{rel}(x) \).

4. \( x = x + \nu^r_{rel}(x) = x + \nu^r_{rel}(x) + \nu_{rel}(x) = x + \nu_{rel}(x) \).

In the proof of the last item we conveniently used the first item.

3.2 Parallel composition operator

Now we have all machinery in place to define parallel composition. The resulting process algebra is called ACP\textsuperscript{art}. The parallel composition of two processes \( x \) and \( y \), notation \( x \parallel y \), describes the interleaving of their behaviour. An action can be executed by \( x \parallel y \) iff one of the operands can execute this action or this action is the result of the simultaneous execution of an action from \( x \) and an action from \( y \). This last possibility is usually called communication. The possible communications are specified by the communication function \( \gamma : A \times A \rightarrow A \). This function is a parameter of the process algebra and can be chosen dependent on the application. The function \( \gamma \) is partial, commutative and associative. We extend \( \gamma \) to a total function \( \gamma^* \) as follows: for \( a, b \in A \)

\[
\begin{align*}
\gamma^*(a, \delta) &= \delta, \\
\gamma^*(\delta, a) &= \delta, \\
\gamma^*(\delta, \delta) &= \delta, \\
\gamma^*(a, b) &= \gamma(a, b) \quad \text{if } \gamma(a, b) \text{ is defined}, \\
\gamma^*(a, b) &= \delta \quad \text{if } \gamma(a, b) \text{ is not defined}.
\end{align*}
\]

In the sequel we will write \( \gamma^* \) as \( \gamma \).

The process \( x \parallel y \) can let time pass if both operands can let time pass or if only one of them can let time pass and the other process has no other option than immediate termination.

Besides the parallel composition operator we also present the left-merge operator \( | \) and communication merge operator \( \cdot \). We present the operational rules in Tables 7 and 8. These auxiliary operators are needed in order to axiomatise parallel composition. The process \( x \parallel y \) behaves as the process \( x \parallel y \) except that the first action to be executed must be from process \( x \). The process \( x \cdot y \) also behaves as the process \( x \parallel y \) except that this time the first action must be a communication. This intuition of the operators explains the main axiom of parallel composition: \( x \parallel y = (x \parallel y + y \parallel x) + x \parallel y \).

Strong bisimilarity (Definition 2) is a congruence for the operators of the process algebra ACP\textsuperscript{art}.

**Definition 20** We extend the definition of \( | \cdot | \) on closed terms as follows: for closed terms \( x \) and \( y \)

\[
|x \parallel y| = |x| + |y| = |x| + |y|.
\]

19
<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \xrightarrow{a} x' \quad y \not\rightarrow$</td>
<td>$x \not\rightarrow y \xrightarrow{a} y'$</td>
<td>$x \xrightarrow{a} x' \quad y \uparrow \quad y \downarrow$</td>
</tr>
<tr>
<td>$x \parallel y \xrightarrow{a} x' \parallel y$</td>
<td>$x \parallel y \xrightarrow{a} x \parallel y'$</td>
<td>$x \parallel y \xrightarrow{a} x'$</td>
</tr>
<tr>
<td>$x \uparrow \quad x \downarrow$</td>
<td>$x \parallel y \xrightarrow{a} y'$</td>
<td>$x \parallel y \xrightarrow{a} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \parallel y \xrightarrow{a} y'$</td>
<td>$x \parallel y \xrightarrow{a} x' \parallel y'$</td>
<td>$x \not\rightarrow x' \quad y \not\rightarrow y'$</td>
</tr>
<tr>
<td>$x \xleftarrow{r} x' \quad y \downarrow \quad y \uparrow$</td>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
</tr>
<tr>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
</tr>
</tbody>
</table>

Table 8: Deduction rules for auxiliary operators for parallelism ($a, b, c \in A, r > 0$)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Rule</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \xrightarrow{a} x' \quad y \not\rightarrow$</td>
<td>$x \xrightarrow{a} x' \quad y \uparrow \quad y \downarrow$</td>
<td>$x \xrightarrow{a} x' \quad y \not\rightarrow y'$</td>
</tr>
<tr>
<td>$x \parallel y \xrightarrow{a} x' \parallel y$</td>
<td>$x \parallel y \xrightarrow{a} x'$</td>
<td>$x \parallel y \xrightarrow{a} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \xleftarrow{r} x' \quad y \not\rightarrow y'$</td>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x'$</td>
<td>$x \parallel y \xrightarrow{r} x' \parallel y'$</td>
</tr>
<tr>
<td>$x \downarrow \quad y \downarrow$</td>
<td>$x \uparrow \quad x \not\rightarrow$</td>
<td>$x \parallel y \uparrow \quad y \not\rightarrow$</td>
</tr>
<tr>
<td>$x \parallel y \downarrow$</td>
<td>$x \parallel y \uparrow$</td>
<td>$x \parallel y \uparrow$</td>
</tr>
<tr>
<td>$x \uparrow \quad y \uparrow$</td>
<td>$x \parallel y \uparrow$</td>
<td>$x \parallel y \uparrow$</td>
</tr>
<tr>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
</tr>
<tr>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
<td>$x \leq y \parallel z$</td>
</tr>
</tbody>
</table>
Theorem 21 (Congruence) Strong bisimilarity is a congruence for the operators of the process algebra $ACP^{Finit}_\epsilon$.

Proof The term deduction is still stratifiable and in panth format. Therefore congruence follows immediately.

In Table 9 we present axioms for the operators introduced in this section. As explained before parallel composition is expressed in terms of the auxiliary operators. Next, we will explain these.

The axioms for the communication merge operator are based on the structure of basic terms. Note that the two processes involved in the communication merge have to let time pass simultaneously (see the axiom on the bottom in the right-hand column of Table 9).

For the left-merge operator the situation is more complex. For the left operand we distinguish cases based on the structure of basic terms. For the case $a \cdot x$ we additionally make assumptions about the structure of the right operand. The following cases are distinguished: $\epsilon$, $\delta$, and $y + \delta$ for arbitrary $y$. In the first case the right operand has already terminated and hence the right operand is on its own. In the second case the process has deadlocked. In the third case the right argument has not yet terminated and hence the action $a$ can be executed. The following lemma indicates that every basic term can be written in one of the forms discussed above.

Lemma 22 (Representation) For every basic term $x$ we have $x = \epsilon$ or $x = \delta$ or $x = x + \delta$.

Proof Easy by induction on the structure of basic term $x$.

If the left operand is of the form $\sigma r \cdot x$, we must determine whether the right operand can also let time progress. Four cases can be distinguished: $\epsilon$, $\delta$, $\nu_{rel}(y) + \delta$ and $\nu_{rel}(y) + \sigma r \cdot z$. The first two cases are treated as before. The third case expresses that the right operand cannot let time pass and hence a deadlock remains. The last case expresses that the right operand can at least let $r$ time pass.

For understanding the axiomatisation of $\parallel$ it is necessary to realize that every closed term $x$ can be written as $\epsilon$, as $\nu_{rel}(x) + \delta$ or as $\nu_{rel}(x) + \sigma r \cdot x'$ for some $r > 0$ and closed term $x'$.

Theorem 23 (Representation) For basic term $x$ we have $x = \nu_{rel}(x)$ or there exists $r > 0$ such that there exists basic term $x'$ such that $x = \nu_{rel}(x) + \sigma r \cdot x'$ and $|x'| < |x|$.

Proof For a proof of this theorem we refer to Appendix B.

It is a routine exercise to check that these axioms are sound for our model of transition systems modulo strong bisimilarity.

Theorem 24 (Soundness) The process algebra $ACP^{Finit}_\epsilon$ is a sound axiomatisation of strong bisimilarity on closed $ACP^{Finit}_\epsilon$-terms.

