Process algebra with autonomous actions

Citation for published version (APA):

Document status and date:
Published: 01/01/1996

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 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: 11. Nov. 2019
Process Algebra with Autonomous Actions

by

M. Voorhoeve and T. Basten

96/01

ISSN 0926-4515

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

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

Computing Science Report 96/01
Eindhoven, February 1996
Process Algebra with Autonomous Actions

Marc Voorhoeve and Twan Basten
Eindhoven University of Technology
e-mail: (wsinmarc.t basten)@win.tue.nl

Abstract

This paper introduces autonomous observable actions into process algebra. These actions can be observed but cannot be controlled by an environment. The proposed extension of ACP allows verifications without silent steps and fairness assumptions. The inclusion of inequalities makes it possible to verify that an implementation satisfies a given specification, with the specification indicating exactly where the implementation may reduce nondeterminism.

1 Introduction

Process algebra studies processes that are built from actions and operators. Two kinds of actions have been extensively studied: the observable ones and the unobservable (internal or silent) action τ. An observable action can occur within a process if the process is ready to perform the action and some controlling environment allows it to occur. In contrast, the silent action τ may occur without any consent. In ACP, the process algebra studied in this paper, the operators by which processes can be controlled from an environment are encapsulation and communication. Observable actions can be disabled by an environment by means of encapsulation; silent actions cannot be disabled. Similarly, observable actions are capable of communicating with actions from an environment, whereas τ is not.

This paper presents autonomous actions, which are observable, but not controllable by an environment. So they share with τ the property that they can neither be encapsulated nor participate in a communication.

The difference between autonomous actions and τ is that autonomous actions can be observed, whereas τ cannot. The fact that τ is unobservable means that an environment can only detect an occurrence of τ if the process possesses less options for continuation after the occurrence than before. This is reflected by equations for processes, such as the branching equations B1 and B2 in ACP that allow to eliminate silent actions. No such elimination is possible for autonomous actions, as each occurrence can be observed.

A process that is capable of performing an initial autonomous action is called unstable. A process in which no initial autonomous action can occur is called stable, as an environment can stop its progress by encapsulating its initial actions. A choice involving an unstable process is nondeterministic, since an environment cannot prevent the autonomous action from occurring. We call this kind of nondeterminism unstable nondeterminism. Another kind of nondeterminism arises when several non-autonomous actions with the same visible effect can occur, leading to different subsequent processes. We call this stable nondeterminism.

Instead of only considering process equivalence, a partial order ≤ on processes is introduced. If processes p and q satisfy p ≤ q, then either both processes are equal, or both p and q are unstable and the behaviour of p is a subset of the behaviour of q. Since there must exist an autonomous action that both p and q can perform, and since this action cannot be controlled, an observer, when confronted with p, cannot be sure that he is not dealing with q instead. So ≤ is based on reduction of unstable nondeterminism.
Partial order algebras that are based on reduction of nondeterminism are important for software engineering. Often, design decisions are made that reduce nondeterminism in a specification. The advantage of the approach in this paper is that in the specification one can distinguish unstable nondeterminism that may be reduced and stable nondeterminism that may not.

The main inspiration of the present paper is [BaBe94], where CSP-like choice operators [Hoa85] are introduced in ACP. The "partial bisimulation" preorder on processes from [BaBe94] that is also used in the present paper is related to the preorders in [Abr87] and [Wal90] that deal with the notion of divergence: a possibly infinite chain of autonomous actions (in their case silent actions τ).

The reason for introducing CSP-like choice operators in ACP in [BaBe94] is that CSP distinguishes between deterministic and nondeterministic choice. Autonomous actions offer an alternative to nondeterministic choice operators, with the advantage of considerably simpler equations and models. Also, in this paper axioms and proof rules are given for reasoning algebraically with infinite processes, whereas the reasoning in [BaBe94] (and CSP for that matter) is model-based.

The authors wish to thank Jos Baeten and Pedro D'Argenio for their suggestions and corrections. Jan Bergstra is thanked for sharing with us his insights regarding communication and autonomous behaviour.

2 The Basic Algebra

2.1 Signature and Axioms

Our basic algebra BPAaa≤ is an extension of BPAδ: basic process algebra with inaction (c.f. [BaWe90]). The signature and axioms are given in Table 1. First, in the header, the sort of atoms (A) is declared, which is a parameter of the theory. Each atom is an action. An overlining operator creates from an atom an autonomous action. The sort F of autonomous actions is thus equal to \{a: A • δ\}.

The set A \cup F of all actions is denoted AC. Note that AC is only an abbreviation and not a sort. The actions a and \overline{a} have the same visible effect and only differ in the way they can be controlled by an environment. The sort P of processes is a supersort of both A and F. Sorts and their subsort relation are introduced in the first table entry.

The operator → as well as the standard BPA operators • • and • • • and the constant δ are introduced in the second table entry, followed by the declaration of some variables and the axioms of the theory. As usual, the sequencing operator (·) has priority over choice (+) and may be omitted. The axiom system consists of equations and inequalities, giving rise to a partial order algebra (c.f. [Hen88]), which means that in reasoning with inequalities, one is allowed to use reflexivity, transitivity, antisymmetry, closedness under contexts and substitutivity. Note that, by antisymmetry, for two terms x and y, x = y iff x ≤ y and y ≤ x.

The I axioms (for "inequality") state that unstable nondeterminism may be reduced in an implementation. The algebra BPAaa is obtained from BPAaa≤ by removing these inequalities from the axioms in Table 1. Note that BPAaa with atoms A equals BPA δ with atoms AC.

The BPAaa≤ terms can be normalized. We inductively define the set B of basic terms as follows.

\{δ\} \cup AC ≤ B,∀t:B; e:AC • e·t ∈ B,∀s,t:B • s + t ∈ B.

Property 1 For every closed term p : P, there exists a t : B such that BPAaa ⊢ p = t (and thus BPAaa≤ ⊢ p = t).

