This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

All rights reserved
editors:  prof.dr.M.Rem
          prof.dr.K.M.van Hee.
Process Algebra with Partial Choice

J.C.M. Baeten
Department of Computer 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 objective of this paper is to bridge the gap between ACP and TCSP. To this end, ACP is extended with two non-deterministic choice operators in a setting of bisimulation semantics. With these operators, we can express safety properties of systems without the use of silent steps, and we can verify safety properties in a setting in which no assumption on fairness (or unfairness) has been made.

1980 Mathematics Subject Classification (1985 revision): 68Q55, 68Q10, 68Q45.
Key words & Phrases: non-determinism, choice, partial choice, safety property.

Note: This research was supported in part by ESPRIT basic research action 7166, CONCUR2. The second author is also partially supported by ESPRIT basic research action 6454, CONFER.

1. INTRODUCTION.

The objective of this paper is to bridge the gap between ACP [BEK84, BAW90] and TCSP [BRHR84, HOA85]. Earlier, in [BEKO88] and [BABK87], ready, failure and ready trace semantic models have been provided for ACP. However, these models do not take into account TCSP's rather sophisticated views on non-determinism versus choice. We agree with the observation underlying TCSP that externally influencible choice and non-determinism are to be distinguished and that more than one operator is needed.

After many experiments, we have concluded that it may be useful to extend ACP with two more operators:

- \( \mp \) static or partial alternative composition
- \( \sqcup \) collecting alternative composition.

These operators exist together with the usual alternative composition operator

\( + \) called dynamic alternative composition here.

A choice \( p + q \) is made exactly when the first action of \( p \) or \( q \) is executed, so the choice is made at the root of the system. On the other hand, a choice \( p \sqcup q \) is made a long time ago, before any action is executed in the system whatsoever. Finally, a choice \( p \mp q \) is somewhere in between the previous actions, but we don't know where; \( p \mp q \) behaves somewhere between \( p \sqcup q \) and \( p + q \).
The choice \( p \uplus q \) is called static because when control arrives at its root state probably just one option is left; the choice \( p \sqcup q \) is called collecting because it represents \( \{p, q\} \) (with the notational convention in mind that \( p = \{p\} \)). The choice \( p \uplus q \) is called partial because only partial information about the timing of the choice is available.

In [BAB94] we have provided a connection between the communication mechanisms of CCS (see [Mil80]) and ACP. This paper complements that one. It deviates from [BKO88] by being based on bisimulation entirely. In this way we hope to have it both ways: combine TCSP's sophistication concerning choice and non-determinism with the mathematical clarity of bisimulations, the cornerstone of CCS.

Further we find that the introduction of partial choice and collecting choice allows us in many cases to express safety properties of systems without the use of silent steps and to verify safety properties in a setting in which no assumption about fairness (or unfairness) has yet been made. Avoiding silent steps as well as fairness considerations are virtues of TCSP that we consider advantages indeed. We hope to have extended ACP in such a way that these virtues have become available in the ACP setting as well.

Moreover, our introduction of partial choice may shed new light on probabilistic choice (see [BABS92]) and the combination of probabilistic choice with other choice operators (cf. [JON93]).

Finally, our development is such that in the absence of silent steps, a model can also be found along the lines of the projective limit model of [BEK84], which is in fact an alternative presentation of the metric space model of [BAZ82].

2. PROCESS ALGEBRA WITH PARTIAL CHOICE.

We start by extending process algebra with the partial choice operator \( \uplus \).

