Discrete time process algebra

Baeten, J.C.M.; Bergstra, J.A.

Published: 01/01/1995

Document Version
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 author's 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

Citation for published version (APA):

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?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Discrete Time Process Algebra

by

J.C.M. Baeten and J.A. Bergstra

95/09

ISSN 0926-4515

All rights reserved
editors: prof.dr. J.C.M. Baeten
prof.dr. M. Rem

Computing Science Report 95/09
Eindhoven, April 1995
Discrete Time Process Algebra
(revised version)

J.C.M. Baeten
Department of Computing Science, Eindhoven University of Technology,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands

J.A. Bergstra
Programming Research Group, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
and
Department of Philosophy, Utrecht University,
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

The axiom system ACP of [BEK84a] was extended with real time features in [BAB91]. Here we proceed to define a discrete time extension of ACP, along the lines of ATP [NIS94]. We present versions based on relative timing and on absolute timing. Both approaches are integrated using parametric timing. The time free ACP theory is embedded in the discrete time theory.

Note: Partial support received from ESPRIT Basic Research Action 7166, CONCUR2.

1. INTRODUCTION.
Process algebra in the form of ACP [BEK84a, BAW90], CCS [Mil89] or CSP [BRHR84] describes the main features of imperative concurrent programming without explicit mention of time. Implicitly, time is present in the interpretation of sequential composition: in \( p \cdot q \) (ACP notation) the process \( p \) must be executed before \( q \). A quantitative view on the relation between process execution and progress of time is absent in these calculi, however. Process algebras can be developed that provide standardised features to incorporate a quantitative view on time. An option is to represent time by means of non-negative reals, and to have time stamps on actions. This is done in [BAB91] for ACP, in [MoT90] for CCS and in [RER88] for CSP. The timed versions of ACP, CCS and CSP in these papers differ concerning the degree to which time stamping is explicit in the notational format of the proposed process algebras. This in turn influences the form of equations, axiomatisations and the appearance of examples.

Another option is to divide time in slices indexed by natural numbers, to have an implicit or explicit time stamping mechanism that provides each action with the time slice in which it occurs and to have a time order within each slice only. This has been worked out in ATP [NSVR90], [NiS94], a process algebra that adds time slicing to a version of ACP based on action prefixing rather than sequential composition. Further, [GR090a] has extended ACP with time slices whereas [MoT89] have added these features to CCS. We propose to use the phrase discrete time process algebra if an enumeration of time slices is used.

The objective of this paper is to extend ACP to a discrete time process algebra. We present three views on discrete time process algebra. In section 3, we consider discrete time process algebra with relative timing, where timing refers to the execution of the previous action. We refer to [BAB92] for a
version where all actions have a time stamp. Here, we present the so-called two-phase version, where the passage of time and the execution of actions is separated. In section 4, we have discrete time process algebra with *absolute timing*, where all timing refers to an absolute clock. Here again, we only consider the two-phase version. In section 5, we have discrete time process algebra with *parametric timing*, where absolute and relative timing are integrated. For parametric timing, we introduce a model based on time spectrum sequences.

There are many practical uses conceivable for timed process algebras. In particular, we mention the TOOLBUS (see [BEK94, 95]). This TOOLBUS contains a program notation called T which is syntactically sugared discrete time process algebra. Programs in T are called T-scripts. The runtime system is also described in terms of discrete time process algebra. By using randomised symbolic execution the TOOLBUS implementation enacts that the axioms of process algebra can be viewed as correctness preserving transformations of T-scripts. A comparable part of discrete time process algebra that is used to describe T-scripts has also been used for the description of φSDL, flat SDL, a subset of SDL that leaves out modularisation and concentrates on timing aspects (see [BEM95]).

We design our algebras (or rather their specifications) in a modular, incremental way. We have absolute time algebras BPA_dat, PA_dat, ACP_dat, relative time algebras BPA_drt, PA_drt, ACP_drt, and parametric time algebras BPA_dpt, PA_dpt, ACP_dpt. This segmentation resembles the segmentation of ACP that can be found in [BAW90]. ACP_dat can be seen as the discretised version of the real time theory ACPp of [BAB91], ACP_dpt the discretised version of the real time theory ACPp\* of [BAB93]. It appears that ACP_drt is an axiom system quite similar to ATP of [NIS94]. The key new operator is a new process constructor \( \sigma_{rel} : P \to P \). The notation \( \sigma \) has been taken from [HER90], the subscript \( rel \) draws attention to the relative time setting. \( \sigma_{rel}(x) \) delays the process \( x \) till the next time slice. We can define all operators of ATP [NSVR90, NIS94] in ACP_drt. These features include: unit delay \( \langle x \rangle \langle y \rangle \) and maximal progress composition \( \oplus \). It turns out that these operators can be eliminated in the presence of those of ACP_drt. (We notice that ATP's \( \oplus \) is already present as \( + \) in TCCS of [MOT89].) As ACP_drt, ATP and TCCS each use strong bisimulation equivalence to obtain a semantic domain, they may be considered as different (and intertranslatable) sets of generators for the same semantic world of processes. The view of [NIS94] that progress of time by itself should not be allowed to introduce non-determinism has had major impact on our work. If time is represented by an action \( \sigma \) then \( \sigma(x + y) = \sigma x + \sigma y \) is an appropriate axiom called time factorisation in [NIS94] (the proposals of [GRO90a], [GM91] do not satisfy time factorisation).

ACKNOWLEDGEMENT: We thank members of the PAM seminar, in particular Michel Reniers (Eindhoven University of Technology), for their helpful remarks.

2. ACP with Immediate Time Stop.
As the basis for our development, we start out from the time free theory ACP, with an additional constant \( \delta \). This process will play a special role, it stands for immediate (and catastrophic) deadlock. The syntax has constants \( a \) (for each \( a \in A \), some given set of atomic actions), a constant \( \delta \) (inaction) and a constant \( \delta \) (immediate deadlock). We have operators \( + \) (alternative composition) and \( \cdot \) (sequential
The theory $\text{BPA}_b^*$ has the axioms $A1-5$, $A6A$, $A7$, $A6ID$, $A7ID$ from Table 1 below, the theory $\text{PA}_b^*$ has in addition the axioms $\text{LMID1}_1$, $\text{CM2*}_1$, $\text{CM2*}_2$, and the axiom $\text{MI}$, which is $\text{CM1}$ without third summand $(X \parallel Y = X \parallel Y + Y \parallel X)$, the theory $\text{ACP}_b^*$ has all axioms from Table 1. The theory $\text{ACP}$ of [BEK84a] is obtained by omitting the constant $\delta$ and the axioms $A6ID$, $A7ID$, $\text{LMID1}_1$, $\text{CM1}_1$, $\text{CM2*}_1$, and strengthening axiom $A6A$ to $A6 (X + \delta = X)$.

**Table 1. ACP$_b^*$**

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X + Y = Y + X$</td>
<td>A1 $a \mid b = b \mid a$</td>
</tr>
<tr>
<td>$(X + Y) + Z = X + (Y + Z)$</td>
<td>A2 $(a \mid b) \mid c = a \mid (b \mid c)$</td>
</tr>
<tr>
<td>$X + X = X$</td>
<td>A3 $\delta \cdot a = \delta$</td>
</tr>
<tr>
<td>$(X + Y) \cdot Z = X \cdot Z + Y \cdot Z$</td>
<td>A4 $X \parallel Y = X \parallel Y + Y \parallel X \parallel Y$</td>
</tr>
<tr>
<td>$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$</td>
<td>A5 $\delta \parallel X = \delta$</td>
</tr>
<tr>
<td>$a + \delta = a$</td>
<td>A6A $X \parallel \delta = \delta$</td>
</tr>
<tr>
<td>$\delta \cdot X = \delta$</td>
<td>A7 $a \parallel (X + \delta) = a \cdot (X + \delta)$</td>
</tr>
<tr>
<td>$X + \delta = X$</td>
<td>A6ID $a \cdot X \parallel (Y + \delta) = a \cdot (X \parallel (Y + \delta))$</td>
</tr>
<tr>
<td>$\delta \cdot X = \delta$</td>
<td>A7ID $(X + Y) \parallel Z = X \parallel Z + Y \parallel Z$</td>
</tr>
<tr>
<td>$\partial H(\delta) = \delta$</td>
<td>DID $a \cdot X \parallel b = (a \mid b) \cdot X$</td>
</tr>
<tr>
<td>$\partial H(a) = a$ \hspace{1cm} if $a \notin H$</td>
<td>D1 $a \parallel b \cdot X = (a \mid b) \cdot X$</td>
</tr>
<tr>
<td>$\partial H(a) = \delta$ \hspace{1cm} if $a \in H$</td>
<td>D2 $a \cdot X \parallel b = (a \mid b) \cdot (X \parallel Y)$</td>
</tr>
<tr>
<td>$\partial H(X + Y) = \partial H(X) + \partial H(Y)$</td>
<td>D3 $(X + Y) \parallel Z = X \parallel Z + Y \parallel Z$</td>
</tr>
<tr>
<td>$\partial H(X \cdot Y) = \partial H(X) \cdot \partial H(Y)$</td>
<td>D4 $X \parallel (Y + Z) = X \parallel Y + X \parallel Z$</td>
</tr>
</tbody>
</table>

3. DISCRETE TIME PROCESS ALGEBRA WITH RELATIVE TIMING.

We present axioms for process algebras in which actions are timed relatively. With $\text{cts}(a)$ we denote the process that will execute $a$ in the time slice in which it is initialised. So if $\text{cts}(a)$ is enabled during slice 7, then $a$ will be performed in the course of slice 7. With the operator $\sigma_{rel}$ processes can be delayed one time slice. So if the process $\sigma_{rel}(\sigma_{rel}(\text{cts}(a) + \text{cts}(b))$ is initialised in slice 5, this has the effect that in slice 7 a choice between $a$ and $b$ must be made. The algebra $\text{ACP}_{\text{drt}}$ presented is a version of ATP tailored towards ACP. It is a slight modification of the axiom system $\text{ACP}_{\text{drt}}$ introduced in [BAB92].
3.1 BASIC PROCESS ALGEBRA.