We use the Z-style notation \{a : A • f(a)\} instead of \{f(a) | a ∈ A\} throughout the paper.
Proof Standard term rewriting, using A3 . . . A7 as rewrite rules from left to right (c.f. [BaVe95]). □

Modulo axioms A1 and A2, and using A6, a basic term can be represented as $\sum_i a_i + \sum_j b_j \cdot t_j$, where $I, J$ are finite index sets, the $a_i$ and $b_j$ actions and the $t_j$ again basic terms. An empty sum is equal to $\delta$. By A3, it may be assumed that all summands are different.

Example The theory $\text{BPAaa}^{\leq}$ may be used to demonstrate equivalence of the following processes that describe login procedures. The first process $L_1$ prompts for a user identification and, having received one, prompts for a password. After having checked the combination, access is granted or refused. The user of the process perceives the prompting, granting and refusing as autonomous actions he cannot influence. The other actions are interactions between the process and the user. So $L_1$ can be described as $\overline{pui} \cdot \overline{ui} \cdot M_1$, where $M_1 = \overline{ppw} \cdot \overline{pw} (\overline{ok} + \overline{rej})$.

The second process $L_2$ checks the user id before prompting for the password. If the user id is found correct, it proceeds as above; if not, it will still prompt for the password and reject regardless of the answer. The process $L_2$ is thus described as $\overline{pui} \cdot \overline{ui} (M_1 + M_2)$, with $M_1$ as above and $M_2 = \overline{ppw} \cdot \overline{pw} \cdot \overline{rej}$.

In $\text{BPAaa}^{\leq}$, both login processes are equal, because we can demonstrate equality of $M_1 + M_2$ and $M_1$. Since $M_1$ is unstable, it follows from I2 that $M_1 \leq M_1 + M_2$. It follows from I1 that $\overline{ok} + \overline{rej} \geq \overline{rej}$. Closedness under contexts yields $M_1 \geq M_2$, which implies by A3 that $M_1 = M_1 + M_2$. Thus $M_1 = M_1 + M_2$ by antisymmetry. As a result, $L_1 = L_2$.

The equality of $L_1$ and $L_2$ conforms to intuition and is desirable as well, since potential intruders are left uninformed whether they "guessed" a correct user id. At the end of Section 2.3, we will argue that equality of $L_1$ and $L_2$ cannot be proved without the inequalities I1 and I2.

2.2 Process Theory

Let $AC$ be a set of actions partitioned into disjoint subsets $A$ (non-autonomous actions) and $\mathcal{F}$ (autonomous actions). A process space is a pair $(\mathcal{P}, \rightarrow)$, where $\mathcal{P}$ is a set of processes and $\rightarrow: \mathcal{P}(\mathcal{P} \times AC \times \mathcal{P} \cup \{\|\})$ a ternary relation, the event relation. The event $p \xrightarrow{a} \| \rightarrow \mathcal{P}$ represents successful termination of $p$ with action $a$. A stable process cannot perform any autonomous actions in the first step. The set $\mathcal{S}$ of stable processes is thus defined as $\mathcal{S} = \{p: \mathcal{P} | f: \mathcal{F} ; p': \mathcal{P} \cup \{\|\} \rightarrow p'\}$. We define a class of relations called partial bisimulations. A simulation $R$ is a partial bisimulation if $R^{-1}$ is a local simulation for stable processes.
Definition 1 A relation $R : \mathcal{P}(\mathcal{P} \cup \{\top\} \times \mathcal{P} \cup \{\top\})$ is called a partial bisimulation (PBS) iff it satisfies the following requirements. For $p, p', q, q' : \mathcal{P} \cup \{\top\}$ and $e : \mathcal{AC}$,

\begin{enumerate}
  \item $pRq \Rightarrow (p = \top \iff q = \top) \land (pRq \land p \xrightarrow{e} p' \Rightarrow \exists q' : \mathcal{P} \bullet (q \xrightarrow{e} q' \land p'Rq'))$.
  \item $(pRq \land p \in S \land q \xrightarrow{e} q') \Rightarrow \exists p' : \mathcal{P} \bullet (p \xrightarrow{e} p' \land p'Rq')$.
\end{enumerate}

Process $p : \mathcal{P}$ is said to implement $q : \mathcal{P}$ (notation $p \preceq q$) iff there exists a PBS $R$ such that $pRq$. The processes $p, q$ are said to be equivalent (notation $p \simeq q$) iff $p \preceq q$ and $q \preceq p$.

As usual, we can also define strong bisimulation on processes (see e.g. [BaWe90]). A strong bisimulation between processes $p$ and $q$ is a PBS relating both $p$ to $q$ and $q$ to $p$. If $\mathcal{F} = \emptyset$, each PBS is a strong bisimulation.

Property 2 The relation $\preceq$ is a preorder and, consequently, $\simeq$ is an equivalence relation.

Proof We have to show that $\preceq$ is reflexive and transitive. The identity relation on processes is a PBS, proving reflexivity. For transitivity, assume that $R, S$ are PBSs such that $pRq$ and $qSr$. Obviously, the relation composition $R \circ S$ is a simulation. If $p$ is stable, then $R^{-1}$ is a local simulation, so $q$ is stable and hence $S^{-1}$ is a local simulation. So $(R \circ S)^{-1}$ is a local simulation and $R \circ S$ a PBS. So $p \preceq r$.

With preorder relations like partial bisimulation, special care must be taken with respect to termination and nontermination of processes. It is undesirable that a terminating process can be implemented by a nonterminating one. Therefore, we shall define a termination property for processes. Intuitively, a process is called terminating if it has a path leading to $\top$ and it can only make transitions to terminating processes. A terminating process cannot deadlock, nor can it contain a cycle from which escape is impossible.