Next, we will take some steps towards a result that states that the parallel composition operator (and all auxiliary operators) can be eliminated from closed terms. The specific form
Table 9: Axioms of $ACP_{\ddot{r}t}$ ($a, b, c \in A_{\ddot{r}}, p, q \geq 0, r > 0$)

\[
x \parallel y = (x \parallel y + y \parallel x) + x \parallel y
\]
\[
\dot{\delta} \parallel x = \dot{\delta}
\]
\[
x \parallel \delta = \delta
\]
\[
\dot{c} \parallel (x + y) = \dot{c} \parallel x + \dot{c} \parallel y
\]
\[
x \parallel \alpha \cdot x = \delta
\]
\[
\dot{\delta} \parallel \delta = \dot{\delta}
\]
\[
\dot{c} \parallel x = \dot{\delta}
\]
\[
x \parallel \dot{c} = \delta
\]
\[
\dot{\delta} \parallel \dot{c} = x
\]
\[
(x + y) \parallel z = x \parallel z + y \parallel z
\]
\[
\alpha \cdot x \parallel (y + \delta) = \alpha \cdot (x \parallel (y + \delta))
\]
\[
\delta^{\ddot{r}} \cdot x \parallel (v_{\ddot{r}}(y) + \delta) = \delta
\]
\[
\delta^{\ddot{p}} \cdot x \parallel (v_{\ddot{p}}(y) + \delta^{\ddot{p}} \cdot z) = \delta^{\ddot{p}} \cdot (x \parallel z)
\]

of the axioms for left merge require that each basic term can be written in a certain format. Below we present this result.

**Theorem 25 (Elimination)** For every closed $ACP_{\ddot{r}t}$-term $x$ there exists a basic term $y$ such that $ACP_{\ddot{r}t} x = y$.

**Proof** See Appendix C.

It is not hard to prove that the process algebra $ACP_{\ddot{r}t}$ is a conservative extension of the process algebra $BPA_{\ddot{r}t}$. Combining this result with the elimination theorem and the completeness of $BPA_{\ddot{r}t}$ gives that $ACP_{\ddot{r}t}$ is complete.

**Theorem 26 (Completeness)** The process algebra $ACP_{\ddot{r}t}$ is a complete axiomatisation of strong bisimilarity on closed $ACP_{\ddot{r}t}$-terms.

**Proof** The proof of completeness of $ACP_{\ddot{r}t}$ is similar to the proof of Theorem 18.

As a last part of this section we present some basic results about parallel composition that can be derived with the theory at hand.

**Theorem 27** For closed terms $x$ and $y$ we have

1. $\dot{c} \parallel x = x \parallel \dot{c} = x$
2. $\dot{\delta} \parallel x = x \parallel \dot{\delta} = \dot{\delta}$
3. $x \parallel y = y \parallel x$
4. $(x \parallel y) \parallel z = x \parallel (y \parallel z)$

22
Proof The first item is proven easily with induction on the structure of basic term \( x \).
The second item follows directly from the axioms as follows: \( \hat{\delta} \parallel x = \hat{\delta} \parallel x + x \parallel \hat{\delta} \parallel x = \hat{\delta} + \hat{\delta} + \hat{\delta} = \hat{\delta} \) and \( x \parallel \hat{\delta} = x \parallel \hat{\delta} + \hat{\delta} \parallel x + x \parallel \hat{\delta} = \hat{\delta} + \hat{\delta} + \hat{\delta} = \hat{\delta} \). For the third item we prove the propositions \( x \parallel y = y \parallel x \) and \( x \parallel y = y \parallel x \) simultaneously by induction on \( |x| + |y| \). For the first proposition we use in addition case distinction on the structure of the basic terms \( x \) and \( y \). The third item then follows easily: \( x \parallel y = x \parallel y \parallel x \parallel x \parallel y = y \parallel x \parallel y \parallel x = y \parallel x \).

For the proof of associativity of parallel composition we refer to Appendix D. This proof is long and tedious.

4 Relation with other basic process algebras

In this section, we consider the relation of the process algebra \( \text{BPA}_{\text{crt}} \) with other process algebras. As the means to compare different process algebras we consider the notions of conservative extensions and embeddings.

4.1 Comparing process algebras

A process algebra \( T' = (\Sigma', E') \) is a conservative extension of the process algebra \( T = (\Sigma, E) \) if (1) \( \Sigma \subseteq \Sigma' \) and (2) for all closed terms \( s \) and \( t \) over the signature \( \Sigma : (\Sigma, E) \vdash s = t \) if and only if \( (\Sigma', E') \vdash s = t \). This is an interesting notion of comparing process algebras as it says that the smaller process algebra is contained precisely in the larger process algebra. No identities are lost and none are added.

An explicit definition of an operator or constant \( f \) in the process algebra \( (\Sigma, E) \) is an equation \( f(x_1, \ldots, x_n) = t \) where \( t \) is a term over the signature \( \Sigma \) that does not contain other free variables than \( x_1, \ldots, x_n \). An extension of \( (\Sigma, E) \) with only explicitly defined operators and constants is called a definitional extension of \( (\Sigma, E) \). The following theorem states that a definitional extension is a special type of conservative extension.

Theorem 28 If the process algebra \( T' \) is a definitional extension of the process algebra \( T \), then \( T' \) is a conservative extension of \( T \).

An embedding of a process algebra \( T = (\Sigma, E) \) in a process algebra \( T' = (\Sigma', E') \) is a term structure preserving injective mapping \( h \) from the terms of \( T \) to the terms of \( T' \) such that for all closed terms \( s, t \) of \( T \), \( T \vdash s = t \) implies \( T' \vdash h(s) = h(t) \).

Theorem 29 (Baeten and Middelburg) Let \( T = (\Sigma, E) \) and \( T' = (\Sigma', E') \) be process algebras such that all operators from the signature \( \Sigma \) that are not in the signature \( \Sigma' \) can be defined in \( T' \) by explicit definitions. Let \( T'' = (\Sigma'', E'') \) be the resulting definitional extension of \( T' \). If the axioms of \( T \) are derivable for closed terms in \( T'' \), then \( T \) can be embedded in \( T' \) as follows: \( h(x) = x \), \( h(f(t_1, \ldots, t_n)) = f(h(t_1), \ldots, h(t_n)) \) if \( f \) in the signature of \( T' \), and \( h(f(t_1, \ldots, t_n)) = t[h(t_1), \ldots, h(t_n)/x_1, \ldots, x_n] \) if the explicit definition of \( f \) is \( f(x_1, \ldots, x_n) = t \).
4.2 Without empty process

First we compare the process algebra \( \text{BPA}^{\text{rst}} \) with the fragment of the process algebra \( \text{BPA}^{\text{rt}} \) from [BM99] where we omit the relative time-out and relative initialisation operators. The signature of this fragment consists of the urgent actions, undelayable deadlock, the deadlocked process, alternative and sequential composition, and the unary relative delay operator \( \sigma^p_{\text{rel}} \).

The axioms of this fragment of \( \text{BPA}^{\text{rst}} \) are given in Table 1.

The relative delay operator \( \sigma^p_{\text{rel}} \) of \( \text{BPA}^{\text{rst}} \) can be defined explicitly on \( \text{BPA}^{\text{rt}} \) by the equation

\[
\sigma^p_{\text{rel}}(x) = \delta^p \cdot x = \delta \cdot x = x,
\]

For closed terms over the signature of \( \text{BPA}^{\text{rst}} \), the axioms of \( \text{BPA}^{\text{rst}} \) are derivable from the axioms of \( \text{BPA}^{\text{rt}} \).

Then, as a consequence of Theorem 29, we have that \( \text{BPA}^{\text{rst}} \) can be embedded in \( \text{BPA}^{\text{rt}} \) using \( \text{h} \) defined by \( \text{h}(x) = x \), \( \text{h}(\delta) = \delta \), \( \text{h}(\delta \cdot x) = \delta \cdot \text{h}(x) \), \( \text{h}(x + y) = \text{h}(x) + \text{h}(y) \), and \( \text{h}(x \cdot y) = \text{h}(x) \cdot \text{h}(y) \).