2.1 BASIC PROCESS ALGEBRA.

Assume that we have a (finite) set of constants \( A = \{a, b, c, \ldots\} \) and a special constant \( \delta \in A \). Further, we have the binary operators + (dynamic choice), \( \uplus \) (partial choice), \( \cdot \) (sequential composition). The axioms for +, \( \cdot \) are the standard axioms for BPA\(_A\), except that axiom A3 \((X + X = X)\) is restricted to atomic actions. See table 1 \((a \in A)\). A3 is restricted, because it does not hold anymore for processes involving the new choice operator. Note that the axioms in table 1 still provide a complete axiomatisation of the standard model of BPA\(_A\). This is a consequence of the following lemma.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X + Y = Y + X )</td>
<td>A1</td>
</tr>
<tr>
<td>((X + Y) + Z = X + (Y + Z))</td>
<td>A2</td>
</tr>
<tr>
<td>( a + a = a )</td>
<td>AA3</td>
</tr>
<tr>
<td>((X + Y) \cdot Z = X \cdot Z + Y \cdot Z)</td>
<td>A4</td>
</tr>
<tr>
<td>((X \cdot Y) \cdot Z = X \cdot (Y \cdot Z))</td>
<td>A5</td>
</tr>
<tr>
<td>( X + \delta = X )</td>
<td>A6</td>
</tr>
<tr>
<td>( \delta \cdot X = \delta )</td>
<td>A7</td>
</tr>
</tbody>
</table>

**Table 1.** BPA\(_A\) with restricted A3.
2.2 LEMMA. Let $t$ be a closed term over BPA. Then $A1,2,4,6, A3 \vdash t + t = t$.
PROOF: We use induction on the structure of basic terms (see e.g. [BAW90]).

i. Constants: for atoms, this is $A3$, for $\delta$ it follows from $A6$.

ii. Prefix multiplication: $a \cdot x + a \cdot x = (a + a) \cdot x = a \cdot x$.

iii. Dynamic sum: $(x + y) + (x + y) = x + (y + (x + y)) = x + ((x + y) + y) = x + (x + (y + y)) = (x + x) + (y + y) = x + y$.

2.3 PARTIAL SUM.
The partial alternative composition operator has the same laws as the dynamic alternative composition operator, except that we have the full analogue of axiom $A3$ of BPA $(X + X = X)$, but no $A6$. As far as mutual distributivity of alternative composition operators is concerned, partial sum comes before dynamic sum. The partial sum operator obeys exactly the laws of the sum operator in [BABS92]. Thus it is the partial sum operator that can be extended to a probabilistic sum.

We call the basic theory with the two alternative composition operators BPA$\oplus$.

| $X \uplus Y = Y \uplus X$                     | PAC1          |
| $$(X \uplus Y) \uplus Z = X \uplus (Y \uplus Z)$$ | PAC2          |
| $X \uplus X = X$                             | PAC3          |
| $$(X \uplus Y) \cdot Z = X \cdot Z \uplus Y \cdot Z$$ | PAC4          |
| $$(X \uplus Y) + Z = (X + Z) \uplus (Y + Z)$$   | PAC5          |

**TABLE 2.** Partial alternative composition.

2.4 REALISATION.
Now we want to express a notion of realisation or implementation. We use a partial order $\leq$ in order to express this. Thus, if $X \leq Y$, then $X$ realises $Y$, i.e. $X$ has less static non-determinism than $Y$. The central law is PAC6 in table 3.

| $X \leq X \uplus Y$                     | PAC6          |

**TABLE 3.** Realisation.

Thus, our axiom system contains equations and inequalities. We assume the standard properties of equality in reasoning about systems of equations. Similarly, we have the following properties for reasoning about inequalities:

- $X \leq X$                       reflexivity
- $X \leq Y \land Y \leq X \Leftrightarrow X = Y$ antisymmetry
- $X \leq Y \land Y \leq Z \Rightarrow X \leq Z$ transitivity
- $X \leq Y \Rightarrow C[X] \leq C[Y]$ closed under contexts
- $X \leq Y \Rightarrow \rho(X) \leq \rho(Y)$ substitutivity (\(\rho\) a mapping from variables to terms), i.e. we have a partial order algebra in the terminology of [HEN88].
2.5 LEMMA. For all closed terms \( t \) over \( \text{BPA}_\bot \), we have \( t \leq t + t \).