Definition 2 The set $\mathcal{AC}^*$ of process traces consists of finite sequences $e_1 \ldots e_n$ of actions, including the empty trace $\epsilon$. The reachability relation $\Rightarrow : \mathcal{P}(\{\top\} \times \mathcal{AC}^* \times \{\top\})$ is defined as the smallest relation (ordered by set inclusion) such that for all $p, p', p'' : \mathcal{P} \cup \{\top\}; e : \mathcal{AC}; \alpha : \mathcal{AC}^*$,

$p \Rightarrow p \land p \xrightarrow{e} p' \land p'' \xrightarrow{\alpha} p'' \Rightarrow p \xrightarrow{e} p''$.

The termination predicate for a process $p$ (notation $p \Downarrow$) is defined as follows.

$p \Downarrow \iff \forall p' : \mathcal{P}; \alpha : \mathcal{AC}^* \bullet (p \xrightarrow{e} p' \Rightarrow \exists \beta : \mathcal{AC}^* \bullet p' \xrightarrow{\beta} \top)$.

2.3 A Process Model for BPAaa≤

In order to construct a process model for BPAaa≤, we have to give interpretations $A, F$ and $\mathcal{P}$ respectively for the sorts $A, F$ and $\mathcal{P}$. Let the process space $\mathcal{P}$ be the set of closed BPAaa≤ terms. Let $A$ and $F$ simply be equal to $A$ and $F$ respectively. Obviously, these interpretations do satisfy the requirements $A \subseteq \mathcal{P}$ and $F \subseteq \mathcal{P}$. Let $\mathcal{AC} = A \cup F$. The relation $\Rightarrow$ is given as the smallest relation satisfying the requirements in Table 2. In this table, the variables $a : A; p, p', q : \mathcal{P}; e : \mathcal{AC}$ are used.

Our model will be the set of equivalence classes of $\mathcal{P}$ modulo PBS equivalence, denoted $\mathcal{P} / \simeq$. The following property means that $\mathcal{P} / \simeq$ with $\leq$ is a partial order algebra.

Property 3 The relation $\preceq$ is a precongruence w.r.t. the BPAaa≤ operators, i.e., for all $a, b : A$ such that $a \preceq b$, $\overline{a} \preceq \overline{b}$ and for all $p_1, p_2, q_1, q_2 : \mathcal{P}$ such that $p_1 \preceq p_2$ and $q_1 \preceq q_2$, $p_1 \cdot q_1 \preceq p_2 \cdot q_2$ and $p_1 + q_1 \preceq p_2 + q_2$.
For closed terms p and q, \( \text{BPAaa} \models p \leq q \) signifies that \( p \leq q \) can be derived from the axioms, whereas \( \mathcal{P}/\simeq \models p \leq q \) signifies that \( p \leq q \) is valid in the model \( \mathcal{P}/\simeq \). That is, \( \mathcal{P}/\simeq \models p \leq q \) if \( p \leq q \). The following theorem states soundness and completeness of \( \text{BPAaa} \) with respect to \( \mathcal{P}/\simeq \).

**Theorem 1** Let \( p, q \) be closed \( \text{BPAaa} \) terms. Then \( \text{BPAaa} \models p \leq q \) if \( \mathcal{P}/\simeq \models p \leq q \).

**Proof** It follows from Property 3 that it suffices to check the validity of the axioms to prove soundness. In each case, PBSs can be constructed easily. For the axioms A1 \ldots A7, even strong bisimulations can be constructed.

Completeness is more involved. Suppose \( p \leq q \). Then there exists a PBS \( R \) such that \( p R q \). We use induction on the number of symbols in \( p \) plus the number of symbols in \( q \). The basic case, where \( p \) and \( q \) each consist of a single symbol follows from the fact that \( R \) is a PBS such that \( p R q \). Processes \( p \) and \( q \) are either both equal to \( \delta \) or both consist of the same action. Hence, in the basic case, \( \text{BPAaa} \models p \leq q \).

In the induction step, from Property 1, we may assume that \( p \) and \( q \) are basic terms; so

\[
p = \sum_{i \in I} a_i + \sum_{j \in J} b_j \cdot s_j, \quad q = \sum_{k \in K} c_k + \sum_{l \in L} d_l \cdot t_l,
\]

where \( I, J, K \) and \( L \) are finite index sets, the \( a_i, b_j, c_k \) and \( d_l \) are actions from \( \text{AC} \) and the \( s_j \) and \( t_l \) are basic terms. We distinguish two cases.

First, assume that \( p \) is stable. Then \( R^{-1} \) is a local simulation. We may conclude from the derivation rules in Table 2 that

\[
\forall i: I \cdot \exists k: K \cdot a_i = c_k, \quad \forall j: J \cdot \exists l: L \cdot b_j = d_l \land s_j R t_l,
\]

\[
\forall k: K \cdot \exists i: I \cdot a_i = c_k, \quad \forall l: L \cdot \exists j: J \cdot d_l = b_j \land s_j R t_l.
\]

The induction hypothesis yields

\[
\forall j: J \cdot \exists l: L \cdot b_j = d_l \land \text{BPAaa} \models s_j \leq t_l.
\]

It follows immediately that \( \text{BPAaa} \models p \leq q \).