The signature of $\text{BPA}_{\text{drt}}$ has besides the constant $\cdot$, constants $\text{cts}(a)$ (for $a \in A_\delta$), denoting $a$ in the current time slice. The superscript $\sim$ denotes that time free atoms are not part of the signature. We can also denote $\text{cts}(a)$ as $\cdot a[1]$ or $a\!\cdot$ (the alternative notations were used in [BAB92]). Within a time slice, there is no explicit mention of the passage of time, we can see the passage to the next time slice as a clock tick. Thus, the $\text{cts}(a)$ are the nondelayable actions as can be found e.g. in [Nis94]: the action must occur before the next clock tick. The operators are alternative and sequential composition, and the relative discrete time unit delay $\sigma_{\text{rel}}$ (adapted from [Her90], can also be denoted $\sigma_{\text{rel}}(1)$ or $\sigma_d$; the last notation was used in [BAB92]). The process $\sigma_{\text{rel}}(X)$ will start $X$ after one clock tick, i.e. in the next time slice.

The signature of $\text{BPA}_{\text{drt}}$ adds to this the constants $a$, for $a \in A_\delta$. These are the delayable actions, a number of clock ticks can occur before the action is executed. The axioms of $\text{BPA}_{\text{drt}}$ are A1-5, A6ID, A7ID of $\text{BPA}_\delta$ plus DRT1-4 of table 2 below, $\text{BPA}_{\text{drt}}$ adds to this A6A, A7 and the axiom ADRT.

The axiom DRT1 is the time factorization axiom: it says that the passage of time by itself cannot determine a choice. Axiom DRT3 identifies processes that can be distinguished: $\text{cts}(\delta)$ denotes a deadlock at the end of the current time slice, and $\sigma_{\text{rel}}(\delta)$ denotes an immediate deadlock at the beginning of the next time slice. This is the difference between open and closed time stops that we discussed in [BAB93], section 8.1. Here, as in [BAB93], we abstract from this difference. This simplifies the algebra considerably, but makes the operational semantics somewhat more complicated.

$$\sigma_{\text{rel}}(X) + \sigma_{\text{rel}}(Y) = \sigma_{\text{rel}}(X + Y) \quad \text{DRT1}$$
$$\sigma_{\text{rel}}(X) \cdot Y = \sigma_{\text{rel}}(X \cdot Y) \quad \text{DRT2}$$
$$\sigma_{\text{rel}}(\delta) = \text{cts}(\delta) \quad \text{DRT3}$$

<table>
<thead>
<tr>
<th>Table 2. $\text{BPA}<em>{\text{drt}} = \text{BPA}</em>\delta + \text{DRT1}-4$, ADRT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{\text{rel}}(X) + \sigma_{\text{rel}}(Y) = \sigma_{\text{rel}}(X + Y)$</td>
</tr>
<tr>
<td>$\sigma_{\text{rel}}(X) \cdot Y = \sigma_{\text{rel}}(X \cdot Y)$</td>
</tr>
<tr>
<td>$\sigma_{\text{rel}}(\delta) = \text{cts}(\delta)$</td>
</tr>
<tr>
<td>$a = \text{cts}(a) + \sigma_{\text{rel}}(a)$</td>
</tr>
<tr>
<td>$\sigma_{\text{rel}}(X) \cdot Y = \sigma_{\text{rel}}(X \cdot Y)$</td>
</tr>
<tr>
<td>$\sigma_{\text{rel}}(\delta) = \text{cts}(\delta)$</td>
</tr>
<tr>
<td>$\text{cts}(a) + \text{cts}(\delta) = \text{cts}(a)$</td>
</tr>
</tbody>
</table>

3.2 STRUCTURED OPERATIONAL SEMANTICS.

We give a semantics in terms of Plotkin-style operational rules. As set of states $S$ we have the set of closed process expressions, and we define the following relations on states:

- time step $\subseteq S \times S$, notation $s \xrightarrow{\text{ts}} s'$ (denotes passage to the next time slice)
- action step $\subseteq S \times A \times S$, notation $s \xrightarrow{a} s'$ (denotes action execution)
- action termination $\subseteq S \times A$, notation $s \xrightarrow{a} \top$ (execution of a terminating action)
- immediate deadlock $\subseteq S$, notation $s \xrightarrow{\text{ID}}$ (holds only for terms equal to $\delta$).

A relation holds on $S$ just when it derivable (as defined in [Ver94]) from the following rules.
Discrete time process algebra

\[
\begin{align*}
\text{cts}(a) & \xrightarrow{a} \top \\
\neg \text{ID}(x) & \xrightarrow{\sigma_{\text{rel}}(x) \; a \quad x} \\
\text{ID}(\delta) & \xrightarrow{a \quad \top} \\
\delta & \xrightarrow{a} \delta \\
x & \xrightarrow{a} x' \\
x & \xrightarrow{\top} \\
x \cdot y & \xrightarrow{\top} y \\
x \cdot y & \xrightarrow{x' \cdot y} \\
x & \xrightarrow{\top} \\
x & \xrightarrow{x' \cdot y} \\
x & \xrightarrow{\top} \\
x \cdot y & \xrightarrow{\top} \\
x & \xrightarrow{\top} \\
x & \xrightarrow{\top} \\
\end{align*}
\]

\text{TABLE 3.} Operational rules for BPA_{drf}.

In order to show that these rules give rise to a unique transition relation on closed terms, it helps to use results and terminology of [VER94]. Bisimulation is defined as usual, so a symmetric binary relation \( R \) on process terms is a bisimulation iff the following transfer conditions hold:

i. if \( R(p,q) \) and \( p \xrightarrow{a} p' \quad (u \in A \cup \{o\}) \), then there is \( q' \) such that \( q \xrightarrow{a} q' \) and \( R(s',t') \);

ii. if \( R(p,q) \), then \( p \xrightarrow{a} \top \quad (a \in A) \) iff \( q \xrightarrow{a} \top \).

iii. if \( R(p,q) \), then \( \text{ID}(p) \) iff \( \text{ID}(q) \).

Two terms \( p,q \) are bisimilar, \( p \leftrightarrow q \), if there exists a bisimulation relating them. We can make the set of process terms modulo bisimulation into a model for BPA_{drf}. In order to do this, we first need to know that bisimulation is a congruence. [VER94] proves that this is true, as long as the operational rules satisfy his "panih" format. It is easy to establish that this is indeed the case.

3.3 Basic Terms.

We define the set of basic terms \( B_{drf} \) over BPA_{drf} inductively. We use the auxiliary sets \( B_{drf}^1 \) (basic terms that start in the current time slice) and \( B_{drf}^\infty \) (basic terms that can start after arbitrary many ticks), to be defined simultaneously.

1. \( B_{drf}^1 \subseteq B_{drf} \)
2. \( B_{drf}^\infty \subseteq B_{drf} \)
3. if \( a \in A \), then \( \text{cts}(a) \in B_{drf}^1 \)
4. \( \delta \in B_{drf}^\infty \)
5. if \( a \in A \) and \( t \in B_{drf} \), then \( \text{cts}(a) \cdot t \in B_{drf}^1 \)
6. if \( a \in A \), then \( a \in B_{drf}^\infty \)
7. if \( t,s \in B_{drf}^1 \), then \( t+s \in B_{drf}^1 \)
8. if \( a \in A \) and \( t \in B_{drf} \), then \( a \cdot t \in B_{drf}^\infty \)
9. \( \delta \in B_{drf} \)
10. if \( t,s \in B_{drf}^1 \), then \( t+s \in B_{drf}^1 \)
11. if \( t \in B_{drf}^1 \), then \( \sigma_{\text{rel}}(t) \in B_{drf} \)
12. if \( t \in B_{drf}^1 \) and \( s \in B_{drf} \), then \( t + \sigma_{\text{rel}}(s) \in B_{drf} \).

Basic terms over BPA_{drf} are defined by clauses 1-12; we obtain basic terms over BPA_{drf} by omitting clauses 2,4,6,8,10. Some reflection leads to the insight that the basic terms over BPA_{drf} are exactly the terms of one of the following three forms:
3.4 Elimination.
Let \( t \) be a closed \( \text{BPA}_{\text{drt}} \)-term. Then there is a basic term \( s \) over \( \text{BPA}_{\text{drt}} \) such that \( \text{BPA}_{\text{drt}} \vdash t = s \).

Let \( t \) be a closed \( \text{BPA}_{\text{drt}} \)-term. Then there is a basic term \( s \) over \( \text{BPA}_{\text{drt}} \) such that \( \text{BPA}_{\text{drt}} \vdash t = s \).

3.5 Soundness and Completeness.
Let \( p, q \) be closed \( \text{BPA}_{\text{drt}} \)-expressions. Then we have \( \text{BPA}_{\text{drt}} \vdash p = q \) if and only if \( p \sim q \), i.e. \( \text{BPA}_{\text{drt}} \) is sound and complete for the set of transition graphs modulo bisimulation.

Let \( p, q \) be closed \( \text{BPA}_{\text{drt}} \)-expressions. Then we have \( \text{BPA}_{\text{drt}} \vdash p = q \) if and only if \( p \sim q \), so also \( \text{BPA}_{\text{drt}} \) is a sound and complete axiomatisation.