PROOF: For terms over \( \text{BPA}_\emptyset \), this is a consequence of lemma 2.2. It remains to consider the partial sum operator. By induction, assume the lemma holds for \( x, y \), then:

\[
x \vdash y \leq x \vdash y \vdash (x + y) \leq (x + x) \vdash (y + x) \vdash (x + y) \vdash (y + y) =
\]

\[
= ((x \vdash y) + x) \vdash ((x \vdash y) + y) = (x + (x \vdash y)) \vdash (y + (x \vdash y)) = (x \vdash y) + (x \vdash y).
\]

We can give an intuition for this inequality as follows: \( t + t \) has more non-determinism than \( t \), because different non-deterministic choices can have been made in the different copies of \( t \).

2.6 SEMANTICS.

We define a process graph model, similar in spirit to the stratified model of [HAJ90, HAN91]. We distinguish between a partial sum branching and a dynamic sum branching.

We define a \( p \)-graph to be a quintuple \( \langle S, D, \rightarrow, \rightarrow', r \rangle \) such that:

- \( S \) is the set of static or partial states
- \( D \) is the set of dynamic states; \( S \) and \( D \) are disjoint
- \( \rightarrow \subseteq D \times A \times S \) is the labeled transition relation, denoting dynamic choice
- \( \rightarrow' \subseteq S \times D \) is the unlabeled transition relation, denoting static choice
- \( r \in S \) is the root node.

Define \( \mathcal{PG} \) to be the set of all p-graphs that satisfy the following restrictions:

1. \( S \) and \( D \) are countable
2. the root node has at least one outgoing transition.

A static state without outgoing transitions is called an end node. A dynamic state without outgoing transitions is called a deadlock node. We write \( s \rightarrow d \) iff \( \langle s, d \rangle \in \rightarrow \), and \( d \rightarrow' s \) iff \( \langle d, a, s \rangle \in \rightarrow' \). The names of the states do not matter, so actually we are considering equivalence classes of p-graphs under graph isomorphism. This assumption enables us to assume that state sets are disjoint whenever this is convenient.

The interpretation function will assign a p-graph from \( \mathcal{PG} \) to each process. We proceed to define the interpretation. First, constants.

\[
[a] = \{(0, 2), (1), (1, a, 2), (0, 1)\}, 0
\]

\[
[I] \emptyset = \{(0), (1), \emptyset, (0, 1)\}, 0.
\]

Fig. 1 illustrates these definitions. We use the following pictorial conventions: closed circles denote static states, open circles dynamic states, and a small incoming arrow denotes the root node.

![FIGURE 1. Constants.](image-url)

Partial sum. We define \( g \vdash h \) for two p-graphs \( g, h \). Let \( g,h \) be given with disjoint state sets. Then \( g \vdash h \) is obtained by creating a new root node \( r \), and adding, for each unlabeled transition \( s \rightarrow d \) where \( s \) is the root of \( g \) or \( h \), a new unlabeled transition \( r \rightarrow d \).
Dynamic sum. Let two p-graphs \( g,h \) be given with disjoint state sets. We obtain \( g+h \) by creating a new root node \( r \), and creating a new dynamic node \( (d,e) \) for each pair consisting of a dynamic node \( d \) in \( g \) reachable from the root by an unlabeled arrow, \( e \) a similar node in \( h \). We have unlabeled transitions \( r \rightarrow (d,e) \) for each such new dynamic node, and a labeled transition \( (d,e) \xrightarrow{a} s \) whenever there is a transition \( d \xrightarrow{a} s \) or \( e \xrightarrow{a} s \). As an example, we show \( (a \rightarrow b) + c \) in fig. 2 (omitting unreachable nodes).

Sequential composition. Let two p-graphs \( g,h \) be given with disjoint state sets. We obtain \( g \cdot h \) by appending a copy of \( h \) to each endnode of \( g \).

2.6 Structured operational semantics.

![Figure 2. Dynamic sum.](image)

Table 4. Operational semantics of closed terms over BPA\( _{\sqcup} \).