Second, assume that \( p \) is unstable. Since \( R \) is a simulation, we may conclude from the derivation rules in Table 2 that

\[
\forall i: I \cdot \exists k: K \cdot a_i = c_k, \quad \forall j: J \cdot \exists l: L \cdot b_j = d_l \land s_j R t_l.
\]

The induction hypothesis again yields

\[
\forall j: J \cdot \exists l: L \cdot b_j = d_l \land \text{BPAaa} \models s_j \leq t_l.
\]

Since \( p \) is unstable, one of its initial actions is autonomous. Using axioms I1 and I2, \( \text{BPAaa} \models p \leq q \) follows easily. \( \square \)

The set of equivalence classes of \( \mathcal{P} \) modulo strong bisimulation equivalence is a sound and complete model for \( \text{BPAaa} \). This result follows from the one-to-one correspondence between closed \( \text{BPAaa(A)} \) terms and closed \( \text{BPA}_{\text{A}}(\text{AC}) \) terms together with soundness and completeness of \( \text{BPA}_{\text{A}}(\text{AC}) \) for the set of equivalence classes of closed \( \text{BPA}_{\text{A}}(\text{AC}) \) terms modulo strong bisimulation equivalence. It means that all actions in \( \text{AC} \) are considered to be non-autonomous and thus all processes are considered to be stable. The consequence is that, as long as no inequality is used, two closed \( \text{BPAaa} \) terms that are
derivably equal are strongly bisimilar. The use of an inequality in a verification represents the reduction of unstable nondeterminism. It is important to keep track at which points this reduction occurs. This can be achieved by delaying the use of $I_1$ and $I_2$ until they are really needed, i.e., proceeding within BPAaa as long as possible.

At this point, we can also justify our claim that the equivalence of the login procedures $L_1$ and $L_2$ in the example of Section 2.1 cannot be proved without the inequalities $I_1$ and $I_2$. This follows immediately from the observations above and the fact that $L_1$ and $L_2$ are not bisimilar.

3 Iteration

In this section, an iteration operator, c.q. the binary Kleene star (BKS) is added to the basic theory. The additional signature and axioms, which have been taken from [BeBP94], are stated in Table 3.

<table>
<thead>
<tr>
<th>BPAaa$^\subseteq$ (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPAa$^\subseteq$ (A)</td>
</tr>
<tr>
<td>$\cdot$ : $P \times P \rightarrow P$</td>
</tr>
<tr>
<td>$x, y, z : P$</td>
</tr>
</tbody>
</table>
| $x(x^*y) + y = x^*y$ 
| $x^*(y \cdot z) = (x^*y)z$ 
| $x^*(y((x + y) \cdot z) + z) = (x + y)^*z$ |

Table 3: Axioms and SOS rules for the binary Kleene star

A process model is derived as follows. Again, let $A$ and $F$ be equal to $A$ and $F$. Recall that $AC = A \cup F$. Define $P$ as the set of closed BPA$^\subseteq$aa terms. The relation $\rightarrow$ is defined as the smallest relation satisfying the rules in Tables 2 and 3, where $e : AC$ and $p, p', q : P$.

Theorem 2 Let $p, q$ be closed BPA$^\subseteq$aa terms. Then BPA$^\subseteq$aa $\vdash p \leq q$ implies $P/\approx \models p \leq q$.

Proof Standard. First, show that $\leq$ is a precongruence w.r.t. the binary Kleene star. Second, check the validity of $BKS1 \ldots BKS3$ by constructing PBSs. Note that even strong bisimulations can be constructed for these axioms.

The following theorem states that the theory BPA$^\subseteq$aa does not introduce any inequalities (and therefore equations) between closed BPAaa$^\subseteq$ terms which were not already derivable from BPAaa$^\subseteq$. That is, the theory BPA$^\subseteq$aa is a conservative extension of BPAaa$^\subseteq$.

Theorem 3 Let $p$ and $q$ be closed BPAaa$^\subseteq$ terms. Then BPA$^\subseteq$aa $\vdash p \leq q$ iff BPAaa$^\subseteq$ $\vdash p \leq q$.

Proof (sketch) We use the theory and terminology developed in [D'Ar95]. It follows from the format of the derivation rules that the term deduction system defined by Table 2 is source dependent. Furthermore, it has no rules with negative premises, so it has a unique well supported model. For the same reason, the term deduction system defined by Tables 2 and 3 has a unique well supported model. As a result, it is an operational conservative extension (up to $\leq$) of the term deduction system of Table 2. It then follows from Theorems 1 and 2 that BPA$^\subseteq$aa is a conservative extension of BPAaa$^\subseteq$. □

A complete finite axiomatization of BPA$^*_a$ and thus BPA$^\subseteq$aa is impossible ([Sew94]; c.f. [FoZa94] for related results).
For reasoning with iterative processes, we formulate some conditional equations and inequalities. \textit{RSP*} is the iteration variant of the “Recursive Specification Principle” (c.f. e.g. [BaWe90]); the other names are derived from “Iteration Inequality.” For any \(v, w, x, y, z : \mathcal{P}\),

\[
\begin{align*}
  \frac{v \cdot x + w \leq x}{v^* w \leq x} & \quad \text{III} \\
  \frac{x \leq y \cdot x + z}{x \leq y^* z} & \quad \text{IIIR} \\
  \frac{x = y \cdot x + z}{x = y^* z} & \quad \text{RSP*}
\end{align*}
\]

Clearly, \((\text{III} \land \text{IIIR}) \Rightarrow \text{II} \Rightarrow \text{RSP*}\). It is not hard to show that \text{RSP*} and \text{BKS1} imply \text{BKS2} and \text{BKS3}.

**Theorem 4** The above derivation rules are valid in the model \(\mathcal{P}/\approx\).

**Proof (sketch)** Inequality \text{IIIR} is proved by constructing a PBS as follows. Let \(p, q\) and \(r\) be closed \text{BPA}^*aaS terms and let \(R\) be a PBS such that \(pR(q \cdot p + r)\). \(R^k\) is the k-fold composition of \(R\), where \(R^0\) equals the identity relation and \(R^1\) equals \(R\). The relation \(Q\) is defined as the smallest relation such that \(\sqrt[\infty]{Q} \sqrt[\infty]{\sqrt{\cdots}}\) and for any \(k \geq 0\), closed term \(\xi\) and open term \(E(X)\) with \(X\) as its only variable, \(\xi Q \xi\) and \(\xi R^k E(p) \Rightarrow \xi Q E(q^* r)\). Using the operational semantics, it can be shown that this relation is a PBS relating \(p\) and \(q^* r\). So \(\text{IIIR}\) is valid. The validity of \(\text{III}L\) is shown using a symmetrical argument. \(\Box\)

We have thus shown the validity of \(\text{III}L, \text{IIIR}, \text{II}\) and \(\text{RSP*}\). However, the inequalities \(\text{III}L\) and \(\text{IIIR}\) permit the derivation of, e.g., \(\overline{a}^* \delta \leq \overline{a}^* b\). This is undesirable, since the second process is terminating, whereas the first process is not. For that reason, in the remainder, we only use the weaker inequality \(\text{II}\). In \(\text{BPA}^*aaS + \text{II}\), a terminating process cannot be implemented by a nonterminating process.