3.6 Conservativity.
\( \text{BPA}_{\text{drt}} \) is a conservative extension of \( \text{BPA}^\delta \), i.e. for all closed \( \text{BPA}^\delta \)-terms \( s, t \) we have
\[
\text{BPA}^\delta \vdash s = t \iff \text{BPA}_{\text{drt}} \vdash s = t.
\]

3.7 Graph Model.
Thus, we have found a sound and complete model for \( \text{BPA}_{\text{drt}} \). It is also possible to define a graph model for \( \text{BPA}_{\text{drt}} \) directly, not by giving a transition system specification, but by defining an interpretation of the constants and operators on process graphs. We define a set of process graphs as in [BAW90] with labels from \( A \cup \{\alpha\} \) satisfying the extra condition that every node has at most one outgoing \( \sigma \)-labeled edge. Moreover, a termination node can have a label \( \checkmark \) (for successful termination), or \( \text{ID} \) (for immediate deadlock). A \( \sigma \)-edge may not lead to a \( \checkmark \)-node.

Let \( \mathcal{G} \) be the set of such process graphs. To state this precisely, an element of \( \mathcal{G} \) is a sixtuple \( \langle N, E, r, T, \text{ID}, \checkmark \rangle \) where \( N \) is the set of nodes, \( E \subseteq (N-T) \times (A \cup \{\alpha\}) \times N \) is the set of edges, \( r \in N \) is the root node, \( T \subseteq N \) is the set of termination nodes, and \( \text{ID} \subseteq T, \checkmark \subseteq T-\text{ID} \).

We define an interpretation as follows (\( n \in N \)):
1. The constant \( \delta \) is mapped to the graph with 1 node, the root, labeled \( \checkmark \).
2. The constant \( \text{cts}(\alpha) \) is mapped to the process graph with 2 nodes, the first the root, the last a \( \checkmark \)-node, connected by an \( \alpha \)-edge. The constant \( \alpha \) adds to this graph a \( \sigma \)-edge from the root to itself.
3. The constant \( \text{cts}(\delta) \) is mapped to the process graph with 1 node, the root, and no edges, no node-labels. The constant \( \delta \) adds to this graph a \( \sigma \)-edge from the root to itself.
4. Given graphs \( g, h \), we first have to root-unwind both graphs in order to form the sum (see e.g. [BAW90]). In case one graph is the \( \delta \)-graph, the sum is simply the other graph. Then, the graph \( g+h \) is
formed by identifying the roots of $g$ and $h$. Next, if both roots have an outgoing $\sigma$-edge, both these edges are removed, and a new $\sigma$-edge is added to the sum of the graphs the original $\sigma$-edges where going to. If necessary, the procedure is repeated.

5. Given graph $g,h$, in order to form $g \cdot h$, append at each $v'$-node of $g$ a copy of $h$.

6. Given graph $g$, the tree $\text{Orel}(g)$ is formed by removing the ID-label in case $g$ is the $\delta$-graph, and otherwise, by adding a new root, with a $\sigma$-edge from the new root to the old root.

3.8 PROJECTION.
We can also define a model using a projective limit construction as in [BAB84b]. In order to do this, we define projections. The relative time $n$th projection operator $\pi_n^{rel}$ cuts off a process after $n$ steps have been executed. Here, we count both time steps and action steps. We give axioms in table 4 below. We have that every closed term over the extended signature, including projection operators, can also be written as a basic term.

To give an example, $\pi_0^{rel}(\text{cts}(a) \cdot \text{cts}(\delta)) = \delta$, $\pi_1^{rel}(\text{cts}(a) \cdot \text{cts}(\delta)) = \text{cts}(a) \cdot \delta$, and $\pi_n^{rel}(\text{cts}(a) \cdot \text{cts}(\delta)) = \text{cts}(a) \cdot \text{cts}(\delta)$ for $n \geq 2$. The projective limit model has as domain sequences of closed terms $(p_n)_n \in \mathbb{N}$ such that $\pi_n^{rel}(p_{n+1}) = p_n$. Operators are defined pointwise. The given axiomatisation for BPA$_{drt}$ is also sound and complete for this model.

<table>
<thead>
<tr>
<th>$\pi_0^{rel}(X) = \delta$</th>
<th>PRR1</th>
<th>$\pi_n^{rel}(\text{cts}(a)) = \text{cts}(a)$</th>
<th>PRR4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_n^{rel}(\delta) = \delta$</td>
<td>PRR2</td>
<td>$\pi_n^{rel}(\text{cts}(a) \cdot X) = \text{cts}(a) \cdot \pi_n^{rel}(X)$</td>
<td>PRR5</td>
</tr>
<tr>
<td>$\pi_{n+1}^{rel}(\text{Orel}(X)) = \text{Orel}(\pi_n^{rel}(X))$</td>
<td>PRR3</td>
<td>$\pi_n^{rel}(X + Y) = \pi_n^{rel}(X) + \pi_n^{rel}(Y)$</td>
<td>PRR6</td>
</tr>
</tbody>
</table>

TABLE 4. Projections.

Next, we introduce parallel composition. We start by describing a system with merge without communication, a so-called free merge. Different from the real time case of [BAB91], in the discrete time case, it is possible to consider merge without communication, as we will have $\text{cts}(a) \parallel \text{cts}(b) = \text{cts}(a) \cdot \text{cts}(b) + \text{cts}(b) \cdot \text{cts}(a)$.

3.9 FREE MERGE.
In PA$_{drt}$, we add the parallel composition operator $\parallel$ (merge) with its auxiliary operator $\parallel$ (left merge).

PA$_{drt}$ has the axioms of BPA$_{drt}$, axioms M1, M4, LMID1,2 of section 2 and the axioms in table 5 below ($a \in A, x,y,z \in P$). Notice that $\text{Orel}(X) \parallel \text{cts}(\delta) = \text{cts}(\delta)$ can be derived using DRT7, DRT3, LMID2. PA$_{drt}$ adds the additional axioms of BPA$_{drt}$ and the axioms CM2*,3* of section 2.

The operational semantics adds the rules of table 6 to the rules of BPA$_{drt}$.

<table>
<thead>
<tr>
<th>$\text{cts}(a) \parallel (X + \text{cts}(\delta)) = \text{cts}(a) \cdot (X + \text{cts}(\delta))$</th>
<th>DRTCM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{cts}(a) \cdot X \parallel (Y + \text{cts}(\delta)) = \text{cts}(a) \cdot (X \parallel (Y + \text{cts}(\delta)))$</td>
<td>DRTCM3</td>
</tr>
<tr>
<td>$\text{Orel}(X) \parallel (\text{cts}(a) + Y) = \text{Orel}(X) \parallel (\text{cts}(\delta) + Y)$</td>
<td>DRT5</td>
</tr>
<tr>
<td>$\text{Orel}(X) \parallel (\text{cts}(a) \cdot Y + Z) = \text{Orel}(X) \parallel (\text{cts}(\delta) + Z)$</td>
<td>DRT6</td>
</tr>
<tr>
<td>$\text{Orel}(X) \parallel \text{Orel}(Y) = \text{Orel}(X \parallel Y)$</td>
<td>DRT7</td>
</tr>
</tbody>
</table>

TABLE 5. PA$_{drt} = $ BPA$_{drt}$ + M1,4, LMID1,2, DRTCM2,3, CM2*,3*, DRT5-7.
\[
x \xrightarrow{a} x', \quad \neg \text{ID}(y)
\]
\[
x \| y \xrightarrow{a} x' \| y, \quad y \| x \xrightarrow{a} y \| x',\quad x \| y \xrightarrow{a} x \| y
\]
\[
x \xrightarrow{a} \sqrt{\cdot}, \quad \neg \text{ID}(y)
\]
\[
x \| y \xrightarrow{a} y, \quad y \| x \xrightarrow{a} y, \quad x \| y \xrightarrow{a} y
\]

<table>
<thead>
<tr>
<th>Table 6. Additional operational rules for PA\text{dr}.</th>
</tr>
</thead>
<tbody>
<tr>
<td>x \xrightarrow{a} x', y \xrightarrow{a} y'</td>
</tr>
<tr>
<td>x | y \xrightarrow{a} x' | y', x | y \xrightarrow{a} x | y'</td>
</tr>
</tbody>
</table>

We have results as before:

Let \( t \) be a closed PA\text{dr}-term. Then there is a basic term \( s \) over BP\text{dr} such that PA\text{dr} \vdash t = s.

Let \( t \) be a closed PA\text{dr}-term. Then there is a basic term \( s \) over BP\text{dr} such that PA\text{dr} \vdash t = s.

Let \( p, q \) be closed PA\text{dr}-expressions. Then we have PA\text{dr} \vdash p = q \iff p \leftrightarrow q.

Let \( p, q \) be closed PA\text{dr}-expressions. Then we have PA\text{dr} \vdash p = q \iff p \leftrightarrow q.

PA\text{dr} is a conservative extension of PA\text{dr}.