<table>
<thead>
<tr>
<th>Term</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \rightarrow a )</td>
<td>( a \rightarrow \top )</td>
</tr>
<tr>
<td>( x \rightarrow x' )</td>
<td>( x \rightarrow x' )</td>
</tr>
<tr>
<td>( x \cdot y \rightarrow x \cdot y' )</td>
<td>( x \cdot y \rightarrow x \cdot y' )</td>
</tr>
<tr>
<td>( x \rightarrow x', y \rightarrow y' )</td>
<td>( x \rightarrow x', y \rightarrow y' )</td>
</tr>
<tr>
<td>( x+y \rightarrow x', y+x \rightarrow y' )</td>
<td>( x+y \rightarrow x', y+x \rightarrow y' )</td>
</tr>
<tr>
<td>( x \rightarrow x' )</td>
<td>( x \rightarrow x' )</td>
</tr>
<tr>
<td>( x \rightarrow x' )</td>
<td>( x \rightarrow x' )</td>
</tr>
</tbody>
</table>

TABLE 4. Operational semantics of closed terms over BPA\( _{\sqcup} \).
We can also obtain a p-graph for each closed term by means of a structured operational semantics. We show the rules in table 4. Static states are given by terms (over BPA-H), dynamic states by boxed terms (over BPAl). Modulo the definition of bisimulation that follows, the structured operational semantics yields the same graphs as the definitions above. We will omit the proof of this fact.

2.7 PARTIAL BISIMULATION.
Let two P-graphs g, h be given, and let R be a relation between nodes of g and nodes of h. R is a partial bisimulation if:

i. the roots are related

ii. if R(s, t) and s \rightarrow d then there is node e in h such that t \rightarrow e and R(d, e)

iii. if R(d, e) and d \rightarrow a s then there is node t in h such that e \rightarrow a t and R(s, t)

iv. if R(d, e) and e \rightarrow a t then there is node s in g such that d \rightarrow a s and R(s, t)

v. if R(s, t) then s is an end node iff t is an end node.

We say that g is partially bisimilar to h, denoted g \leftrightarrow h, if there is a partial bisimulation from g to h.

We say that two P-graphs g, h are bisimulation equivalent if g \leftrightarrow h and h \leftrightarrow g. It is easy to see that this is indeed an equivalence relation. Denote by PG/\leftrightarrow the set of equivalence classes. Put PG/\leftrightarrow = \{s \leq t \mid [s] \leftrightarrow [t]\} and PG/\leftrightarrow = s = t if \[s\] and \[t\] are bisimulation equivalent.

2.8 THEOREM (SOUNDNESS). Let S, t be closed terms. Then BPA \vdash S \leq t \iff [S] \leftrightarrow [t].

HINT OF PROOF: First we need to prove that PG/\leftrightarrow is a partial order algebra. Then, it is sufficient to check each axiom separately.

We proceed to investigate the converse of theorem 2.8, i.e. the completeness of our axiomatisation. In order to do this, we first need the notion of a basic term.

2.9 DEFINITION. We define the set of basic terms B inductively, with the help of an intermediary set \(B_+\). In B, the top operator is a partial sum and in \(B_+\) a dynamic sum.

i. \(A \cup \{\delta\} \subseteq B_+ \subseteq B\)

ii. \(a \in A, t \in B \Rightarrow a \cdot t \in B_+\)

iii. \(t, s \in B_+ \Rightarrow t + s \in B_+\)

iv. \(t, s \in B \Rightarrow t + s \in B\).

An example of a basic term: \((a \cdot (b \leftrightarrow (c + d)) + b) \leftrightarrow \delta\).

Let p be a process in BPA. We say p has a dynamic head normal form (p \in HNF+) if there are \(n, m \in \mathbb{N}\) with \(n + m > 0\), \(a_1, ..., a_n \in A, b_1, ..., b_m \in A \cup \{\delta\}\) and processes \(p_1, ..., p_n\) such that

\[BPA \vdash p = a_1 \cdot p_1 + ... + a_n \cdot p_n + b_1 + ... + b_m.\]

We say a process q has a partial head normal form (q \in HNF+) if there is \(k > 0\) and processes \(q_1, ..., q_k\) in dynamic head normal form such that

\[BPA \vdash q = q_1 \leftrightarrow ... \leftrightarrow q_k.\]

Note that PG/\leftrightarrow = p = p + p holds for all \(p \in HNF+.\) It is easy to see that this law does not hold for all processes, as the counterexample \(p = a \leftrightarrow b\) shows.