**Property 4** Let \(p, q : \mathcal{P}\). If \(\text{BPA}^*aaS + \text{II} \vdash p \leq q\) and \(q \Downarrow\), then \(p \Downarrow\).

**Proof (sketch)** Structural induction on \(p\) is used. The basic case is trivial. In the induction step, we consider three cases. First, if \(p = p_1 + p_2 \leq q\), we may assume that \(\delta \neq \delta_1 \neq \delta_2 \neq \delta\). The only way that \(p \leq q\) can be deduced from \(\text{BPA}^*aaS + \text{II}\) is when \(q\) has the form \(q_1 + q_2\) and \(\text{BPA}^*aaS + \text{II} \vdash (p_1 \leq q_1 \land p_2 \leq q_2)\). If \(q \Downarrow\), then \(q_1 \Downarrow\) and \(q_2 \Downarrow\), allowing completion of the induction step in this case. Second, if \(p = p_1 \cdot p_2\), the argument is similar. Third, if \(p = p_1^* p_2\), \(p \leq q\) can be derived in two ways.

The first possibility is that \(q\) has the form \(q_1^* q_2\) and \(\text{BPA}^*aaS + \text{II} \vdash (p_1 \leq q_1 \land p_2 \leq q_2)\), which is dealt with as above. The second possibility follows from \(\text{II}\) if \(p_1 q + p_2 \leq q \leq r q + s\) and \(p_2 \leq s\), for some processes \(r\) and \(s\). Now suppose \(q \Downarrow\). We have to show that \(p \Downarrow\), i.e. that \(p_1 \Downarrow\) and \(p_2 \Downarrow\). Assume \(p_2 = \delta\). Then \(s = \delta\) and \(q \leq r \cdot q\). Soundness gives \(q \leq r \cdot q\). Since \(q \Downarrow\), it follows that \(q \xrightarrow{\alpha} \Downarrow\) for some trace \(\alpha\). So there exist nonempty traces \(\beta, \gamma\) such that \(\alpha = \beta \gamma \cdot r \xrightarrow{\beta} \Downarrow\) and \(q \xrightarrow{\gamma} \Downarrow\). So for any path of \(q\) leading to \(\Downarrow\), a shorter path can be found, which is a contradiction. So \(p_2 \neq \delta\). Hence, one deduces again that \(q\) has the form \(q_1 + q_2\) and \(\text{BPA}^*aaS + \text{II} \vdash (p_1 \cdot q \leq q_1 \land p_2 \leq q_2)\), which can again be treated as above. \(\Box\)

## 4 Concurrency and Communication

Table 4 gives the additional operators and axioms for the extension of ACP with autonomous actions. An extra parameter, the communication function \(\gamma : \mathcal{A} \cup \{\delta\} \times \mathcal{A} \cup \{\delta\} \rightarrow \mathcal{A} \cup \{\delta\}\), is added, that is commutative and associative and satisfies \(\gamma(a, \delta) = \delta\) for all atoms \(a\).
The \textit{UCM} axioms disallow synchronization of unstable processes. The \textit{SCM} axioms are \textit{CM8} and \textit{CM9} of ACP restricted to stable processes. The other merge-related axioms are standard. The auxiliary predicate \(S(.)\) stating that a process is stable is defined as follows. For \(e : \text{AC}; x, y : \text{P},\)

\[
S(\delta) \quad S(e) \quad S(x) \quad S(x + y) \quad S(x) \land S(y)
\]

Only a few of the other operators need explanation. Encapsulation of autonomous actions is impossible (D), which is the essence of being autonomous, as explained in the introduction. The renaming operator \(t\) renames the atoms from a given set \(I\) into a special atom \(t\). An operator \(\text{aut}_I\) is added, that autonomizes the atoms from \(I\). The composed operator \(t_1 \circ \text{aut}_I\) has the properties of pre-abstraction \([\text{BaBe88}]\).

\[
\text{ACPaa}^\leq(A, y) \quad \text{BPAaa}^\leq(A)
\]

TABLE 4: ACP with autonomous actions

The following theorem shows that the non-BPAaa\(^\leq\) operators can be eliminated.

**Theorem 5** For every closed ACPaa\(^\leq\) term \(p\), there is a basic BPAaa\(^\leq\) term \(s\) such that ACPaa\(^\leq\) \(\vdash p = s\).