4.3 Time-free

The process algebra \( \text{BPA}_{\delta \epsilon} \) consists of the special constants \( \delta \) and \( \epsilon \) which represent deadlock and termination respectively, the atomic actions \( a \in A \), and operators for sequential and alternative composition [BW90]. In this process algebra, where timing is only implicitly available in the interpretation of sequential composition, deadlock is a unit for alternative composition and a left-zero for sequential composition and termination is a unit for sequential composition. The axioms of \( \text{BPA}_{\delta \epsilon} \) are the axioms A1-A5 of BPA and in addition the following:

\[
\begin{align*}
A6 \quad & x + \delta = x, \\
A7 \quad & \delta \cdot x = \delta, \\
A8 \quad & x \cdot \epsilon = x, \\
A9 \quad & \epsilon \cdot x = x.
\end{align*}
\]

Although not treated in detail in this paper we we frequently discuss the axiom \( \partial_H(a) = \delta \) in cases where \( a \in H \) as it limits choices of embedding the time-free theory in the real-time theory.

With respect to the interpretation of the (time-free) atomic action \( a \) of the time-free theory \( \text{BPA}_{\delta \epsilon} \) in the standard real-time process algebra \( \text{BPA}^{\text{rst}} \) there are three possibilities:
Table 10: Deduction rules for initial delay constants \((r, r' > 0, p \geq 0)\)

\[
\begin{align*}
\delta(0, r) & \downarrow \delta(0, o) \uparrow \delta(0, r + r') \quad r \quad \delta(0, r') \quad \delta(p \leq \delta(0, p + r) \quad \delta(0, p) \leq \delta(0, p + r)
\end{align*}
\]

1. In the timed setting, the untimed action \(a\) takes place immediately and termination follows immediately.

2. In the timed setting, the process \(a\) executes the action \(a\) after an arbitrary delay and upon the execution of \(a\) the process \(a\) terminates immediately.

3. In the timed setting, the process \(a\) executes the action \(a\) after an arbitrary delay and upon the execution of \(a\) the process \(a\) terminates after an arbitrary delay.

In the first interpretation the atomic action \(a\) can be defined explicitly by the equation \(a = \delta\). This interpretation is not very interesting as only one moment in time is considered and hence one might as well stick with the time-free theory. For this reason, we will not consider embeddings based on this interpretation anymore.

For a thorough discussion of the other interpretations we need to extend the real-time process algebra \(BPA_{rt}\) with the time iteration constant \(\delta^*\). However, introducing the time iteration constant by itself does not suffice. In that case the time-out operator cannot be eliminated from closed terms anymore. For example, it is impossible to eliminate the time-out operator from the term \(v_{rel}(\delta^* \cdot \delta)\).

Therefore, we will introduce the initial delay constants \(\delta(0, p)\) for all \(p \in R^{\geq 0}\). The constant \(\delta(0, p)\) describes an arbitrary delay in the time interval \([0, p)\). The deduction rules for these constants are given in Table 10. The axioms of these constants are given in Table 11. If \(p = \infty\) we obtain the so-called time iteration constant also denoted as \(\delta^*\). With these initial delay constants it is possible to eliminate the time-out operator as can be seen from the additional axioms for the time-out operator in Table 11.

Next, we discuss the embedding where the atomic action \(a\) is interpreted as the timed process \(\delta^* \cdot a \cdot \delta^*\). With the intuition described by the equation \(\partial_H(a) = \delta\) where \(a \in H\) from \(BPA_{\delta^*}\), we need to interpret \(\delta\) as the process \(\delta^* \cdot \delta\). Then, axiom A6 forces us to interpret \(\epsilon\) as the process \(\delta^* \cdot (\epsilon + \delta)\), i.e., as the process \(\delta^* \cdot \delta\).

We can easily derive the axioms of \(BPA_{\delta^*}\) for closed terms as follows. The axioms A1-A5 are also part of \(BPA_{\delta^*}\) and need therefore not be considered. Axiom A7 is easily derived as follows:

\[
\delta \cdot x = (\delta^* \cdot \delta) \cdot x = \delta^* \cdot (\delta \cdot x) = \delta^* \cdot \delta = \delta.
\]

For the axioms A8 and A9 we use induction on the structure of closed \(BPA_{\delta^*}\)-terms and the fact that \(\epsilon \cdot x = \delta^* \cdot x + \delta\). We present the two proofs simultaneously:
Table 11: Axioms of initial delay constants \((p, q \geq 0, r, r' > 0)\)

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\overline{\sigma}^{[0,0]} = \delta)</td>
<td></td>
</tr>
<tr>
<td>(\partial_H(\overline{\sigma}^{[0,p]}, x) = \overline{\sigma}^{[0,p]}, \partial_H(x))</td>
<td></td>
</tr>
<tr>
<td>(\overline{\sigma}^{[0,p+q]} + \overline{\sigma}^{[0,p]} = \overline{\sigma}^{[0,p+q]})</td>
<td></td>
</tr>
<tr>
<td>(\overline{\sigma}^{[0,p]} \cdot x + \overline{\sigma}^{[0,p]}, y = \overline{\sigma}^{[0,p]} \cdot (x + y))</td>
<td></td>
</tr>
<tr>
<td>(\overline{\sigma}^{[0,r]}, \overline{\sigma}^{[0,r']} = \overline{\sigma}^{[0,r+r']})</td>
<td></td>
</tr>
<tr>
<td>(\overline{\sigma}^{[0,p]} \cdot \overline{\sigma}^{[0,p+r]} = \overline{\sigma}^{[0,p]} \cdot \overline{\sigma}^{[0,p+r]})</td>
<td></td>
</tr>
<tr>
<td>(\nu_{\text{rel}}(\overline{\sigma}^{[0,r]}, \nu_{\text{rel}}(x)) = \overline{\sigma}^{[0,r]} \cdot \nu_{\text{rel}}(x))</td>
<td></td>
</tr>
<tr>
<td>(\nu_{\text{rel}}(\overline{\sigma}^{[0,0+r]}, x) = \overline{\sigma}^{[0,0]} \cdot \delta)</td>
<td></td>
</tr>
<tr>
<td>(\nu_{\text{rel}}(\overline{\sigma}^{[0,0]}, x) = \nu_{\text{rel}}(x))</td>
<td></td>
</tr>
</tbody>
</table>

- \(x \equiv \epsilon\). Then \(x \cdot \epsilon = \epsilon \cdot x = \epsilon \cdot \epsilon = \overline{\sigma}^{[0]} \cdot \epsilon + \delta = \overline{\sigma}^{[0]} \cdot \overline{\sigma}^{[0]} \cdot (\epsilon + \delta) + \delta = \overline{\sigma}^{[0]} \cdot (\epsilon + \delta) + \overline{\sigma}^{[0]} \cdot \delta = \overline{\sigma}^{[0]} \cdot (\epsilon + \delta) = \epsilon = x\).

- \(x \equiv \delta\). Then \(x \cdot \epsilon = \delta \cdot \epsilon = \overline{\sigma}^{[0]} \cdot \delta \cdot \epsilon = \overline{\sigma}^{[0]} \cdot \overline{\sigma}^{[0]} \cdot \delta = \delta \) and \(\epsilon \cdot x = \epsilon \cdot \delta = \overline{\sigma}^{[0]} \cdot \delta + \delta = \overline{\sigma}^{[0]} \cdot \delta + \delta + \delta = \delta = \delta = \delta = \delta = \delta \).

- \(x \equiv x_1 \cdot x_2\). By induction we have \(\overline{\sigma} \cdot x_1 = x_1\) and \(x_2 \cdot \epsilon = x_2\). Then \(x \cdot \epsilon = x_1 \cdot x_2 \cdot \epsilon = x_1 \cdot x_2 = x\) and \(\epsilon \cdot x = \epsilon \cdot x_1 \cdot x_2 = x_1 \cdot x_2 = x\).