2.10 THEOREM: Let S be a closed term. Then there is a basic term t such that BPA \vdash S = t.
HINT OF PROOF: By term rewrite analysis. It is straightforward to show that the term rewrite system shown in table 5 is confluent and normalising. Thus, each term has a unique normal form. Each normal form is a basic term. The reduction to normal form constitutes a proof in \( \text{BPA}_\rightarrow \).

\[
\begin{align*}
(X + Y) \cdot Z &\rightarrow X \cdot Z + Y \cdot Z \\
(X \cdot Y) \cdot Z &\rightarrow X \cdot (Y \cdot Z) \\
\delta \cdot X &\rightarrow \delta \\
(X + Y) \cdot Z &\rightarrow X \cdot (Z + Y) \\
(X + Y) + Z &\rightarrow (X + Z) + (Y + Z) \\
X + (Y + Z) &\rightarrow (X + Y) + (X + Z)
\end{align*}
\]

<table>
<thead>
<tr>
<th>Table 5. Rewrite system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA4</td>
</tr>
<tr>
<td>RA5</td>
</tr>
<tr>
<td>RA7</td>
</tr>
<tr>
<td>RPAC4</td>
</tr>
<tr>
<td>RPAC5</td>
</tr>
<tr>
<td>RPAC5'</td>
</tr>
</tbody>
</table>

2.11 THEOREM (COMPLETENESS). Let \( s, t \) be closed terms. Then \( \lbrack s \rbrack \leftrightarrow \lbrack t \rbrack \Rightarrow \text{BPA}_\rightarrow \vdash s \leq t \).

HINT OF PROOF: By 2.10, it is enough to consider basic terms. Using the structured operational semantics of 2.6, we show that for each basic term \( t \) we have:

- \( t \rightarrow \vec{a} \rightarrow t' \Leftrightarrow \exists t'' \in B \land \exists s, s' \in B_+ \) ((PAC1,2 \vdash t = t'' \Leftrightarrow s \lor PAC1,2 \vdash t = s) \land (A1,2 \vdash s = s' + a \lor A1,2 \vdash s = a \cdot t'))

- \( t \rightarrow \vec{a} \rightarrow \vec{1} \Leftrightarrow \exists t'' \in B \exists s, s' \in B_+ \) ((PAC1,2 \vdash t = t'' \Leftrightarrow s \lor PAC1,2 \vdash t = s) \land (A1,2 \vdash s = s' + a \lor A1,2 \vdash s = a))

On the basis of these equivalences, the theorem follows easily.

2.12 ADDITIONAL OPERATORS.

We sketch the extension of \( \text{BPA}_\rightarrow \) with additional operators. First, free merge. In table 6, \( a \in A \cup \{ \delta \} \).

\[
\begin{align*}
X \parallel Y &= X \parallel Y + Y \parallel X \\
a \parallel X &= a \cdot X \\
a \cdot X \parallel Y &= a \cdot (X \parallel Y) \\
(X \parallel Y) \parallel Z &= X \parallel Z \parallel Y \parallel Z \quad \text{for } \oplus \in \{ +, \parallel \}
\end{align*}
\]

Table 6. Additional axioms for \( \text{PA}_\rightarrow \).

Next, merge with communication, and encapsulation. In table 7, \( a \in A \cup \{ \delta \} \).
Finally, we will consider systems of recursive equations. Treatment of these is standard, see e.g. [BAW90]. We will only consider completely guarded equations in the sequel. As an open question, we leave the formulation of axioms or proof rules in order to deal with such systems. Thus, we will only use semantic reasoning, where infinite processes are concerned.