As an example, we calculate
\[
(c\text{ts}(a) + \sigma_{\text{rel}}(c\text{ts}(b))) \| (c\text{ts}(b) + \sigma_{\text{rel}}(c\text{ts}(a))) =
\]
\[
= c\text{ts}(a) \cdot (c\text{ts}(b) + \sigma_{\text{rel}}(c\text{ts}(a))) + c\text{ts}(b) \cdot (c\text{ts}(a) + \sigma_{\text{rel}}(c\text{ts}(b))) + \sigma_{\text{rel}}(c\text{ts}(a) \cdot c\text{ts}(b) + c\text{ts}(b) \cdot c\text{ts}(a)).
\]

3.10 COMMUNICATION.

We add the communication merge \( \| \) based on a given commutative, associative function on \( A_\delta \) with \( \delta \) as neutral element, and the encapsulation operator \( \delta_H \). ACP\text{dr} modifies PA\text{dr} by replacing axiom M1 by CM1 and adding axioms C1-3, CMID1,2, CM8,9, D3,4 of ACP\text{dr} and the additional axioms in table 7, ACP\text{dr} adds also the remaining axioms of ACP\text{dr} to PA\text{dr}. Operational rules are given in table 8. We obtain elimination, soundness, completeness, conservativity as before.

\[
\begin{align*}
\text{cts}(a) \| c\text{ts}(b) &= \text{cts}(a \| b) & \text{DRTC} & \text{cts}(a) \| \sigma_{\text{rel}}(X) &= \text{cts}(\delta) & \text{DRT8} \\
\text{cts}(a) \cdot X \| c\text{ts}(b) &= c\text{ts}(a \| b) \cdot X & \text{DRTC1} & \sigma_{\text{rel}}(X) \| c\text{ts}(a) &= c\text{ts}(\delta) & \text{DRT9} \\
\text{cts}(a) \| c\text{ts}(b) \cdot X &= \text{cts}(a \| b) \cdot X & \text{DRTC1} & \sigma_{\text{rel}}(X) \| c\text{ts}(a) &= c\text{ts}(\delta) & \text{DRT9} \\
\text{cts}(a) \cdot X \| c\text{ts}(b) \cdot Y &= \text{cts}(a \| b) \cdot (X \| Y) & \text{DRTC1} & \sigma_{\text{rel}}(X) \| c\text{ts}(a) \cdot Y &= c\text{ts}(\delta) & \text{DRT11} \\
\delta_H(c\text{ts}(a)) &= c\text{ts}(a) & \text{DRTD1} & \sigma_{\text{rel}}(X) \| \sigma_{\text{rel}}(Y) &= \sigma_{\text{rel}}(X \| Y) & \text{DRT12} \\
\delta_H(c\text{ts}(a)) &= c\text{ts}(\delta) & \text{DRTD2} & \sigma_{\text{rel}}(X) \| \sigma_{\text{rel}}(Y) &= \sigma_{\text{rel}}(\delta_H(X)) & \text{DRT13}
\end{align*}
\]

| Table 7. ACP\text{dr} = PA\text{dr} - M1 + ACP\text{dr} + DRTC, DRTC1-7, DRTD1,2, DRT8-13. |
Discrete time process algebra

\[
\begin{align*}
\text{Table 8. Additional operational rules of } & \text{ACP}_{dt}. \\
\end{align*}
\]

An example of the elimination of the merge is as follows (with \(a \mid b = c\)):

\[
\begin{align*}
\text{(cts}(a) + \sigma_{rel}(\text{cts}(b))) \parallel (\text{cts}(b) + \sigma_{rel}(\text{cts}(a))) &= \text{cts}(a) \cdot (\text{cts}(b) + \sigma_{rel}(\text{cts}(a))) + \\
+ \text{cts}(b) \cdot (\text{cts}(a) + \sigma_{rel}(\text{cts}(b))) + \text{cts}(c) + \sigma_{rel}(\text{cts}(a) \cdot \text{cts}(b)) + \text{cts}(b) \cdot \text{cts}(a) + \text{cts}(c).
\end{align*}
\]

4. DISCRETE TIME PROCESS ALGEBRA WITH ABSOLUTE TIMING.

We present a version of the theory in section 3 using absolute timing, where all timing is related to a global clock. Similar to section 3, we do not consider a version with timestamped actions (as we did in [BAB92]), but instead separate the execution of actions from the passage of time, a two-phase theory in the terminology of [NIS94].

4.1 BASIC PROCESS ALGEBRA.

We start with constants \(\text{fts}(a)\), denoting \(a\) in the first time slice \((a \in A_0)\). An alternative notation is \(a(1)\). Besides, we have the constant \(\delta\), and operators \(+\cdot\) as before. In addition, we have the absolute discrete time unit delay \(\sigma_{abs}\). Axiom DAT7 uses the extra operator \(\mathbb{I} \parallel (\mathbb{I})\) (absolute value). In case of ambiguities or overlap with additional syntax one may write \(\mathbb{I} \odot\) for this operator. This operator is the identity for all processes using absolute timing only. We will need this operator when we want to mix relative and absolute timing further on. The axioms of BPA\(_{dat}\) are A1-5, A6ID, A7ID of BPA\(_{\delta}\) plus DAT1-7, AV1-5 of table 9 below, BPA\(_{dat}\) adds to this A6A, A7, AV6 and the axiom ADAT. Notice that we can derive \(X + \text{fts}(\delta) = X\) for all closed terms \(X\) except \(\delta\), i.e. for all closed terms \(X\) with \(\neg \text{ID}(X)\). Also notice that \(\text{fts}(\delta) \cdot X = \text{fts}(\delta)\) can be derived for all closed terms.
\[
\begin{align*}
\sigma_{abs}(\delta) &= ftz(\delta) & \text{DAT1} & \sigma_{abs}(\delta) = \delta & \text{AV1} \\
fts(a) + ftz(\delta) &= ftz(a) & \text{DAT2} & \lfts(a) = ftz(a) & \text{AV2} \\
\sigma_{abs}(X) + \sigma_{abs}(Y) &= \sigma_{abs}(X + Y) & \text{DAT3} & |X + Y| = |XI| + |Y| & \text{AV3} \\
\sigma_{abs}(X)\delta &= \sigma_{abs}(X\delta) & \text{DAT4} & |X-Y| = |XI-Y| & \text{AV4} \\
\sigma_{abs}(X)\sts(a) + Y &= \sigma_{abs}(X\sts(a) + Y) & \text{DAT5} & l\sigma_{abs}(X)l = \sigma_{abs}(lXIl) & \text{AV5} \\
\sigma_{abs}(X)\sts(a)\cdot Y + Z &= \sigma_{abs}(X\sts(a)\cdot Y + Z) & \text{DAT6} & |a| = a & \text{AV6} \\
\sigma_{abs}(X)\cdot \sigma_{abs}(Y) &= \sigma_{abs}(X\cdot \sigma_{abs}(Y)) & \text{DAT7} & a = \sts(a) + \sigma_{abs}(a) & \text{ADAT} \\
\end{align*}
\]

\textbf{Table 9.} \(BPA_{dat} = BPA_{\delta} + \text{DAT1-7}, \text{AV1-6, ADAT}.\)

4.2 \textbf{Structured operational semantics.}

The operational rules are more complicated in this case, as we have to keep track of which time slice we are in, we have to keep track of the global clock. \(\langle x,n \rangle\) denotes \(x\) in the \((n+1)\)st time slice. We have:

- if \(\langle x,n \rangle \not\rightarrow \langle x',n' \rangle\), then \(x = x', n' = n+1\)
- if \(\langle x,n \rangle \not\rightarrow \langle x',n' \rangle\) or \(\langle x,n \rangle \not\rightarrow \langle v,n' \rangle\), then \(n' = n\).

The operational rules for the absolute value operator are trivial.

\[
\begin{align*}
\langle \fts(a), 0 \rangle \not\rightarrow \langle v, 0 \rangle & \quad \text{ID}((\delta, n)) \quad \langle a, n \rangle \not\rightarrow \langle a, n+1 \rangle \\
\langle \sigma_{abs}(x), 0 \rangle \not\rightarrow \langle \sigma_{abs}(x), 1 \rangle & \quad \text{ID}((\sigma_{abs}(x), n+1)) \\
\langle x, n \rangle \not\rightarrow \langle x', n \rangle & \quad \text{ID}((\sigma_{abs}(x), n+1)) \not\rightarrow \langle \sigma_{abs}(x), n+2 \rangle \\
\langle x+y, n \rangle \not\rightarrow \langle x', n \rangle, \langle y+x, n \rangle \not\rightarrow \langle x', n \rangle & \quad \langle x+y, n \rangle \not\rightarrow \langle v, n \rangle, \langle y+x, n \rangle \not\rightarrow \langle v, n \rangle \\
\langle x+y, n \rangle \not\rightarrow \langle x+y+n+1, \langle y+x, n \rangle \not\rightarrow \langle y+x, n+1 \rangle & \quad \text{ID}((\langle x, n \rangle), \text{ID}((\langle y, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle x', n \rangle & \quad \text{ID}((\langle x+y, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle x', n \rangle & \quad \text{ID}((\langle x+y, n \rangle)) \\
\langle x+y, n \rangle \not\rightarrow \langle x+y, n+1 \rangle, \langle y+x, n \rangle \not\rightarrow \langle y+x, n+1 \rangle & \quad \langle x+y, n \rangle \not\rightarrow \langle v, n \rangle, \langle y+x, n \rangle \not\rightarrow \langle v, n \rangle \\
\langle x, n \rangle \not\rightarrow \langle x', n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle x, n+1 \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle v, n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle v, n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, y, n \rangle \not\rightarrow \langle x', y, n \rangle & \quad \langle x, y, n \rangle \not\rightarrow \langle x', y, n+1 \rangle \\
\langle x+y, n \rangle \not\rightarrow \langle x+y, n+1 \rangle & \quad \langle x+y, n \rangle \not\rightarrow \langle y, n \rangle \\
\langle x, n \rangle \not\rightarrow \langle v, n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle v, n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\langle x, n \rangle \not\rightarrow \langle v, n \rangle & \quad \text{ID}((\langle x, n \rangle)) \\
\end{align*}
\]

\textbf{Table 10.} Operational semantics of \(BPA_{dat}\).

4.3 \textbf{Bisimulation.}

We also have to adapt the definition of bisimulation. A \textit{bisimulation} is a symmetric binary relation \(R\) on \(P \times N\) such that \((a \in A)\):
i. whenever $R((s, n), (s', n'))$ then $n = n'$ and $\text{ID}(s) \iff \text{ID}(s')$

ii. whenever $R((s, n), (t, n))$ and $(s, n) \xrightarrow{a} (s', n)$, then there is a process expression $t'$ such that $(t, n) \xrightarrow{a} (t', n)$ and $R((s', n), (t', n))$

iii. whenever $R((s, n), (t, n))$ and $(s, n) \not\xrightarrow{a} (s', n)$, then there is a process expression $t'$ such that $(t, n) \not\xrightarrow{a} (t', n)$ and $R((s', n), (t', n))$

iv. whenever $R((s, n), (t, n))$ and $(s, n) \xrightarrow{a} (s, n+1)$, then $(t, n) \not\xrightarrow{a} (t, n+1)$ and $R((s, n+1), (t, n+1))$

We say process expressions $x$ and $y$ are bisimilar, denoted $x \leftrightarrow y$, if there exists a bisimulation with $R((x,0), (y,0))$. We state without proof that bisimulation is a congruence relation on these transition systems.