**Proof (sketch)** Consider ACPaa\(^\leq\) as a term rewriting system. That is, consider the axioms \(A3 \ldots A7\) plus all the axioms in Table 4 as rewrite rules from left to right. We can use the method of recursive path ordering to show that this term rewriting system is strongly normalizing (see [BaVe95] for a detailed description). Here, we only give the partial ordering on the ranked operators of ACPaa\(^\leq\). For \(k > 1\) and \(a : \text{A},\)

\[
\delta < a \quad \{a, \cdot, +\} \quad \{\ldots\}_{1, k} \quad \text{aut}_I, t_1 \quad \{\ldots\}_{k, k+1} \quad \{\ldots\}
\]

Now, it can be verified by structural induction that the normal forms are basic BPAaa\(^\leq\) terms. The basic step is simple. For the induction step, note that for any normal form containing a non-BPAaa\(^\leq\) operator applied to normal forms, which by the induction hypothesis are basic BPAaa\(^\leq\) terms, a rewrite rule can
A process model is defined as before. The process space \( \mathcal{P} \) is defined as the set of closed ACPaa\( \leq \) terms. The relation \( \Rightarrow \) is the smallest relation satisfying the rules in Tables 2 and 5. Recall that in Section 2.2, \( \mathcal{S} \) is defined as the set of stable processes. Note that for all \( p, p' : \mathcal{P} \), \( p \Rightarrow S(p) \). Let \( a, b, c : A \); \( e : AC \); \( H, I : \mathcal{P}(A) \); \( p, p', q, q' : \mathcal{P} \) and \( \vec{p}, \vec{q} : \mathcal{S} \).

\[
\begin{align*}
& p \xrightarrow{e} p' & p \xrightarrow{e} \not\in H \\
& p \parallel q \xrightarrow{e} \parallel p' \parallel q, q \parallel p \xrightarrow{e} \parallel p' \parallel q & p \parallel q \xrightarrow{e} q, q \parallel p \xrightarrow{e} q, q \parallel p \xrightarrow{e} q \\
& \vec{p} \xrightarrow{e} \vec{p}', \vec{q} \xrightarrow{b} \vec{q}', \gamma(a, b) = c & \vec{p} \xrightarrow{e} \not\in H, \vec{q} \xrightarrow{b} \not\in I, \gamma(a, b) = c \\
& \vec{p} \parallel \vec{q} \xrightarrow{c} p' \parallel q', p \parallel \vec{q} \xrightarrow{c} p' \parallel q' & \vec{p} \parallel \vec{q} \xrightarrow{c} \not\in H, p \parallel \vec{q} \xrightarrow{c} \not\in H
\end{align*}
\]

Table 5: SOS rules for non-BPAaa\( \leq \) operators

The following property and theorem deal with soundness of ACPaa\( \leq \).

**Property 5** The \( \leq \) preorder is a precongruence w.r.t. the ACPaa\( \leq \) operators.

**Proof** Let \( R_1 \) and \( R_2 \) be PBSs. Let \( Q = \bigcup Q_1, Q_2 \) : \( \mathcal{P} \mid p_1 R_1 q_1 \land p_2 R_2 q_2 \cdot (p_1 \parallel p_2, q_1 \parallel q_2) \). It is easy to check that \( Q \cup R_1 \cup R_2 \) is a PBS, noticing that \( x \parallel y \) is stable iff both \( x \) and \( y \) are stable. So for any \( p_1, p_2, q_1, q_2 : \mathcal{P} \), \( (p_1 \leq q_1 \land p_2 \leq q_2) \Rightarrow p_1 \parallel p_2 \leq q_1 \parallel q_2 \). So \( \leq \) is a precongruence for the merge operator. For the left merge operator, under the same assumptions, \( Q \cup R_1 \cup R_2 \cup \bigcup Q_1, Q_2 \) : \( \mathcal{P} \mid p_1 R_1 q_1 \land p_2 R_2 q_2 \cdot (p_1 \parallel p_2, q_1 \parallel q_2) \) is a PBS; a similar PBS is constructed for the communication merge. In the latter case it is essential that communication with unstable processes is \( \delta \), so the autonomous actions of an unstable process cannot be selectively blocked.

Let \( R \) be a PBS. Define \( Q = \{ p, q : \mathcal{P} \mid p R q \cdot (\partial_H(p), \partial_H(q)) \} \). It is easy to show that \( Q \cup \{ (\not\in H, \not\in I) \} \) is a PBS, whence \( \leq \) is a precongruence for encapsulation. Like before, it is essential that the encapsulation of an autonomous action is not \( \delta \). The \( t \) and \( aut \) operators are treated likewise. □

We can now prove soundness and completeness. For Theorem 6 it suffices to check the validity of all the axioms. In each case it is straightforward to give a strong bisimulation for the added axioms. Theorem 7 is proved in the same way as Theorem 3. After that, completeness follows from Theorems 1, 5, 6.
and 7. See [D'Ar95] for more details.

**Theorem 6** Let p and q be closed ACPaa² terms. Then ACPaa² ⊢ p ≤ q implies P/≃ ⊢ p ≤ q.

**Theorem 7** Let p and q be closed BPAaa² terms. Then ACPaa² ⊢ p ≤ q iff BPAaa² ⊢ p ≤ q.

**Theorem 8** Let p and q be closed ACPaa² terms. Then ACPaa² ⊢ p ≤ q iff P/≃ ⊢ p ≤ q.

5 Example: ABP

The benchmark example for process algebra is the Alternating Bit Protocol (ABP). As the ABP has recursion as well as communication, we define the theory ACP*aa² by combining ACPaa² and BPA*aa², adding the axiom L4:

\[ L(x * y) = L(x) * L(y) \]

for \( H, I : P(A); L : \{ \delta_H, aut_I, t_I \}; x, y : P. \)

A process model can be defined as before. Soundness of ACP*aa² follows from Theorems 2 and 6 and the validity of axiom L4. ACP*aa² is a conservative extension of ACPaa² (and thus of BPAaa²). A finite complete axiomatization does not exist. Inequality II and hence equation RSP* are still valid. Furthermore, Property 4 can be generalized to closed ACP*aa² terms.

A protocol specification usually is of the form \( \sum_{d:D} ind_d : out_d \) \* \( \delta \), a never-ending loop of reading a message at one end and producing the same message at the other end. Verifications based on abstraction give an implementation with internal actions renamed to \( \tau \) and use the special properties of the silent action \( \tau \) to prove the implementation equal to the specification. Use is made of a "fairness" axiom, a simple variant of which can be stated as \( x(\tau * y) = x \cdot y \). Although such a fairness axiom is sound in many bisimulation-oriented models, one may disagree with it and use more discriminating divergence-sensitive models instead, in which case the specification equals \( \sum_{d:D} \sum_{\tau} ind_d : out_d \) \( \tau \) \* \( \delta \).