- \(x \equiv x_1 + x_2\). By induction we have \(\overline{\sigma} \cdot x_1 = x_1, \epsilon \cdot x_2 = x_2, x_1 \cdot \epsilon = x_1, \) and \(x_2 \cdot \epsilon = x_2\). Then \(x \cdot \epsilon = (x_1 + x_2) \cdot \epsilon = x_1 \cdot \epsilon + x_2 \cdot \epsilon = x_1 + x_2 = x\) and \(\epsilon \cdot x = \epsilon \cdot (x_1 + x_2) = \overline{\sigma}^{[0]} \cdot (x_1 + x_2) = \overline{\sigma}^{[0]} \cdot x_1 + \overline{\sigma}^{[0]} \cdot x_2 = \epsilon \cdot x_1 + \epsilon \cdot x_2 = x_1 + x_2 = x\).

Using the above, we easily obtain derivability of axiom A6 for closed BPA_δ¯ε-terms:

\[x + \delta = \epsilon \cdot x + \delta = \overline{\sigma}^{[0]} \cdot x + \delta + \delta = \overline{\sigma}^{[0]} \cdot x + \delta = \epsilon \cdot x = x\]

Finally, we consider the interpretation where the atomic action \(a\) is interpreted as the timed process \(\overline{\sigma}^{[0]} \cdot \delta\). Again, we are forced to adopt \(\overline{\sigma}^{[0]} \cdot \delta\) as the timed interpretation of \(\delta\) and \(\overline{\sigma}^{[0]} \cdot \bar{\epsilon}\) as the timed interpretation of \(\epsilon\). This embedding, however, is incompatible with the law \(x \cdot \epsilon = x\). Recently, Baeten [Bae00] has published a redesign of the basic process algebras BPA_δ and BPA_δε, that facilitate this embedding. This redesign is based on the separation of action execution and termination. Instead of actions as constants, an action prefix operator is introduced for each action.

**5 Conclusion**

Using our theory, we can replace the delay operators of timed ACP by delay constants, thus simplifying axioms and calculations. We have a clear separation of action execution and termination in our operational rules. All states in transition systems correspond to process terms. In order to ensure that the axiom of time-determinism holds in our operational model, we use the technique of saturation of transition systems. This technique was used for silent
steps in [BK88, Gla87, BBR00]; here, we use saturation of time steps. This avoids the use of bisimulations relating sets of states and allows to use the results that are based on the format of the SOS rules. This is an improvement in the treatment of time steps.

On the basis of the material presented here, we can extend also the other timed process algebras of the framework of [BM99] with explicit termination. Thus, first we define a variant of the present theory using absolute timing instead of relative timing. Both absolute and relative timing can be integrated using parametric timing ([BB96, BB97]). Then, absolute-time discrete-time process algebra is a subtheory of absolute-time dense-time process algebra as in [BM99]. Finally, relative-time discrete-time process algebra and parametric time discrete-time process algebra can be developed.

The combination of the empty process and discrete, relative time process algebra has been studied in [Ver97, BV97]. Among the conclusions in [Ver97, BV97] are the following two: the empty process cannot be straightforwardly combined with the deadlocked process of [BB96, BB97, BBR00], and the behaviour of the empty process is not always into accordance with one's first intuition. In this paper we present a combination of the empty process with standard real-time process algebra in relative timing with the deadlocked process in which the behaviour of the empty process is clear in all cases.

References


A Proof of completeness of BPA^{str}

In this appendix we prove that BPA^{str} is a complete axiomatisation of strong bisimilarity on closed BPA^{str}-terms. First, we give a lemma that relates the operational semantics to the equational derivability.

Lemma 30

1. if \( x \downarrow \), then \( x = \varepsilon + x \);
2. if \( x \Uparrow \), then \( x = x + \overline{a} \);
3. if \( x \stackrel{a}{\rightarrow} x' \), then \( x = \overline{a} \cdot x' + x \);
4. if \( x \preceq y \), then \( x + y = y \);
5. if \( x \stackrel{r}{\rightarrow} x' \), then \( x = \overline{r} \cdot x' + x \).
Proof

1. By induction on the structure of closed term \( x \).

2. By induction on the structure of closed term \( x \), using the first item.

3. By induction on the structure of closed term \( x \), using the first item.

4. By induction on the depth of the proof of \( x \leq y \), using the first two items. The last step in the proof is an application of deduction rule:

   [23] Obviously we have \( x + x = x \).

   [24] Then \( x \equiv \delta \). We obviously have \( x + y = \delta + y = y \).

   [25] Then \( x \equiv \delta \). From \( y \gamma \) and the second item we have \( y = y + \delta \). Then \( x + y = \delta + y = y \).

   [26] Then \( x \equiv \epsilon \). By the first item and \( y \downarrow \) we have \( y = \epsilon + y \). Hence \( x + y = \epsilon + y = y \).

   [27] Then \( y = x' + y' \) and \( x \leq x' \). By induction we then have \( x + x' = x' \). Then \( x + y = x + x' + y' = x' + y' = y \).

   [28] Similar to the previous case.

   [29] Then \( x \equiv x' + x'' \) and \( y \equiv y' + y'' \) for some \( x', x'', y', y'' \) such that \( x' \leq y' \) and \( x'' \leq y'' \). By induction we then have \( x' + y' = y' \) and \( x'' + y'' = y'' \). Then \( x + y = x' + x'' + y' + y'' = x' + y' + x'' + y'' = y' + y'' = y \).

   [30] Then \( x \equiv x' \cdot z \) and \( y \equiv y' \cdot z \) for some \( x', y', z \) such that \( x' \leq y' \). By induction we then have \( x' + y' = y' \). Thus, \( x + y = x' \cdot z + y' \cdot z = (x' + y') \cdot z = y' \cdot z = y \).

   [31] Then \( y \equiv y' \cdot z \) for some \( y', z \) such that \( y' \downarrow \) and \( x \leq z \). By the first item we have \( y' + \epsilon = y' \). By induction we have \( x + z = z \). Thus, \( x + y = x + y' \cdot z = x + (\epsilon + y') \cdot z = x + (\epsilon + z + y' \cdot z) = z + y' \cdot z = z + y' \cdot z \).

   [32] Then \( x \equiv \overline{\delta} \cdot x' \) and \( y \equiv \overline{\delta} \cdot y' \) for some \( x', y' \) such that \( x' \leq y' \). By induction we have \( x' + y' = y' \). Thus, \( x + y = \overline{\delta} \cdot x' + \overline{\delta} \cdot y' = \overline{\delta} \cdot (x' + y') = \overline{\delta} \cdot y' = y \).

5. By induction on the depth of the proof of \( x \mapsto x' \) using the first and fourth item. The last step in the proof is an application of deduction rule:

   [7] Then we must have \( x \equiv \overline{\delta}^{p+r} \) and \( x' \equiv \overline{\delta}^p \) for some \( p, r \in \mathbb{R}^\geq 0 \) and \( r > 0 \). Then \( \overline{\delta} \cdot x' + x = \overline{\delta} \cdot \overline{\delta}^{p-r} + \overline{\delta}^p = \overline{\delta}^p + \overline{\delta}^p = \overline{\delta}^p = x \).

   [13] Then \( x \equiv x_1 + x_2 \) for some \( x_1, x_2 \) such that \( x_1 \mapsto x' \). By induction we then have \( \overline{\delta} \cdot x' + x_1 = x_1 \). Then \( \overline{\delta} \cdot x' + x = \overline{\delta} \cdot x'_1 + x_1 + x_2 = x_1 + x_2 = x \).

   [14] Similar to the previous case.