4.4 BASIC TERMS.

We define the notion of a basic term as in section 3. The set of basic terms is $B_{\text{dat}}$, and we have auxiliary sets $B_{\text{dat}}^1$ (basic terms that start in the first time slice) and $B_{\text{dat}}^\infty$ (basic terms that start in an arbitrary time slice).

1. $B_{\text{dat}}^1 \subseteq B_{\text{dat}}$
2. $B_{\text{dat}}^\infty \subseteq B_{\text{dat}}$
3. if $a \in A$, then $\text{fts}(a) \in B_{\text{dat}}^1$
4. $\delta \in B_{\text{dat}}^\infty$
5. if $a \in A$ and $t \in B_{\text{dat}}$, then $\text{fts}(a) \cdot t \in B_{\text{dat}}^1$
6. if $a \in A$, then $a \in B_{\text{dat}}^\infty$ and $a \delta \in B_{\text{dat}}^\infty$
7. if $t, s \in B_{\text{dat}}^1$, then $t + s \in B_{\text{dat}}^1$
8. if $a \in A$ and $t \in B_{\text{dat}}^\infty$, then $a \cdot t \in B_{\text{dat}}^\infty$
9. $\delta \in B_{\text{dat}}^\infty$
10. if $t, s \in B_{\text{dat}}^\infty$, then $t + s \in B_{\text{dat}}^\infty$
11. if $t \in B_{\text{dat}}$, then $\sigma_{\text{abs}}(t) \in B_{\text{dat}}$
12. if $t \in B_{\text{dat}}^1$ and $s \in B_{\text{dat}}$, then $t + \sigma_{\text{abs}}(s) \in B_{\text{dat}}$.

Basic terms over $BPA_{\text{dat}}$ are defined by clauses 1-12; we obtain basic terms over $BPA_{\text{dat}}$ by omitting clauses 2,4,6,8,10. Notice the different formulation in clauses 6 and 8. As an example, consider the following calculation: $a \cdot \sigma_{\text{abs}}(\text{fts}(b)) = \text{fts}(a) \cdot \sigma_{\text{abs}}(\text{fts}(b)) + \sigma_{\text{abs}}(\text{fts}(a) \cdot \text{fts}(b) + \sigma_{\text{abs}}(a \delta))$

Again, the basic terms over $BPA_{\text{dat}}$ are exactly the terms of one of the following three forms:

a. $\delta$

b. $\sum_{k=0}^{n} \text{fts}(a_k) \cdot t_k + \sum_{j=0}^{m} \text{fts}(b_j)$, with $n+m>0$, $a_k, b_j \in A$ and $t_k$ basic;

c. $\sum_{k=0}^{n} \text{fts}(a_k) \cdot t_k + \sum_{j=0}^{m} \text{fts}(b_j) + \sigma_{\text{abs}}(t)$, with $n+m>0$, $a_k, b_j \in A$ and $t, t_k$ basic.

In the case of $BPA_{\text{dat}}$, we can have extra summands of the form $a (a \in A_b)$ or $a \cdot t$ in form b. As in 3.4-6, we obtain elimination, soundness and completeness. As in 3.7, we can define a graph model.

4.5 PROJECTION.

We can define projections as in 3.8. We give axioms in table 11 below. As before, we can define a projective limit model. Again, we have elimination, soundness and completeness.

| PRA1 | $\pi_n^\text{abs}(\text{fts}(a)) = \text{fts}(a)$ |
| PRA2 | $\pi_n^\text{abs}(a \cdot X) = a \cdot \pi_n^\text{abs}(X)$ |
| PRA3 | $\pi_n^\text{abs}(X + Y) = \pi_n^\text{abs}(X) + \pi_n^\text{abs}(Y)$ |

TABLE 11. Projections.
4.6 FREE MERGE.

The addition of parallel composition is analogous to the relative time case. We show the axioms in table 12, the operational rules in table 13. We have elimination, soundness, completeness as before. A sample calculation: \( \text{fts}(a) \parallel \text{fts}(b) \cdot \delta = \text{fts}(a) \cdot \text{fts}(b) \cdot \delta + \text{fts}(b) \cdot \delta \).

\[
\begin{align*}
\text{fts}(a) \parallel (X + \text{fts}(\delta)) &= \text{fts}(a) \cdot (X + \text{fts}(\delta)) & \text{DATCM2} \\
\text{fts}(a) \cdot X \parallel (Y + \text{fts}(\delta)) &= \text{fts}(a) \cdot (X \parallel (Y + \text{fts}(\delta))) & \text{DATCM3} \\
\sigma_{\text{abs}}(X) \parallel (\text{fts}(a) + Y) &= \sigma_{\text{abs}}(X) \parallel (\text{fts}(\delta) + Y) & \text{DAT8} \\
\sigma_{\text{abs}}(X) \parallel (\text{fts}(a) \cdot Y + Z) &= \sigma_{\text{abs}}(X) \parallel (\text{fts}(\delta) + Z) & \text{DAT9} \\
\sigma_{\text{abs}}(X) \parallel \sigma_{\text{abs}}(Y) &= \sigma_{\text{abs}}(X \parallel Y) & \text{DAT10}
\end{align*}
\]

**TABLE 12.** PA\(_{\text{dat}}\) = BPA\(_{\text{dat}}\) + M1,4, LMID1,2, DATCM2,3, CM2*,3*, DAT8-10.

\[
\begin{align*}
\langle x, n \rangle^a \rightarrow \langle x', n \rangle, \sim \text{ID}(\langle y, n \rangle) \\
\langle x, n \rangle^a \rightarrow \langle y, n \rangle, \langle x \parallel x, n \rangle^a \rightarrow \langle y \parallel x, n \rangle, \langle x \parallel y, n \rangle^a \rightarrow \langle x' \parallel y, n \rangle
\end{align*}
\]

\[
\begin{align*}
\langle x, n \rangle^a \rightarrow \langle y, n \rangle, \langle x \parallel x, n \rangle^a \rightarrow \langle y, n \rangle, \langle x \parallel y, n \rangle^a \rightarrow \langle y, n \rangle
\end{align*}
\]

\[
\begin{align*}
\langle x, n \rangle^a \rightarrow \langle x, n + 1 \rangle, \langle y, n \rangle^a \rightarrow \langle y, n + 1 \rangle \\
\langle x \parallel y, n \rangle^a \rightarrow \langle x \parallel y, n + 1 \rangle, \langle x \parallel y, n \rangle^a \rightarrow \langle x \parallel y, n + 1 \rangle
\end{align*}
\]

**TABLE 13.** Operational semantics of PA\(_{\text{dat}}\).

4.7 ALGEBRA OF COMMUNICATING PROCESSES.

The addition of communication is analogous to the relative time case. We show the axioms in table 14, the operational rules in table 15. We have elimination, soundness, completeness as in the case of relative time.

\[
\begin{align*}
\text{fts}(a) \parallel \text{fts}(b) &= \text{fts}(a \parallel b) & \text{DATC} \\
\text{fts}(a) \cdot X \parallel \text{fts}(b) &= \text{fts}(a \cdot b) \cdot X & \text{DATCM5} \\
\text{fts}(a) \parallel \text{fts}(b) \cdot X &= \text{fts}(a \cdot b) \cdot X & \text{DATCM6} \\
\text{fts}(a) \cdot X \parallel \text{fts}(b) \cdot Y &= \text{fts}(a \cdot b) \cdot (X \parallel Y) & \text{DATCM7} \\
\partial_H(\text{fts}(a)) &= \text{fts}(a) & \text{if } a \notin H & \text{DATD1} \\
\partial_H(\text{fts}(a)) &= \text{fts}(\delta) & \text{if } a \in H & \text{DATD2}
\end{align*}
\]

**TABLE 14.** ACP\(_{\text{dat}}\) = PA\(_{\text{dat}}\) - M1 + ACP\(_{\delta}^*\) + DATC, DATCM5-7, DRTD1,2, DAT11-16.
5. PARAMETRIC TIME.

In this section we will integrate the absolute time and the relative time approach. Our aim is to present a finite axiomatization, that allows an elimination theorem. As a consequence, we can expand expressions like \( \text{cts}(a) \parallel \text{fts}(b) \), \( \text{cts}(a) \parallel (\text{fts}(b) + 1) \). In [BAB92], we used a different approach, using variable binding, that did not allow a finite axiomatization.

We introduce two new operators: \( \mathcal{O} \), the (relative) time spectrum combinator (comparable to the push operator on stacks or the cons operator on lists), and \( \mu \), the spectrum tail operator (comparable to a pop, tail or rest operator). The absolute value operator introduced in 4.1 can also be called the spectrum head operator. \( P \mathcal{O} Q \) is a process that when initialised in the first time slice behaves as \( P \); when initialised in slice \( n+1 \) its behaviour is determined by \( Q \) as follows: initialise in slice \( n \) thereafter apply \( \mathcal{O} \). \( \mu(X) \) computes a process such that \( X = |X| \mathcal{O} \mu(X) \). For a parametric discrete time process we have the time spectrum sequence \( |X|, |\mu(X)|, |\mu^2(X)|, ... \). For each infinite sequence \( (P_n)_{n \in \mathbb{N}} \) one may imagine a process \( P \) with \( |\mu^n(P)| = P_n \) though not all such \( P \) can be finitely expressed.