In this paper the latter approach is followed, but without silent actions and using pre-abstraction instead. Internal actions are renamed into an autonomous, but visible internal action \( \tilde{t} \). Our specification \( ABP_\text{spec} \) now reads \( \sum_{d:D} in_d : out_d \) \( \tilde{t} \) \* \( \tilde{\delta} \), i.e., before any external action, a terminating \( \tilde{t} \)-loop may take place. \( ABP_\text{spec} \) requires at least one \( \tilde{t} \)-step to be made between two external actions; a more complex specification can be given that also allows external actions to succeed one another without \( \tilde{t} \) steps in between. For the sake of simplicity we use \( ABP_\text{spec} \) as given above.

An implementation \( ABP_{\text{imp}} \) must satisfy \( ABP_{\text{imp}} \leq ABP_{\text{spec}} \). A proper implementation never loses the option of proceeding to a next external action; so it should be of the form \( CYC \ast \delta \) with \( CYC \leq \sum_{d:D} in_d : out_d \) \( \ast \delta \) and \( CYC \downarrow \). It follows from the (generalized) Property 4 that this can be established by only using the axioms from ACPaa² + II in the verification and avoiding the use of IIL and IIR.

![Figure 1: ABP implementation.](image)

In general, an implementation takes the form \( t_I \circ aut_J \circ \delta_H(\|_{k=K} P_k) \), where \( I, J \) are sets of actions internal to the implementation, \( H \) the actions that are enforced to communicate and \( P_k \) for \( k \) in \( K \) the component processes. The actions in \( J \setminus I \), while remaining internal, are still visible from outside.
If such actions are not deemed necessary, I and J can be taken equal, as in the ABP implementation below, that has been adapted from [BeKl86]. It consists of a sender S, a receiver R, a message channel K and an acknowledgement channel L. See Figure 1. Let D be the finite set of data to be transmitted. For acknowledgements, the element \( \text{ack} \notin D \) is added and for perturbed communications the element \( \perp \notin D \cup \{\text{ack}\}\). The set \( B = \{0, 1\} \) contains the alternating bit. For \( b : B \), its negation \( 1 - b \) is written \( b' \). We define the set \( M \) of messages as \( ((D \cup \{\text{ack}\}) \times B) \cup \{\perp\} \). For \( d : D \cup \{\text{ack}\}; b : B \), we denote the message \( (d, b) \) by \( db \). The set \( C = \{1, 2, 3, 4\} \) defines the communication ports. We add the rules for standard read/send communication. For \( k : C; m : M \bullet (r_k(m), s_k(m)) = c_k(m) \) (and its commutative variant). All other communications are set to \( \perp \). Let \( H = \bigcup k : C; m : M \bullet (r_k(m), s_k(m)) \) be the set of communication actions and \( I = \{k : C; m : M \bullet c_k(m)\} \) the set of internal actions. Now we can define \( ABP_{imp} \) as follows. For \( b \in B \), \( ABP_{imp} = t_1 \circ aut_1 \circ \partial_H (S \parallel K \parallel L \parallel R) \), where
\[
S = (S_0 \cdot S_1)^* \delta \\
S_b = \sum_{d \in D} in_d \cdot s_1 (db) ((\sum_{m \in M} r_d (m)) s_1 (db) )^* r_d (ack_b) \\
R = (R_0 \cdot R_1)^* \delta \\
R_b = ((\sum_{d \in D} r_d (db)) \sum_{c \in C} s_2 (ack_c) )^* ((\sum_{d \in D} r_d (db)) \sum_{c \in C} s_3 (ack_c) ) \\
K = P_{1,2} L = P_{3,4} P_{t, j} (\sum_{m \in M} r_l (m)) t \cdot s_l (m) + t \cdot s_j (m) )^* \delta
\]
By standard techniques and \( RSP^* \), one derives
\[
ABP_{imp} = (\sum_{d \in D} in_d \cdot t (t^k (\perp) \cdot \delta^k (\perp) ) \cdot ou_d \cdot \delta^k (\perp) )^* \delta.
\]
Here, \( t^k \) is the abbreviation of \( t \cdot t \cdot \ldots \cdot t \) iterated \( k \) times. The fact that \( ABP_{imp} \leq ABP_{spec} \) follows from the following derivation, showing that \( t^k (t^k (\perp)) \leq t^k (t^k (\perp)) \) for any \( k, l, m > 0 \).
By \( BKSK \), \( t^k (\perp) \leq l (t^k (\perp)) + t^k (\perp) \). So from \( I1 \) and \( I2 \),
\[
\perp \leq t^k (\perp) \leq t^k (\perp) + t^k (\perp)
\]
By induction, and the transitivity of \( \leq \), we conclude that for all \( k > 0 \),
\[
t^k (t^k (\perp)) \leq t^k (\perp), \quad \forall t^k \leq \perp
\]
Using these results, \( A3, BKSK1 \) and \( I1 \), we deduce that for any \( l, m > 0 \),
\[
t^k (t^k (\perp)) + t^k (l^k (\perp)) \leq t^k (l^k (\perp)) + t^k (l^k (\perp)) = \perp
\]
Applying \( I1 \), we obtain \( t^k (\perp) \leq \perp \). This result, the fact that for all \( k > 0 \), \( t^k (\perp) \leq \perp \) and the transitivity of \( \leq \) yield
\[
t^k (\perp) \leq \perp
\]
This concludes our verification. Since the verification took place within \( ACP^{*aa} \leq + II \), the implementation \( ABP_{imp} \) is a proper one.