   [20] Then \( x \equiv x_1 \cdot x_2 \) and \( x_1 \mapsto x'_1 \) for some \( x_1, x_2, x'_1 \) such that \( x' \equiv x'_1 \cdot x_2 \). By induction we have \( \overline{\delta} \cdot x'_1 + x_1 = x_1 \). Then \( \overline{\delta} \cdot x' + x = \overline{\delta} \cdot x'_1 + x_2 + x_1 \cdot x_2 = (\overline{\delta} \cdot x'_1 + x_1) \cdot x_2 = x_1 \cdot x_2 = x \).
Then \( x = x_1 \cdot x_2 \) for some \( x_1, x_2 \) such that \( x_1 \downarrow \) and \( x_2 \downarrow \). By the first item and \( x_1 \downarrow \) we have \( x_1 = \varepsilon + x_1 \). By induction we have \( \varepsilon_r \cdot x' + x_2 = x_2 \). Then

\[
\varepsilon_r \cdot x' + x = \varepsilon_r \cdot x' + x_1 \cdot x_2 \\
= \varepsilon_r \cdot x' + (\varepsilon + x_1) \cdot x_2 \\
= \varepsilon_r \cdot x' + \varepsilon \cdot x_2 + x_1 \cdot x_2 \\
= \varepsilon_r \cdot x' + x_2 + x_1 \cdot x_2 \\
= x_2 + x_1 \cdot x_2 \\
= \varepsilon \cdot x_2 + x_1 \cdot x_2 \\
= (\varepsilon + x_1) \cdot x_2 \\
= x_1 \cdot x_2 \\
= x.
\]

Then \( x = x_1 \cdot x_2, x_1 \vdash x_1', x_2' \downarrow \), and \( x_2 \vdash x' \) for some \( x_1, x_2, x_1', x_2' \) and \( r_1, r_2 \) such that \( r = r_1 + r_2 \). By the first item we have \( x_1' = \varepsilon + x_1' \). By induction we have \( \varepsilon_{r_1} \cdot x_1' + x_1 = x_1 \) and \( \varepsilon_{r_2} \cdot x' + x_2 = x_2 \). Then

\[
\varepsilon_{r_1} \cdot x_1' + x = \varepsilon_{r_1} \cdot x_1' + x_1 \cdot x_2 \\
= \varepsilon_{r_1} \cdot x_1' + (\varepsilon_{r_1} \cdot x_1' + x_1) \cdot x_2 \\
= \varepsilon_{r_1} \cdot x_1' + \varepsilon_{r_1} \cdot (x_1 \cdot x_2 + x_1 \cdot x_2) \\
= \varepsilon_{r_1} \cdot (x_1 \cdot x_2 + x_1 \cdot x_2) + x_1 \cdot x_2 \\
= \varepsilon_{r_1} \cdot x_1 \cdot x_2 + x_1 \cdot x_2 \\
= (\varepsilon_{r_1} \cdot x_1 + x_1) \cdot x_2 \\
= x_1 \cdot x_2 \\
= x.
\]

Then \( x \vdash x_1 \) and \( x \vdash x_2 \) for some \( x_1, x_2 \) such that \( x' = x_1 + x_2 \). By induction we have \( \varepsilon_r \cdot x_1 + x = x \) and \( \varepsilon_r \cdot x_2 + x = x. Then

\[
\varepsilon_r \cdot x_1 + x = \varepsilon_r \cdot (x_1 + x_2) + x \\
= \varepsilon_r \cdot x_1 + \varepsilon_r \cdot x_2 + x \\
= \varepsilon_r \cdot x_1 + x \\
= x.
\]

Then \( x \vdash x'' \) for some \( x'' \) such that \( x' \leq x'' \). By the fourth item and \( x' \leq x'' \) we have \( x'' + x' = x'' \). By induction we have \( \varepsilon_r \cdot x'' + x = x \). Then

\[
\varepsilon_r \cdot x' + x = \varepsilon_r \cdot x' + \varepsilon_r \cdot x'' + x \\
= \varepsilon_r \cdot (x' + x'') + x \\
= \varepsilon_r \cdot x'' + x \\
= x.
\]
Now we turn to the completeness of BPA\textsuperscript{st} with respect to strong bisimilarity on closed BPA\textsuperscript{st}-terms. We only have to prove that \( x + y \equiv y \) implies \( x + y = y \).

Suppose \( x + y \equiv y \). By induction on the structure of basic term \( x \).

- \( x \equiv \hat{a} \). Obviously \( x + y = \hat{a} + y = y \).
- \( x \equiv \hat{e} \). Hence \( x + y \downarrow \). Thus we obtain \( y \downarrow \). By Lemma 30.1 this gives \( y = \hat{e} + y \). Clearly \( x + y = \hat{e} + y = y \).
- \( x \equiv \bar{a} \cdot x' \).
  - \( a \equiv \hat{a} \). Then \( x \not\equiv y \). Therefore \( x + y \not\equiv y \). Thus \( y \not\equiv y \). By Lemma 30.2 we thus have \( y = y + \hat{a} \). We find \( x + y = \hat{a} \cdot x' + y = \hat{a} + y = y \).
  - \( a \in A \). Then \( x \not\equiv \hat{e} \cdot x' \). Hence \( x + y \not\equiv \hat{e} \cdot x' \). Therefore \( y \not\equiv y' \) for some \( y' \) such that \( \hat{e} \cdot x' \equiv y' \). Obviously \( x' \equiv y' \). By induction we have \( x' = y' \). By Lemma 30.3 we have \( y = \bar{a} \cdot y' + y \). Then \( x + y = \bar{a} \cdot x' + y = \bar{a} \cdot y' + y = y \).
- \( x \equiv \bar{a} \cdot x' \). Then \( x \not\equiv \hat{a} \cdot x' \). Hence \( x + y \not\equiv \hat{a} \cdot x' \) and therefore \( y \not\equiv y' \) for some \( y' \) such that \( \hat{a} \cdot x' \equiv y' \). Then also \( x' \equiv y' \). By induction we then have \( x' = y' \). By Lemma 30.5 we have \( y = \bar{a} \cdot y' + y \). Then \( x + y = \bar{a} \cdot x' + y = \bar{a} \cdot y' + y = y \).
- \( x \equiv x' + x'' \). Then necessarily \( x' + y \equiv y \) and \( x'' + y \equiv y \). By induction we thus have \( x' + y = y \) and \( x'' + y = y \). Therefore, \( x + y = x' + x'' + y = x' + y = y \).

**B Proof of representation theorem**

The proof of this theorem requires that we strengthen the proposition to be proven. Hence, we consider the following proposition: For basic term \( x \) we have \( x = v_{rel}(x) \) or there exists \( p > 0 \) such that for all \( q: 0 < q \leq p \) there exists basic term \( x' \) such that \( x = v_{rel}^q(x) + \bar{a} \cdot x' \) and \(|x'| < |x|\).

This proposition is proven by induction on the structure of basic term \( x \). The first three cases \((x = \hat{e}, x = \hat{a}, x = \bar{a} \cdot x')\) are trivial as in these cases \( x = v_{rel}(x) \).

The case \( x = \bar{a} \cdot x' \) is also not difficult. Let \( 0 < r' \leq r \). Then \( x = x + v_{rel}'(x) = \bar{a} \cdot x' + v_{rel}'(x) = \bar{a} \cdot (\bar{a}^{r-r'} \cdot x') + v_{rel}'(x) \). Note that \(|\bar{a}^{r-r'} \cdot x'| = (r-r'+1) \times |x'| + 1 < (r+1) \times |x'| + 1 = |\bar{a} \cdot x'| = |x|\).