5.1 BASIC PROCESS ALGEBRA.

We put together the theories BPA_{\text{dat}} and BPA_{\text{dat}}. We have presented all axioms in such a way, that they remain valid on the extended syntax. In one case, DAT7, we needed the absolute value operator, that characterizes absolute timing processes. On the other hand, the spectrum tail operator characterizes relative timing processes. The theory BPA_{\text{dat}} unites BPA_{\text{dat}} and BPA_{\text{dat}} and the additional axioms AV7-
9, ST1-7, SC1-3. BPA_{dpt} adds constants \text{AU}\{\delta\} and axioms A6A, A7, AV6, ADRT, ADAT. Notice that ST6,7 imply \mu(a) = a.

\[
\begin{align*}
\sigma_{\text{abs}}(X) &= \sigma_{\text{abs}}(|XI|) & \text{AV7} & \mu(\delta) = \delta & \text{ST1} \\
\mu_{\text{cts}}(a) &= \mu_{\text{cts}}(a) & \text{AV8} & \mu(\mu_{\text{cts}}(a)) = \mu_{\text{cts}}(a) & \text{ST2} \\
|\sigma_{\text{rel}}(X)| &= \sigma_{\text{abs}}(|\mu(X)|) & \text{AV9} & \mu(X + Y) = \mu(X) + \mu(Y) & \text{ST3} \\
& & & \mu(X \cdot Y) = \mu(X) \cdot \mu(Y) & \text{ST4} \\
|X \circ Y| &= |XI| & \text{SC1} & \mu(\sigma_{\text{rel}}(X)) = \sigma_{\text{rel}}(\mu(X)) & \text{ST5} \\
\mu(X \circ Y) &= Y & \text{SC2} & \mu(\mu_{\text{cts}}(a)) = \delta & \text{ST6} \\
X_0 Y &= |XI| \circ \mu(X) & \text{SC3} & \mu(\sigma_{\text{abs}}(X)) = |XI| & \text{ST7}
\end{align*}
\]

Table 16. BPA_{dpt} = BPA_{dat} + BPA_{dat} + AV7-9, ST1-7, SC1-3.

Note that the three axioms DAT5-7 can be replaced by \sigma_{\text{abs}}(X) \cdot Y = \sigma_{\text{abs}}(X \cdot \mu(Y)). We can define:

- **P** is an absolute time process iff BPA_{dpt} \vdash P = |PI|
- **P** is a relative time process iff BPA_{dpt} \vdash P = \mu(P).

We see that BPA^\bullet constitutes the intersection of the absolute time and relative time theories.

### 5.2 Basic Form.

We claim that each BPA_{dpt} (resp. BPA_{dpt}) process expression can be written in the form

\[
P_1 \circ P_2 \circ \ldots \circ P_n \circ Q
\]

(we omit brackets, using the convention that \circ associates to the right), such that each P_i is a BPA_{dat}-basic term (resp. BPA_{dat}-basic term) and Q is a BPA_{dat}-basic term (resp. BPA_{dat}-basic term).

The way we achieve this is by writing

\[
X = |XI| \circ |\mu(X)| \circ \ldots \circ |\mu^n(X)| \circ \mu^{n+1}(X).
\]

Now one can reduce each |\mu^n(X)| to a BPA_{dat}-basic term and if n is sufficiently large, we can write \mu^{n+1}(X) without any \sigma_{\text{abs}} or \mu_{\text{cts}}(a) using ST1-7, so will be in the relative time signature.

We call (|XI|, |\mu(X)|, |\mu^2(X)|, \ldots) the time spectrum expansion sequence (TSS) of X. The nth component is called \text{tss}_n(X).

The Parametric Time Principle is the principle that processes are equal if they have the same TSS. In this way, each model for absolute time processes induces a model for parametric time processes. We can prove that our axiomatisation is complete for this model.

Some examples of time spectrum sequences:

- The time spectrum sequence of \mu_{\text{cts}}(a) is (\text{fts}(a), \text{fts}(a), \text{fts}(a), \ldots)
- The time spectrum sequence of \text{fts}(a) is (\text{fts}(a), \delta, \delta, \ldots)
- The time spectrum sequence of \sigma_{\text{abs}}(\text{fts}(a)) is (\sigma_{\text{abs}}(\text{fts}(a)), \text{fts}(a), \delta, \delta, \ldots).

The nth component \text{tss}_n distributes over +, \circ, ||, _\text{cts}, _\text{fts}, _\text{fts}. Furthermore \text{tss}_n(X \cdot Y) = \text{tss}_n(X) \cdot \mu^{n}(Y).

### 5.3 Example.

\[
\sigma_{\text{rel}}(\text{fts}(a)) = |\sigma_{\text{rel}}(\text{fts}(a))| \circ \mu(\sigma_{\text{rel}}(\text{fts}(a))) = \sigma_{\text{abs}}(|\mu(\text{fts}(a))|) \circ \sigma_{\text{rel}}(\mu(\text{fts}(a))) = \sigma_{\text{abs}}(\delta) \circ \sigma_{\text{rel}}(\delta) = \text{fts}(\delta) \circ \text{cts}(\delta) \quad \text{(this is the basic form)}
\]
5.4 PARALLEL COMPOSITION.

The extension to PA or ACP is straightforward. The theory PA² unites BPA², PA², and PA² and the additional axioms AV10,11, ST8,9 of Table 17. PA² unites BPA², PA², and PA² and AV10,11, ST8,9. ACP² unites BPA², ACP², and ACP² and AV10-13, ST8-11. ACP² unites BPA², ACP², and ACP² and AV10-13, ST8-11.

\[ \begin{align*}
|X|I|Y| &= |X|I|Y| \quad AV10 \\
|X||Y| &= |X||Y| \quad AV11 \\
|I|Y| &= |I|X| |I|Y| \\
|dH(X)| &= dH(|X|) \quad AV13
\end{align*} \]

TABLE 17. Extension to PA² and ACP².

We can again achieve the basic form of 6.2. As a result, we achieve elimination, soundness and completeness. An example of the elimination of the merge (no communication): 

\[
(O_{abs}(fts(a)) + O_{rel}(cts(b))) \parallel cts(c) = fts(c) \cdot O_{abs}(fts(a) \cdot O_{abs}(fts(b))) \cdot O cts(c) \cdot O_{rel}(cts(b)).
\]

5.5 EXTENSIONALITY.

We formulate initialisation in an arbitrary time slice following [BAB92] by introducing natural numbers with 0 and successor (succ). \( n \gg X \) denotes the behavior of \( X \) upon initialisation at absolute time \( n \).

We have the following defining equations. As is usual, we write \( n+1 \) for \( \text{succ}(n) \), \( n+2 \) for \( \text{succ}(\text{succ}(n)) \) and so on.

\[ \begin{align*}
0 \gg X &= |X|0 \\
n+1 \gg X &= O_{abs}(n \gg \mu(X)).
\end{align*} \]

Now we have the following rule expressing extensionality for parametric discrete time:

\[
\forall n, n \gg P = n \gg Q \quad \text{EPDT.}
\]

It is the intended meaning of (parametric) discrete time process algebra that EPDT is satisfied. We note that with the use of EPDT one can prove elimination and completeness in the case of ACP² without the use of \( \mu \). Following [BAB93] one may use the following notation: if \( F \) is a process expression depending on \( x \), then \( \forall_d x. F \) is the parametric time process defined by \( n \gg (\forall_d x. F) = n \gg F[n/x] \). We call \( \forall_d x. F \) the initial abstraction operator for discrete time.

5.6 PARAMETRIC TIME CONDITIONS.

Another way to denote extensionality is obtained if we introduce parametric time conditions. We consider the following syntax.

- sort \( B_{par} \) sort of parametric time Booleans
- constants true, false standard Booleans
- functions \( \neg : B_{par} \rightarrow B_{par} \) negation
  \( \land, \lor : B_{par} \times B_{par} \rightarrow B_{par} \) conjunction, disjunction
  \( I, I : B_{par} \rightarrow B_{par} \) spectrum head
\[ \mu : B_{\text{par}} \rightarrow B_{\text{par}} \quad \text{spectrum tail} \]
\[ \Theta : B_{\text{par}} \times B_{\text{par}} \rightarrow B_{\text{par}} \quad \text{spectrum combinator}. \]

We have the axioms shown in Table 18. They constitute the theory $BOOL_{\text{dpt}}$. We link parametric time booleans to parametric time processes by means of the conditional operator $\langle \cdot, \cdot \rangle$. \( X \triangleleft b \triangleright Y \) represents if \( b \) then \( X \) else \( Y \) (notation taken from [HHJ+87]).

\begin{align*}
\neg \text{true} &= \text{false} & \text{BOOL1} & \ b \lor c &= c \lor b & \text{BOOL4} \\
\neg \text{false} &= \text{true} & \text{BOOL2} & \ true \lor b &= \text{true} & \text{BOOL5} \\
\neg \neg b &= b & \text{BOOL3} & \ false \lor b &= b & \text{BOOL6} \\
X \triangleleft \text{true} \triangleright Y &= X & \text{BOOL7} & \ b \land c &= \neg (\neg b \lor \neg c) & \text{BOOL9} \\
X \triangleleft \text{false} \triangleright Y &= Y & \text{BOOL8} & \ \\
\neg \text{true} &= \text{false} & \text{BAV1} & \ \mu(\text{true}) &= \text{true} & \text{BST1} \\
\neg \text{false} &= \text{true} & \text{BAV2} & \ \mu(\text{false}) &= \text{false} & \text{BST2} \\
\neg (\neg b) &= \neg b & \text{BAV3} & \ \mu(\neg b) &= \neg \mu(b) & \text{BST3} \\
b \lor c &= \neg b \lor \neg c & \text{BAV4} & \ \mu(b \lor c) &= \mu(b) \lor \mu(c) & \text{BST4} \\
X \triangleleft b \triangleright Y \Leftrightarrow X \triangleright Y & \text{BAV5} & \ \mu(X \triangleleft b \triangleright Y) &= \mu(X) \triangleleft \mu(b) \triangleright \mu(Y) & \text{BST5} \\
b \triangleright c &= b \triangleright c & \text{BSC1} & \ \mu(b \triangleright c) &= c & \text{BSC2} \\
b = b \triangleright c &= \neg (b \triangleright c) & \text{BSC3} & \text{BSC5} \\
(\neg (b \triangleright c) &= (b \triangleright \neg c) & \text{BSC4} & \text{BSC5} \\
\neg (b \triangleright c) &= \neg (b \triangleright c) & \text{BSC5} \end{align*}

Table 18. $BOOL_{dpt} = BOOL1-9, BAV1-5, BST1-5, BSC1-5$.