2.14 GRAPH MODEL.

Definition of parallel composition on the graph model is not so straightforward. We give the definition of merge with communication only for a restricted subset, for graphs in which each dynamic state has exactly one incoming transition. We can do this because each p-graph is bisimulation equivalent with such a graph (proof omitted). Then the definition goes as follows:

let two p-graphs g = (S₁, D₁, →₁, ⊢₁, r₁), h = (S₂, D₂, →₂, ⊢₂, r₂) be given with disjoint state sets and with exactly one incoming transition for each dynamic node. The set of static nodes of g || h is S₁ X S₂, the set of dynamic nodes D₁ X D₂, the root is (r₁, r₂), we have ⟨s,t⟩ → ⟨d,e⟩ iff s →₁ d and t →₂ e and ⟨d,e⟩ →₂ ⟨s,t⟩ if

- d →₁ s and t →₂ e, or
- e →₂ t and s →₁ d, or
- d →₁ s, e →₂ t and b ⊢ c = a ≠ δ.

Next, the definition of encapsulation is straightforward. Let p-graph g be given. ∂₇(g) is obtained from g by erasing all transitions d → s with a ∈ H.

We give a structured operational semantics for the additional operators in table 8 (a,b,c ∈ A). Since after a static step in a parallel composition, the component not chosen has to "take back" the static step, we have to remember the static nodes. This requires an operator with four arguments: _ _ _ _ , the
inner arguments denoting the dynamic states, the outer ones the static states they came from. The last line gives an operational semantics for recursion: here \( X = \tau X \) is an equation in a recursive specification \( E \).

\[
\begin{align*}
\frac{x \rightarrow [x], y \rightarrow [y]}{x \parallel y \rightarrow x[x] \parallel y[y]} & \quad \frac{x \rightarrow x^*}{x \parallel y \rightarrow x[x] \parallel y[y]} \\
\frac{x \rightarrow x^*}{x \parallel y \rightarrow x[x] \parallel y[y]} & \quad \frac{y \rightarrow y^*}{x \parallel y \rightarrow x[x] \parallel y[y]} \\
\frac{x \rightarrow \tau, y \rightarrow \tau, b \mid c = a}{x \parallel y \rightarrow x[x] \parallel y[y]} & \quad \frac{x \rightarrow \tau, y \rightarrow \tau, b \mid c = a}{x \parallel y \rightarrow x[x] \parallel y[y]} \\
\frac{x \rightarrow x', y \rightarrow y'}{x \parallel y \rightarrow x[x] \parallel y[y]} & \quad \frac{x \rightarrow x', y \rightarrow y'}{x \parallel y \rightarrow x[x] \parallel y[y]} \\
\frac{x \rightarrow x'}{\partial H(x) \rightarrow \partial H(x')} & \quad \frac{x \rightarrow x'}{\partial H(x) \rightarrow \partial H(x')} \\
\frac{\tau x \rightarrow y}{x \rightarrow y} & \quad \frac{\tau x \rightarrow y}{x \rightarrow y} \\
\frac{x \rightarrow y}{x \rightarrow y} & \quad \frac{x \rightarrow y}{x \rightarrow y}
\end{align*}
\]

**TABLE 8. Operational semantics of additional operators.**

2.14 EXAMPLE. We illustrate the definition of parallel composition. We assume there is no communication.
3. IMPLEMENTATION.
If a process \( P \) can be written without the \( \top \) operator, we call \( P \) total. If \( P \leq Q \), we say that \( P \) is a realisation of \( Q \). If moreover \( P \) is total and \( P \) is computable, then we say that \( P \) is an implementation of \( Q \).

3.1 STANDARD REALISATION.
We define the standard realisation function \( \text{SR} \) inductively (\( a \in A \cup \{ \delta \} \)):

i. \( \text{SR}(a) = a \)

ii. \( t \in \text{HNF} \Rightarrow \text{SR}(a \cdot t) = a \cdot \text{SR}(t) \)

iii. \( \text{SR}(t + s) = \text{SR}(t) + \text{SR}(s) \)

iv. \( \text{SR}(t \top s) = \text{SR}(t) \top \text{SR}(s) \).

Note that since \( a \cdot (t \top s) = (a + a) \cdot (t \top s) = a \cdot (t \top s) + a \cdot (t \top s) \geq a \cdot t + a \cdot s \), we always have \( \text{SR}(p) \geq p \). Moreover, if \( t \) is in dynamic head normal form, then \( \text{SR}(t) \) is total.