Finally, we treat the case \( x = x_1 + x_2 \) in more detail. By induction, four cases can be distinguished:

1. \( x_1 = v_{rel}(x_1) \) and \( x_2 = v_{rel}(x_2) \). Trivial.
2. \( x_1 = v_{rel}(x_1) \) and there exists \( r_2 > 0 \) such that for all \( 0 < r'_2 \leq r_2 \) there exists \( x'_2 \) with \( x_2 = v_{rel}^{r_2}(x_2) + \bar{a}^{r'_2} \cdot x'_2 \) and \(|x'_2| < |x_2| \). Note that \( x_1 = x_1 + v_{rel}^{r_2}(x_1) = \bar{a} \cdot x'_2 \).
3. there exists $r_1 > 0$ such that for all $0 < r'_1 \leq r_1$ there exists $x'_1$ with $x_1 = v_{rel}^r(x_1) + \sigma^{r_1'}, x'_1$ and $|x'_1| < |x_1|$ and $x_2 = v_{rel}(x_2)$. Similar to the previous case.

4. there exist $r_1 > 0$ and $r_2 > 0$ such that for all $0 < r'_1 \leq r_1$ and $0 < r'_2 \leq r_2$ there exist $x'_1$ and $x'_2$ with $x_1 = v_{rel}^r(x_1) + \sigma^{r_1'}, x'_1$ and $x_2 = v_{rel}(x_2) + \sigma^{r_2}, x'_2$. Without loss of generality we may assume that $r_1 \leq r_2$. By the induction hypothesis we find that there exists $x'_2$ such that $x = v_{rel}^r(x_2) + \sigma^{r_2} x'_2$ and $|x'_2| < |x_2|$. Then $x_1 + x_2 = v_{rel}^r(x_1) + \sigma^{r_1'} x'_1 + v_{rel}(x_2) + \sigma^{r_2} (x'_2 + \sigma^{r_2} x'_2)$. From $|x'_1| < |x_1|$ and $|x'_2| < |x_2|$ we find $|x_1 + x_2| < |x_1 + x_2|_1$.

C Proof of elimination theorem for ACP$^{\text{eq}}$

We prove the following propositions simultaneously for all basic terms $x$ and $y$ with induction on $|x| + |y|$: 

1. there exists a basic term $z$ such that $x \parallel y = z$;

2. there exists a basic term $z$ such that $x \parallel y = z$;

3. there exists a basic term $z$ such that $x \parallel y = z$.

The first proposition is the most complex of the three. We start by distinguishing cases for $x$ according to the definition of basic terms. For the case $x = \bar{c} \cdot x'$ we proceed by distinguishing cases for $y$ according to the definition of basic terms as well. For the case $x = \bar{c} \cdot x'$ the elimination of $\parallel$ is trivial: $\delta \parallel y = \delta$. For the case $x = \bar{c} \cdot x'$ we distinguish the cases $y = \bar{c} \cdot x'$ and $y = \bar{c} \cdot x'$. In case $y = \bar{c} \cdot x'$ we easily find that a basic term results. For $y = \bar{c} \cdot x'$ we find $x \parallel y = \bar{c} \cdot x' \parallel (y + \bar{c}) = \bar{c} \cdot x' \parallel (y + \bar{c}) = \bar{c} \cdot x' \parallel y$. By induction we have the existence of a basic term $z$ such that $x \parallel y = z$. Then $\bar{a} \cdot (x \parallel y) = \bar{a} \cdot z$ which is also a basic term. For the case $x = \bar{c} \cdot x'$ we distinguish four cases for $y$ according to the representation theorem and representation lemma above.

1. $y = \bar{c}$. Then $x \parallel y = x \parallel \bar{c} = x$.

2. $y = \bar{c}$. Then $x \parallel y = x \parallel \bar{c} = \bar{c}$.

3. $y = \nu_{rel}(y) + \bar{c}$. Then $x \parallel y = \bar{c} \cdot x' \parallel (\nu_{rel}(y) + \bar{c}) = \bar{c} \cdot \bar{c} \cdot \bar{c} = \bar{c} \cdot \bar{c}$. 

4. $y = \nu_{rel}(y) + \sigma^r \cdot y'$ for some $r' > 0$ and basic term $y'$ such that $|y'| < |y|$. If $r' \geq r$ then $y = \nu_{rel}(y) + \sigma^r \cdot y'$ and $|y'| < |y|$. Then $x \parallel y = \sigma^r \cdot x' \parallel (\nu_{rel}(y) + \sigma^r \cdot y') = \sigma^r \cdot (x' \parallel y')$. By induction we have the existence of basic term $z$ such that $x' \parallel y' = z$. Hence $x \parallel y = \sigma^r \cdot z$, which is a basic term. If $r' < r$, then $x \parallel y = \sigma^r \cdot x' \parallel (\nu_{rel}(y) + \sigma^r \cdot y') = \sigma^r \cdot \sigma^{r'-r'}$. 

33
\( x' \parallel (\nu_{\text{rel}}(y) + \tilde{\sigma}^{r'} \cdot y') = \tilde{\sigma}^{r'} \cdot (\tilde{\sigma}^{r''} \cdot x' \parallel y'). \) As \( |\tilde{\sigma}^{r''} \cdot x' \parallel y'| = |\tilde{\sigma}^{r'} \cdot x'| + |y'| = (r - r' + 1) \times |x'| + 1 + |y'| < (r + 1) \times |x'| + 1 + |y| = |\tilde{\sigma}^{r'} \cdot x'| + |y| = |\tilde{\sigma}^{r'} \cdot x' \parallel y| = |x \parallel y|, \)
by induction we have the existence of basic term \( z \) such that \( \tilde{\sigma}^{r-r'} \cdot x' \parallel y' = z \). Hence \( x \parallel y = \tilde{\sigma}^{r'} \cdot z \), which is a basic term.

For the case \( x = x_1 + x_2 \) we apply the induction hypothesis and the distributivity law of left merge: \( x \parallel y = (x_1 + x_2) \parallel y = x_1 \parallel y + x_2 \parallel y = z_1 + z_2. \)

For the second proposition we distinguish cases for both \( x \) and \( y \) according to the definition of basic terms. The proof is easy. The third proposition follows immediately from the first and second proposition and the fact that the alternative composition of basic terms gives a basic term again.

**D  Associativity of parallel composition**

In this appendix we prove that parallel composition is associative. Thereto, we prove the following equations simultaneously with induction on \( |x| + |y| + |z|: \)

\[
\begin{align*}
(x \parallel y) \parallel z &= x \parallel (y \parallel z), \\
(x \parallel y) \parallel z &= x \parallel (y \parallel z), \\
(x \parallel y) \parallel z &= x \parallel (y \parallel z), \\
(x \parallel y) \parallel z &= x \parallel (y \parallel z).
\end{align*}
\]

These equations are in ACP-literature usually referred to as the **Axioms of Standard Concurrency**. Although we prove the equations simultaneously we present their proofs separately to enhance readability.

Assume that

\[
x = \sum_{i \in I} \tilde{a}_i \cdot x_i + \tilde{\sigma}^u \cdot x' + \sum_{j \in J} \tilde{e}
\]

and

\[
y = \sum_{k \in K} \tilde{b}_k \cdot y_k + \tilde{\sigma}^v \cdot y' + \sum_{l \in L} \tilde{e}
\]

and

\[
z = \sum_{m \in M} \tilde{c}_m \cdot z_m + \tilde{\sigma}^w \cdot z' + \sum_{n \in N} \tilde{e}
\]

for some \( a_i, b_k, c_m \in A_\delta \) and basic terms \( x_i, x', y_k, y', z_m, z' \) such that \( u = 0 \) iff \( x' = \delta \), \( v = 0 \) iff \( y' = \delta \), \( w = 0 \) iff \( z' = \delta \).

- \( (x \parallel y) \parallel z = x \parallel (y \parallel z) \). If \( y = \delta \) or \( z = \delta \) or \( y = \varepsilon \) or \( z = \varepsilon \), the proposition is trivially proven. Hence, from now on we may assume that \( y = y + \delta \) and \( z = z + \delta \).