Using this syntax, we can define the parametric time booleans $\text{sl}(n)$ (true iff the current time slice is the \( n \)th slice), $\text{sl}_\triangleright(n)$ (true iff the current slice is larger than \( n \)) and the initialisation operator as follows:

\begin{align*}
\text{sl}(0) &= \text{false} & \text{sl}(1) &= \text{true} & \text{sl}(n+2) &= \text{false} & \text{sl}(n+1) \\
\text{sl}_\triangleright(0) &= \text{true} & \text{sl}_\triangleright(n+1) &= \text{false} & \text{sl}_\triangleright(n) \\
0 \triangleright b &= \text{label} & n+1 \triangleright b &= n \triangleright \mu(b). \\
\end{align*}

It follows that $n \triangleright (X \triangleleft b \triangleright Y) = n \triangleright X \triangleright n \triangleright b \triangleright n \triangleright Y$. A useful abbreviation is $b :\rightarrow X$ for $X \triangleleft b \triangleright \delta$. Now extensionality can be written as the time spectrum expansion $TSE$ as follows:

\[ X = \sum_{n=0}^{\infty} \text{sl}(n+1) :\rightarrow (n \triangleright X) \quad \text{TSE}, \]

and approximated with a finite sum, $TSE_k$:

\[ X = \sum_{n=0}^{k-1} \text{sl}(n+1) :\rightarrow (n \triangleright X) + \text{sl}_\triangleright(k) :\rightarrow X. \]

We call the last summand $\text{sl}_\triangleright(k) :\rightarrow X$ the residue of $X$ after slice $k$. We notice that for $X$ a closed expression in the syntax of $ACP_{dpt}$, from some $k$ onward the residue is (extensionally) equal to a process which is the residue of a relative time process. This allows yet another way to obtain basic forms and elimination results for parametric time processes.
An interesting additional operator is the operator $\psi$ defined by equations

$$
\psi(\text{true } \circledast X) = \text{false } \circledast \text{true } \circledast \psi(X), \\
\psi(\text{false } \circledast X) = \text{false } \circledast \psi(X).
$$

Using this operator, all eventually periodic time conditions can be expressed. This is useful in the representation of regular parametric time processes.

5.7 PROJECTION.

We can extend the definition of the absolute time projection operators to parametric time processes by adding to the axioms in 4.5 (table I I) the axiom $\pi^{\text{abs}}_{n+1}(X \circledast Y) = \pi^{\text{abs}}_{n+1}(|X|) \circledast \pi^{\text{abs}}_{n}(Y)$. For a relative time process $P$, the relationship between the relative time and absolute time projection operators is given by the axiom $n \gg \pi^{\text{rel}}_{k}(P) = \pi^{\text{abs}}_{n+k}(n \gg P)$.

5.8 RECURSION.

If $P$ is a closed process expression, then we may view $X = P \circledast X$ as a guarded equation in $X$. In many models, all guarded equations will have unique solutions. As an example, $\text{cts}(a)$ is the unique solution of $X = \text{fts}(a) \circledast X$. For each such equation, the solution $X$ is a relative time process (because $\mu(X) = X$). Further, we have $|X| = |P|$. We write $|P|_{\text{rel}}$ for the solution, so we have $|P|_{\text{rel}} = |P|_{\text{abs}} \circledast |P|_{\text{rel}}$. In this paper, we will not use the operator $|P|_{\text{rel}}$ and consequently, we will not consider axioms for it.

5.9 REGULAR PROCESSES.

In [BeK84b], finite state processes modulo (strong) bisimulation were studied in the context of ACP. The definitions given there can be easily extended to discrete time.

First of all, a relative time process is regular if its operational semantics as given in 3.2,7,8 yields a transition system which is bisimilar to a finite one. Regular relative time processes can be defined using linear systems of equations over $\delta$, $\text{cts}(a)$, $+$, $\sigma_{\text{rel}}$ and $\text{cts}(a) \cdot \_\text{rel}$, the prefix restriction of sequential composition. As an example, the linear equation $X = \text{cts}(a) + \sigma_{\text{rel}}(X)$ defines the process $a$.

Next, if $X$ is a regular relative time process, then $|X|$ is regular in absolute time. This is used as a definition, i.e. an absolute time process $X$ is regular iff for some regular relative time $Y$ we have $X = |Y|$. To define regular absolute time processes using systems of linear equations we need $|P|_{\text{rel}}$ and the detour via relative time.

Finally, a parametric time process is regular if its time spectrum sequence $(|\mu^{n}(X)|)_{n} \in \mathbb{N}$ is an eventually periodic sequence of regular processes.

5.10 REMARK.

Similar to the binary Kleene star operator with defining equation $X^* Y = X \cdot (X^* Y) + Y$ (see [BeBP94]) we can introduce iterated delay: $\sigma_{\text{rel}}^{*}(X) = \sigma_{\text{rel}}(\sigma_{\text{rel}}^{*}(X)) + X$. To give an example, we have $\sigma_{\text{rel}}^{*}(\text{cts}(a)) = a$. A very interesting area of work is to investigate axiomatisations for fragments of process algebra with iteration operators. Along the lines of [FoZ94], [FoK94] several results in the case of discrete time can be obtained.
6. RELATION TO EARLIER WORK.
The relation between the two-phase versions of discrete time presented here and the timestamped versions of [BAB92] is as follows:

\[ a(n+1) = \sigma_{\text{abs}}^n(fts(a)) \quad \quad \quad \quad \quad \quad \quad a(n+1) = \sigma_{\text{rel}}^n(cts(a)). \]

In turn, we can relate these to timestamped real time of [BAB91] as follows:

\[ a(n+1) = \int_{t\in(tn+1)} a(t) \quad \quad \quad \quad \quad \quad \quad a(n+1) = \int_{r\in[1],[n]+1} a(r) \]

Many extensions of ACP have been proposed, we mention: state operator, priority operator, process creation operators, unreliable communication primitives, iteration, early input prefix, process prefix. In all cases we find that it is completely straightforward to determine timed versions in absolute, relative and parametric time.

7. COMPARISON WITH RELATED WORK.
Other discrete time process algebras that we encountered all use relative timing and are two-phase. None of them is related to a time free theory. Therefore, we limit ourselves to comparing other theories to \( \text{ACP}_{\text{drt}} \).

In \( \text{BPA}_{\text{drt}} \), we combine a unary operator for (unit) delay like in TCCS [MoT89], time factorisation as in ATP [NIS94], TCCS [MoT89], TPCCS [Han91] and TPL [HeR90], the interpretation of + as the weak choice of TCCS, but we notice a significant difference with TPL [HeR90] because visible actions cannot idle (here, we follow TCCS, ACP, ATP and ATP).

Our definition of free merge in \( \text{PA}_{\text{drt}} \) is similar to the definitions in TCCS, ATP, \( \text{ACP}_{\text{t}} \) and TPL (the correspondence with the TPL definition holds in the absence of communication; in the presence of communication TPL's merge will give priority to internal communications). So it follows that without communication the relation between merge and discrete time is not controversial. We depart from ATP by allowing time stops. Indeed, in our setting \( \sigma_{\text{rel}}(cts(a)) \parallel cts(b) \) is unable to perform a \( \sigma \)-step.

The merge of \( \text{ACP}_{\text{drt}} \) works just as the merge of TCCS [MoT89] (taking weak choice of TCCS for +). It is also equivalent to the merge of ATP. It differs from merge in [GR090a], set up in \( \text{ACP}_{\text{t}} \). In fact, \( \text{ACP}_{\text{t}} \) contains axioms \( \sigma \ll x = \delta \) and \( \sigma x \ll y = \delta \). Our main objection against these axioms (which occur in ATP as well) is that they render it impossible to injectively embed \( \text{ACP}_{\text{t}} \) into \( \text{ACP}_{\text{p}} \) of [BAB91] or its extension \( \text{ACP}_{\text{pV}} \) of [BAB93].

It follows that the only disagreement in the literature is about the proper semantics of left merge and communication merge. Therefore, our choice for the merge operator itself seems reasonably well motivated.

8. CONCLUDING REMARKS.
We have proposed a theory on discrete time process algebra that extends ACP with the features of ATP in a setting consistent with that of \( \text{ACP}_{\text{p}}, \text{ACP}_{\text{pV}}, \text{ACP}_{\text{pV}} \). Many options for further work remain. We mention some:
i. All extensions of ACP that have been developed can be integrated in the discrete time setting, such as state operators, process creation, signals, priorities, asynchronous communication (see [BAW90] for these ACP extensions).

ii. The analysis of ACPdpp as a term rewriting system may be worth attention.

iii. Like in the case of ACP, many more models than the bisimulation model are possible. We sketched a projective limit model. Investigation of other models such as failures, ready, trace models can be worth while.

REFERENCES.


Computing Science Reports

In this series appeared:

93/01 R. van Geldrop
Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.

93/02 T. Verhoef
A continuous version of the Prisoner's Dilemma, p. 17

93/03 T. Verhoef
Quicksort for linked lists, p. 8.

93/04 E.H.L. Aarts
Deterministic and randomized local search, p. 78.

93/05 I.C.M. Baeten
A congruence theorem for structured operational semantics with predicates, p. 18.