### 6 Conclusions and Further Work

Actions with special properties do enrich process algebra. The present paper discusses autonomous actions, as an alternative to the "partial choice" operator of [BaBe94]. This leads to an algebra with similar expressive power and simpler axioms and models. Terms in the theory \( BPA_{aa} \), Basic Process Algebra with partial choice, can be expressed in \( BPA_{aa} \) by means of a single autonomous action \( \perp \) and following the \( p \)-graph construction of [BaBe94]. For example, the expression \( (a + c) + \perp \) is mapped to \( \perp (a + c) + \perp (b + c) \).
For reasoning with infinite processes, the \( II, III \) and \( IIR \) inequalities have been introduced. These inequalities may be carried over to other simulation-oriented preorders and can probably be widened in scope to (guarded) recursion.
One can imagine other kinds of special actions, like delayable actions \( \cdot \perp \) or irrevocable actions \( \cdot \perp \) that e.g. satisfy the equations \( \cdot \cdot x = \cdot \cdot x \) (c.f. the silent step in weak bisimulation) and \( x \cdot \cdot y + x + \cdot \cdot z = x (\cdot \cdot y + \cdot \cdot z) \) (c.f. ready trace semantics, [Gla90]). Another extension is the addition of the silent action \( \cdot \perp \). Branching bisimulation [GlWe89] can be extended to partial branching bisimulation along the lines.
of this paper, giving an algebra with two different silent actions: \( r \) and \( T \). Both satisfy the branching equations; only the latter satisfies \( I_1 \) and \( I_2 \). Although the ABP example shows that silent actions and full abstraction have ceased to be necessary for some verifications, they nevertheless simplify matters considerably.

The theory in this paper can be applied to Petri Nets like in [BaVo95] to discriminate between the consumption and production of tokens. Production of tokens can be considered as autonomous in the sense of the present paper. This leads to an inequational theory for nets with similar advantages with respect to software engineering as stated in the introduction.

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. Verhoeff
       A continuous version of the Prisoner's Dilemma, p. 17.

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

93/04  E.H.L. Aarts
       J.H.M. Korst
       P.J. Zwiering
       Deterministic and randomized local search, p. 78.

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

93/06  J.P. Velikamp
       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
       Systems Engineering: a Formal Approach

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

93/11  K.M. van Hee
       Systems Engineering: a Formal Approach

93/12  K.M. van Hee
       Systems Engineering: a Formal Approach
       Part IV: Analysis Methods, p. 63.

93/13  K.M. van Hee

93/14  J.C.M. Baeten
       J.A. Bergstra

93/15  J.C.M. Baeten
       J.A. Bergstra
       R.N. Bol

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

93/17  D. Alslein
       P. van der Stok
       Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.

93/18  C. Verhoeif
       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. Godish
       D. Dams
       G. Filé
       M. Bruynooghe
       Freeness Analysis for Logic Programs - And Correctness, p. 24.

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. Klokcs and D. Kratcsch
       Finding all minimal separators of a graph, p. 11.

93/28  F. Kamareddine and
       R. Nederpelts
       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. Klokcs
       D. Kratcsch
       H. Müller
       On Vertex Ranking for Permutation and Other Graphs, p. 11.
Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.


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

Real Time Process Algebra with Infinitesimals, p. 39.

Abstract Reduction and Topology, p. 28.

Non Interleaving Process Algebra, p. 17.

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

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

Job Shop Scheduling by Constraint Satisfaction, p. 22.


Temporal operators viewed as predicate transformers, p. 11.

Automatic Verification of Regular Protocols in P/T Nets, p. 23.

A taxonomy of finite automata construction algorithms, p. 87.

A taxonomy of finite automata minimization algorithms, p. 23.

A precise clock synchronization protocol, p. 16.


Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

The object-oriented paradigm, p. 28.

Canonical typing and II-conversion, p. 51.


Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.


Time and the Order of Abstract Events in Distributed Computations, p. 29.


A Hierarchical Diagrammatic Representation of Class Structure, p. 22.

Process Algebra with Partial Choice, p. 16.
<table>
<thead>
<tr>
<th>Paper ID</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>94/13</td>
<td>R. Seljé</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 β-Reduction in Church's λ→, 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>94/25</td>
<td>T. Kloks</td>
<td>K_{1,3}-free and W_4-free graphs, p. 10.</td>
</tr>
<tr>
<td>94/29</td>
<td>J. Hooiman</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>
</tbody>
</table>
On the equivalence covering number of splitgraphs, p. 4.

Distributed Consensus and Hard Real-Time Systems, p. 34.

Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.

Concatenation of Graphs, p. 7.

Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.

Verifying Sequentially Consistent Memory, p. 160

Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.

The A-cube with classes of terms modulo conversion, p. 16.

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

Fixed-Point Calculus, p. 11.

Process Algebra with Propositional Signals, p. 25.

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

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

Traces and Logic, p. 81

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

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

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

Concrete process algebra, p. 134.

An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.

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

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

Drawing Execution Graphs by Parsing, p. 10.

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

Discrete Time Process Algebra, p. 20.

Deductive Database Systems and integrity constraint checking, p. 36.

Empty Interworkings and Refinement
Semantics of Interworkings Revised, p. 19.


De proceedings: ACP'95, p.

On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.

Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.

Specification of tools for Message Sequence Charts, p. 36.


Discrete Time Process Algebra with Abstraction, p. 15.

On Normalisation, p. 33.

Axiomatising Early and Late Input by Variable Elimination, p. 44.


Petri net based scheduling, p. 20.


Synchronous Sequence Charts In Action, p. 36.

A Class of Petri nets for modeling and analyzing business processes, p. 24.


A Conservative Look at term Deduction Systems with Variable Binding, p. 29.

On Nesting of a Nonmonotonic Conditional, p. 14


Practical Symbolic Model Checking of the full \( \mu \)-calculus using Compositional Abstractions, p. 17.


Context-Free Graph Grammars and Concatenation of Graphs, p. 35.

Record concatenation with intersection types, p. 46.

An algebraic semantics for hierarchical P/T Nets, p. 32.