Now, we will distinguish for \( x \) the basic term cases. The cases \( x = \varepsilon \), \( x = \delta \), \( x = \tilde{a} \cdot x' \), and \( x = x' + x'' \) are trivial and therefore omitted. What remains is the case \( x = \tilde{\sigma}^u \cdot x' \).
(with \( u > 0 \)). Let \( uvw = \min(u, v, w) \). If \( uvw = 0 \), then necessarily \( v = 0 \) or \( w = 0 \). Therefore \( y' = \hat{\delta} \) or \( z' = \hat{\delta} \). We obtain \( \langle \hat{\sigma}^u \cdot x', y \rangle \parallel z = \hat{\delta} \) and \( \hat{\sigma}^u \cdot x' \parallel (y \parallel z) = \hat{\delta} \).

Now suppose that \( uvw > 0 \). Observe that

\[
y \parallel z = \sum_{k \in K} b_k \cdot (yk \parallel z) + \sum_{m \in M} c_m \cdot (zm \parallel y)
\]

\[
+ \sum_{k \in K} \sum_{m \in M} \gamma(b_k, c_m) \cdot (yk \parallel zm)
\]

\[
+ \hat{\sigma}^{uvw} \cdot (\hat{\sigma}^{u-uvw} \cdot y' \parallel \hat{\sigma}^{v-uvw} \cdot z') + \sum_{n \in N} y + \sum_{l \in L} z
\]

Then

\[
\hat{\sigma}^u \cdot x' \parallel (y \parallel z) = \hat{\sigma}^{uvw} \cdot (\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y' \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

and

\[
(\hat{\sigma}^u \cdot x' \parallel y) \parallel z = \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

By induction we have

\[
\hat{\sigma}^{u-uvw} \cdot x' \parallel (\hat{\sigma}^{v-uvw} \cdot y' \parallel \hat{\sigma}^{w-uvw} \cdot z') = \hat{\sigma}^{u-uvw} \cdot x' \parallel (\hat{\sigma}^{v-uvw} \cdot y' \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

\[
y') \parallel (\hat{\sigma}^{u-uvw} \cdot y' \parallel \hat{\sigma}^{w-uvw} \cdot z') \quad \text{since} \quad |\hat{\sigma}^{u-uvw} \cdot x'| < |z|, |\hat{\sigma}^{u-uvw} \cdot y'| < |y|, \text{and} \quad |\hat{\sigma}^{u-uvw} \cdot z'| < |z|.
\]

**•** \((x \parallel y) \parallel z = x \parallel (y \parallel z)\). Let \( uvw = \min(u, v, w) \). If \( uvw > 0 \)

\[
(x \parallel y) \parallel z = \sum_{i \in I} \sum_{k \in K} \sum_{m \in M} \gamma(a_{b_k}, c_m) \cdot ((x_i \parallel y_k) \parallel z_m)
\]

\[
+ \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

and

\[
x \parallel (y \parallel z) = \sum_{i \in I} \sum_{k \in K} \sum_{m \in M} \gamma(a_{b_k}, c_m) \cdot ((x_i \parallel y_k) \parallel z_m)
\]

\[
+ \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

By the induction hypothesis we have \((x' \parallel y') \parallel z' = x' \parallel (y' \parallel z')\). By the associativity of the communication function we have \(\gamma(y(a_{b_k}, c_m)) = \gamma(a_{b_k}, c_m)\). By the induction hypothesis we have

\[
\hat{\sigma}^{u-uvw} \cdot x' \parallel (\hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z' = \hat{\sigma}^{u-uvw} \cdot x' \parallel (\hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z'.
\]

If \( uvw = 0 \) then we have similar results without the alternatives headed by the delay constants.

**•** \((x \parallel y)\parallel z = x \parallel (y \parallel z)\). Let \( uvw = \min(u, v, w), uv = \min(u, v), \text{and} \ vw = \min(v, w)\).

\[
(x \parallel y)\parallel z = (\sum_{i \in I} \sum_{k \in K} \gamma(a_{b_k}, c_m) \cdot ((x_i \parallel y_k) \parallel z_m)
\]

\[
+ \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

\[
= \sum_{i \in I} \sum_{k \in K} \gamma(a_{b_k}) \cdot ((x_i \parallel y_k) \parallel z_m)
\]

\[
+ \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]

\[
= \sum_{i \in I} \sum_{k \in K} \gamma(a_{b_k}) \cdot ((x_i \parallel y_k) \parallel z_m)
\]

\[
+ \hat{\sigma}^{uvw} \cdot ((\hat{\sigma}^{u-uvw} \cdot x' \parallel \hat{\sigma}^{v-uvw} \cdot y') \parallel \hat{\sigma}^{w-uvw} \cdot z')
\]
\[ x \parallel (y \parallel z) = x \parallel (\sum_{k \in K} b_k \cdot (y_k \parallel z)) \]
\[ + x \parallel (\sum_{i \in I} \sum_{n \in N} \hat{c}) \]
\[ = \sum_{i \in I} \sum_{k \in K} \gamma(a_i, b_k) \cdot (x_i \parallel (y_k \parallel z)) \]
\[ + \sum_{i \in I} \sum_{k \in K} \gamma(a_i, b_k) \cdot (x_i \parallel (y_k \parallel z)) \]
\[ + \sum_{i \in I} \sum_{k \in K} \gamma(a_i, b_k) \cdot (x_i \parallel (y_k \parallel z)) \]
\[ + \sum_{i \in I} \sum_{k \in K} \gamma(a_i, b_k) \cdot (x_i \parallel (y_k \parallel z)) \]

Applying the induction hypotheses gives the identity \((x \parallel y) \parallel z = x \parallel (y \parallel z)\).

* \((x \parallel y) \parallel z = x \parallel (y \parallel z)\). Trivial using the previous three propositions.
### Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

If you want to receive reports, send an email to: m.m.j.philips@tue.nl (we cannot guarantee the availability of the requested reports)