93/06 J.P. Veltkamp
On the unavoidability of metastable behaviour, p. 29

93/07 P.D. Moerland
Exercises in Multiprogramming, p. 97

93/08 J. Verhoosel
A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.

93/09 K.M. van Hee

93/10 K.M. van Hee
Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.

93/11 K.M. van Hee

93/12 K.M. van Hee

93/13 K.M. van Hee

93/14 J.C.M. Baeten

93/15 J.C.M. Baeten

93/16 H. Schepers
A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27

93/17 D. Alstein
Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.

93/18 C. Verhoef
A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.

93/19 G-J. Houben
The Design of an Online Help Facility for ExSpect, p. 21.

93/20 F.S. de Boer
93/21 M. Codish, D. Dams, G. Filé, M. Bruynooghe


93/22 E. Poll

A Typechecker for Bijective Pure Type Systems, p. 28.

93/23 E. de Kogel

Relational Algebra and Equational Proofs, p. 23.

93/24 E. Poll and Paula Severi

Pure Type Systems with Definitions, p. 38.

93/25 H. Schepers and R. Gerth


93/26 W.M.P. van der Aalst

Multi-dimensional Petri nets, p. 25.

93/27 T. Kloks and D. Kratsch

Finding all minimal separators of a graph, p. 11.

93/28 F. Kamareddine and R. Nederpelt

A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.

93/29 R. Post and P. De Bra

GOLD, a Graph Oriented Language for Databases, p. 42.

93/30 J. Deogun, T. Kloks, D. Kratsch, H. Müller

On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31 W. Körver

Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.

93/32 H. ten Eikelder and H. van Geldrop


93/33 L. Loyens and J. Moonen

ILLIAS, a sequential language for parallel matrix computations, p. 20.

93/34 J.C.M. Baeten and J.A. Bergstra

Real Time Process Algebra with Infinitesimals, p. 39.

93/35 W. Ferrer and P. Severi

Abstract Reduction and Topology, p. 28.

93/36 J.C.M. Baeten and J.A. Bergstra

Non Interleaving Process Algebra, p. 17.

93/37 J. Brunekreef, J-P. Katoen, R. Koymans, S. Mauw

Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.

93/38 C. Verhoef

A general conservative extension theorem in process algebra, p. 17.

93/39 W.P.M. Nuijten, E.H.L. Aarts, D.A.A. van Erp Taalman Kip, K.M. van Hee

Job Shop Scheduling by Constraint Satisfaction, p. 22.
<table>
<thead>
<tr>
<th>Citation</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/41</td>
<td>A. Bijlsma</td>
<td>Temporal operators viewed as predicate transformers, p. 11.</td>
</tr>
<tr>
<td>93/42</td>
<td>P.M.P. Rambags</td>
<td>Automatic Verification of Regular Protocols in P/T Nets, p. 23.</td>
</tr>
<tr>
<td>93/43</td>
<td>B.W. Watson</td>
<td>A taxonomy of finite automata construction algorithms, p. 87.</td>
</tr>
<tr>
<td>93/44</td>
<td>B.W. Watson</td>
<td>A taxonomy of finite automata minimization algorithms, p. 23.</td>
</tr>
<tr>
<td>93/48</td>
<td>R. Gerth</td>
<td>Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.</td>
</tr>
<tr>
<td>94/01</td>
<td>P. America, M. van der Kammen, R.P. Nederpelt, O.S. van Roosmalen, H.C.M. de Swart</td>
<td>The object-oriented paradigm, p. 28.</td>
</tr>
<tr>
<td>94/02</td>
<td>F. Kamareddine, R.P. Nederpelt</td>
<td>Canonical typing and Π-Conversion, p. 51.</td>
</tr>
<tr>
<td>94/04</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.</td>
</tr>
<tr>
<td>94/06</td>
<td>T. Basten, T. Kunz, J. Black, M. Coffin, D. Taylor</td>
<td>Time and the Order of Abstract Events in Distributed Computations, p. 29.</td>
</tr>
<tr>
<td>94/08</td>
<td>O.S. van Roosmalen</td>
<td>A Hierarchical Diagrammatic Representation of Class Structure, p. 22.</td>
</tr>
<tr>
<td>Volume</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>94/09</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Process Algebra with Partial Choice, p. 16.</td>
</tr>
<tr>
<td>94/10</td>
<td>T. Verhoef</td>
<td>The testing Paradigm Applied to Network Structure, p. 31.</td>
</tr>
<tr>
<td>94/13</td>
<td>R. Seljée</td>
<td>A New Method for Integrity Constraint checking in Deductive Databases, p. 34.</td>
</tr>
<tr>
<td>94/14</td>
<td>W. Peremans</td>
<td>Ups and Downs of Type Theory, p. 9.</td>
</tr>
<tr>
<td>94/16</td>
<td>R.C. Backhouse, H. Doornbos</td>
<td>Mathematical Induction Made Calculational, p. 36.</td>
</tr>
<tr>
<td>94/18</td>
<td>F. Kamareddine, R. Nederpelt</td>
<td>Refining Reduction in the Lambda Calculus, p. 15.</td>
</tr>
<tr>
<td>94/19</td>
<td>B.W. Watson</td>
<td>The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.</td>
</tr>
<tr>
<td>94/20</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>Beyond $\beta$-Reduction in Church's $\lambda\alpha$, p. 22.</td>
</tr>
<tr>
<td>94/22</td>
<td>B.W. Watson</td>
<td>The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.</td>
</tr>
<tr>
<td>94/23</td>
<td>S. Mauw and M.A. Reniers</td>
<td>An algebraic semantics of Message Sequence Charts, p. 43.</td>
</tr>
<tr>
<td>94/24</td>
<td>D. Dams, O. Grumberg, R. Gerth</td>
<td>Abstract Interpretation of Reactive Systems: Abstractions Preserving $\mbox{vCTL}^<em>$, $\mbox{-vCTL}^</em>$ and $\mbox{CTL}^*$, p. 28.</td>
</tr>
<tr>
<td>94/25</td>
<td>T. Kloks</td>
<td>$K_{13}$-free and $W_4$-free graphs, p. 10.</td>
</tr>
<tr>
<td>Paper Number</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>94/29</td>
<td>J. Hooman</td>
<td>Correctness of Real Time Systems by Construction, p. 22.</td>
</tr>
<tr>
<td>94/31</td>
<td>B.W. Watson, R.E. Watson</td>
<td>A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.</td>
</tr>
<tr>
<td>94/33</td>
<td>T. Laan</td>
<td>A formalization of the Ramified Type Theory, p. 40.</td>
</tr>
<tr>
<td>94/34</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>The Barendregt Cube with Definitions and Generalised Reduction, p. 37.</td>
</tr>
<tr>
<td>94/35</td>
<td>J.C.M. Baeten, S. Mauw</td>
<td>Delayed choice: an operator for joining Message Sequence Charts, p. 15.</td>
</tr>
<tr>
<td>94/36</td>
<td>F. Kamareddine, R. Nederpelt</td>
<td>Canonical typing and II-conversion in the Barendregt Cube, p. 19.</td>
</tr>
<tr>
<td>94/38</td>
<td>A. Bijlsma, C.S. Scholten</td>
<td>Point-Free substitution, p. 10.</td>
</tr>
<tr>
<td>94/39</td>
<td>A. Blokhuis, T. Kloks</td>
<td>On the equivalence covering number of splitgraphs, p. 4.</td>
</tr>
<tr>
<td>94/40</td>
<td>D. Alstein</td>
<td>Distributed Consensus and Hard Real-Time Systems, p. 34.</td>
</tr>
<tr>
<td>94/41</td>
<td>T. Kloks, D. Kratsch</td>
<td>Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.</td>
</tr>
<tr>
<td>94/43</td>
<td>R.C. Backhouse, M. Bijnerveld</td>
<td>Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.</td>
</tr>
<tr>
<td>94/45</td>
<td>G.J. Houben</td>
<td>Tutorial voor de ExSpect-bibliotheek voor &quot;Administratieve Logistiek&quot;, p. 43.</td>
</tr>
<tr>
<td>94/46</td>
<td>R. Bloo, F. Kamareddine, R. Nederpelt</td>
<td>The λ-cube with classes of terms modulo conversion, p. 16.</td>
</tr>
</tbody>
</table>
94/47  R. Bloo  
F. Kamareddine  
R. Nederpelt  

On II-conversion in Type Theory, p. 12.

94/48  Mathematics of Program Construction Group  

Fixed-Point Calculus, p. 11.

94/49  J.C.M. Baeten  
J.A. Bergstra  

Process Algebra with Propositional Signals, p. 25.

94/50  H. Geuvers  

A short and flexible proof of Strong Normalization for the Calculus of Constructions, p. 27.

94/51  T. Kloks  
D. Kratsch  
H. Müller  

Listing simplicial vertices and recognizing diamond-free graphs, p. 4.

94/52  W. Penczek  
R. Kuiper  

Traces and Logic, p. 81

94/53  R. Gerth  
R. Kuiper  
D. Peled  
W. Penczek  

A Partial Order Approach to Branching Time Logic Model Checking, p. 20.

95/01  J.J. Lukkien  

The Construction of a small Communication Library, p. 16.

95/02  M. Bezem  
R. Bol  
J.F. Groote  

Formalizing Process Algebraic Verifications in the Calculus of Constructions, p. 49.

95/03  J.C.M. Baeten  
C. Verhoef  

Concrete process algebra, p. 134.

95/04

95/05  P. Severi  

A Type Inference Algorithm for Pure Type Systems, p. 20.

95/06  T.W.M. Vossen  
M.G.A. Vehoeven  
H.M.M. ten Eikelder  
E.H.L. Aarts  

A Quantitative Analysis of Iterated Local Search, p. 23.

95/07  G.A.M. de Bruyn  
O.S. van Roosmalen  

Drawing Execution Graphs by Parsing, p. 10.

95/08  R. Bloo  

Preservation of Strong Normalisation for Explicit Substitution, p. 12.