**In this series appeared:**

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>96/01</td>
<td>M. Voorhoeve and T. Basten</td>
<td>Process Algebra with Autonomous Actions, p. 12.</td>
</tr>
<tr>
<td>96/02</td>
<td>P. de Bra and A. Aerts</td>
<td>Multi-User Publishing in the Web: Dress, A Document Repository Service Station, p. 12</td>
</tr>
<tr>
<td>96/06</td>
<td>W.M.P. van der Aalst and T. Basten</td>
<td>Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.</td>
</tr>
<tr>
<td>96/07</td>
<td>M. Voorhoeve</td>
<td>Structural Petri Net Equivalence, p. 16.</td>
</tr>
<tr>
<td>96/11</td>
<td>T. Laan</td>
<td>AUTOMATH and Pure Type Systems, p. 30.</td>
</tr>
<tr>
<td>96/12</td>
<td>F. Kamareddine and T. Laan</td>
<td>A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.</td>
</tr>
<tr>
<td>96/13</td>
<td>T. Borghuis</td>
<td>Priorean Tense Logics in Modal Pure Type Systems, p. 61</td>
</tr>
<tr>
<td>96/15</td>
<td>M.A. Reniers and J.J. Vereijken</td>
<td>Completeness in Discrete-Time Process Algebra, p. 139.</td>
</tr>
<tr>
<td>96/17</td>
<td>E. Boiten and P. Hoogendijk</td>
<td>Nested collections and polytypism, p. 11.</td>
</tr>
<tr>
<td>96/18</td>
<td>P.D.V. van der Stok</td>
<td>Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.</td>
</tr>
<tr>
<td>96/19</td>
<td>M.A. Reniers</td>
<td>Static Semantics of Message Sequence Charts, p. 71</td>
</tr>
<tr>
<td>96/20</td>
<td>L. Feijt</td>
<td>Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.</td>
</tr>
<tr>
<td>96/23</td>
<td>W.M.P. van der Aalst</td>
<td>Structural Characterizations of sound workflow nets, p. 22.</td>
</tr>
<tr>
<td>96/24</td>
<td>M. Voorhoeve and W. van der Aalst</td>
<td>Conservative Adaption of Workflow, p.22</td>
</tr>
<tr>
<td>96/25</td>
<td>M. Vaccari and R.C. Backhouse</td>
<td>Deriving a systolic regular language recognizer, p. 28</td>
</tr>
<tr>
<td>97/06</td>
<td>M. Franssen</td>
<td>Tools for the Construction of Correct Programs: an Overview, p. 33.</td>
</tr>
<tr>
<td>97/07</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Bounded Stacks, Bags and Queues, p. 15.</td>
</tr>
<tr>
<td>Year</td>
<td>Author(s)</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>97/08</td>
<td>P. Hoogendijk and R.C. Backhouse</td>
<td>When do datatypes commute? p. 35.</td>
</tr>
<tr>
<td>97/13</td>
<td>W.M.P. van der Aalst</td>
<td>Exploring the Process Dimension of Workflow Management, p. 56.</td>
</tr>
<tr>
<td>97/14</td>
<td>J.F. Grooto, F. Monin and J. Springintveld</td>
<td>A computer checked algebraic verification of a distributed summation algorithm, p. 28</td>
</tr>
<tr>
<td>97/15</td>
<td>M. Franssen</td>
<td>λP: A Pure Type System for First Order Logic with Automated Theorem Proving, p. 35.</td>
</tr>
<tr>
<td>97/16</td>
<td>W.M.P. van der Aalst</td>
<td>On the verification of inter-organizational workflows, p. 23</td>
</tr>
<tr>
<td>97/17</td>
<td>M. Vaccari and R.C. Backhouse</td>
<td>Calculating a Round-Robin Scheduler, p. 23</td>
</tr>
<tr>
<td>97/18</td>
<td>Werkgemeenschap Informatiewetenschap redactie: P.M.E. De Bra</td>
<td>Informatiewetenschap 1997 Wetenschappelijke bijdragen aan de Vijfde Interdisciplinaire Conferentie Informatiewetenschap, p. 60</td>
</tr>
<tr>
<td>98/01</td>
<td>W. Van der Aalst</td>
<td>Formalization and Verification of Event-driven Process Chains, p. 26</td>
</tr>
<tr>
<td>98/02</td>
<td>M. Voorhoeve</td>
<td>State / Event Net Equivalence, p. 25</td>
</tr>
<tr>
<td>98/03</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Deadlock Behaviour in Split and ST Bisimulation Semantics, p. 15</td>
</tr>
<tr>
<td>98/04</td>
<td>R.C. Backhouse</td>
<td>Pair Algebras and Galois Connections, p. 14</td>
</tr>
<tr>
<td>98/05</td>
<td>D. Dams</td>
<td>Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers, p. 22</td>
</tr>
<tr>
<td>98/09</td>
<td>K.M. van Hee and H.A. Reijers</td>
<td>An analytical method for assessing business processes, p. 29</td>
</tr>
<tr>
<td>98/10</td>
<td>T. Basten and J. Hoemim</td>
<td>Process Algebra in PVS</td>
</tr>
<tr>
<td>98/11</td>
<td>J. Zwanenburg</td>
<td>The Proof-assistant Yarrow, p. 15</td>
</tr>
<tr>
<td>98/13</td>
<td>J.F. Groote, F. Monin and J. v.d. Pol</td>
<td>Checking verifications of protocols and distributed systems by computer. Extended version of a tutorial at CONCUR ’98, p. 27</td>
</tr>
<tr>
<td>98/14</td>
<td>T. Verhoeff (artikel volgt)</td>
<td></td>
</tr>
<tr>
<td>99/01</td>
<td>V. Bos and J.J.T. Kleijn</td>
<td>Structured Operational Semantics of χ , p. 27</td>
</tr>
<tr>
<td>99/02</td>
<td>H.M.W. Verbeek, T. Basten and W.M.P. van der Aalst</td>
<td>Diagnosing Workflow Processes using Woflan, p. 44</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>99/03</td>
<td>R.C. Backhouse and P. Hoogendijk</td>
<td>Final Dialgebras: From Categories to Allegories, p. 26</td>
</tr>
<tr>
<td>99/04</td>
<td>S. Andova</td>
<td>Process Algebra with Interleaving Probabilistic Parallel Composition, p. 81</td>
</tr>
<tr>
<td>99/06</td>
<td>T. Basten and W. v.d. Aalst</td>
<td>Inheritance of Workflows: An Approach to tackling problems related to change, p. 66</td>
</tr>
<tr>
<td>99/10</td>
<td>T.A.C. Willems</td>
<td>The Analysis of a Conveyor Belt System, a case study in Hybrid Systems and timed μ CRL, p. 44</td>
</tr>
<tr>
<td>99/11</td>
<td>J.C.M. Baeten and C.A. Middelburg</td>
<td>Process Algebra with Timing: Real Time and Discrete Time, p. 50</td>
</tr>
<tr>
<td>99/12</td>
<td>S. Andova</td>
<td>Process Algebra with Probabilistic Choice, p. 38</td>
</tr>
<tr>
<td>99/14</td>
<td>A. Engels and S. Mauw</td>
<td>Why men (and octopuses) cannot juggle a four ball cascade, p. 10</td>
</tr>
<tr>
<td>99/16</td>
<td>G.J. Houben, P. Lemmens</td>
<td>A Software Architecture for Generating Hypermedia Applications for Ad-Hoc Database Output, p. 13</td>
</tr>
<tr>
<td>99/18</td>
<td>J.C.M. Baeten and T. Basten</td>
<td>Partial-Order Process Algebra (and its Relation to Petri Nets), p. 79</td>
</tr>
<tr>
<td>99/19</td>
<td>J.C.M. Baeten and C.A. Middelburg</td>
<td>Real Time Process Algebra with Time-dependent Conditions, p.33</td>
</tr>
<tr>
<td>00/01</td>
<td>J.C.M. Baeten and J.A. Bergstra</td>
<td>Mode Transfer in process Algebra, p. 14</td>
</tr>
<tr>
<td>00/02</td>
<td>J.C.M. Baeten</td>
<td>Process Algebra with Explicit Termination, p. 17</td>
</tr>
<tr>
<td>00/03</td>
<td>S. Mauw and M.A. Reniers</td>
<td>A process algebra for interworkings, p. 63</td>
</tr>
<tr>
<td>00/04</td>
<td>R. Bloo, J. Hooman and E. de Jong</td>
<td>Semantical Aspects of an Architecture for Distributed Embedded Systems*, p. 47</td>
</tr>
<tr>
<td>00/05</td>
<td>J.F. Groote and M.A. Reniers</td>
<td>Algebraic Process Verification, p. 65</td>
</tr>
<tr>
<td>00/06</td>
<td>J.F. Groote and J. v. Wamel</td>
<td>The Parallel Composition of Uniform Processes wit Data, p. 19</td>
</tr>
<tr>
<td>00/07</td>
<td>C.A. Middelburg</td>
<td>Variable Binding Operators in Transition System Specifications, p. 27</td>
</tr>
<tr>
<td>00/08</td>
<td>I.D. van den Ende</td>
<td>Grammars Compared: A study on determining a suitable grammar for parsing and generating natural language sentences in order to facilitate the translation of natural language and MSC use cases, p. 33</td>
</tr>
<tr>
<td>00/09</td>
<td>R.R. Hoogerwoord</td>
<td>A Formal Development of Distributed Summation, p. 35</td>
</tr>
<tr>
<td>00/10</td>
<td>T. Willems, J. Tretmans and A. Klop</td>
<td>A Case Study in Formal Methods: Specification and Validation on the OM/RR Protocol, p. 14</td>
</tr>
<tr>
<td>00/11</td>
<td>T. Basten and D. Bolčanči</td>
<td>Enhancing Partial-Order Reduction via Process Clustering, p. 14</td>
</tr>
<tr>
<td>00/12</td>
<td>S. Mauw, M.A. Reniers and T.A.C. Willems</td>
<td>Message Sequence Charts in the Software Engineering Process, p. 26</td>
</tr>
</tbody>
</table>