EXAMPLE: an unreliable channel, that may corrupt messages is given by the recursive equation

\[
K = \sum_{d \in D} r_1(d) \cdot (s_2(d) \top s_2(\bot)) \cdot K
\]

The standard realisation satisfies \( \text{SR}(K) = \sum_{d \in D} r_1(d) \cdot (i \cdot s_2(d) + i \cdot s_2(\bot)) \cdot \text{SR}(K) \).

3.2 DEFINITION. We call a process \( P \) deadlockfree iff \( P \) has a \( \tau \)-graph without a deadlock node.

3.3 PROPOSITION. Let \( Y \leq X \). If \( X \) is deadlockfree, then \( Y \) is deadlockfree.

HINT OF PROOF: Suppose not, then \( Y \) has a path to a deadlock node \( s \). Since \( \ll Y \ll \iff \ll X \ll \), there is also a dynamic state in \( X \) bisimulating with \( s \). By definition of partial bisimulation, this state can have no outgoing transitions. Thus, \( X \) has a deadlock node, contradiction.

3.4 NOTE. A method for verification may now proceed as follows. First, a system is described in the form \( P = \partial_H(P_1 \parallel ... \parallel P_n) \) for certain processes \( P_i \) that might contain partial choice operators. Then, a proof is given that \( \partial_H(\text{SR}(P_1) \parallel ... \parallel \text{SR}(P_n)) \) is a solution for \( X \), in a specification that is usually in terms of \( \tau_i(X) \), say \( \tau_i(X) = E(\tau_i(X)) \) for a certain recursive equation \( E \). Using standard realisation, each of the components is given an implementation, so that we get an automaton model. Often, the first step is skipped in this procedure. This introduces philosophical complications about the states of the specification.

If \( \partial_H(\text{SR}(P_1) \parallel ... \parallel \text{SR}(P_n)) \) fails the specification, we know that a very plausible implementation fails, so we are warned about the validity of \( \partial_H(P_1 \parallel ... \parallel P_n) \). Often, that \( \partial_H(\text{SR}(P_1) \parallel ... \parallel \text{SR}(P_n)) \) satisfies the specification is proven under a general overall fairness assumption (which is quite reasonable in case of finite state systems). Then, if implementations \( Q_i \leq P_i \) are considered which are sometimes not fair, we must be careful: process \( \partial_H(Q_1 \parallel ... \parallel Q_n) \) may contain an I-livelock.
that $\partial_H(SR(P_1) \parallel \ldots \parallel SR(P_n))$ does not show. An example of this is given by the unreliable channel in 3.1, used in an alternating bit protocol (as in 3.5), taking the implementation

$$Q = \sum_{d \in D} r_1(d) \cdot i \cdot s_2(\perp) \cdot Q$$

of the channel. Even in other cases, the proof of $\tau_1(X) = E(\tau_1(X))$ may use a kind of combined fairness which the $Q_i$ cannot be supposed to have.

3.5 ALTERNATING BIT PROTOCOL.

Let $D$ be a finite set of data. We use the standard read/send communication function given by $r_k(x) \parallel s_k(x) = c_k(x)$ for communication port $k$ and message $x$ (and $\delta$ otherwise). We specify sender $S$, unreliable channels $K,L$ and receiver $R$ as in [BAW90].

$$S = S0 \cdot S1 \cdot S$$

$$S_b = \sum_{d \in D} r_1(d) \cdot S_d$$

for $b = 0,1$.

$$S_d = s_2(db) \cdot (r_6(1-b) + r_6(\perp)) \cdot S_d + r_6(b)$$

for $b = 0,1, d \in D$.

$$K = \sum_{d \in D, b \in B} r_2(db) \cdot i \cdot (s_3(db) + s_3(\perp)) \cdot K$$

$$L = \sum_{b \in B} r_5(b) \cdot i \cdot (s_6(b) + i \cdot s_6(\perp)) \cdot L$$

$$R = R1 \cdot R0 \cdot R$$

$$R_b = (\sum_{d \in D} r_3(db) + r_3(\perp)) \cdot s_5(b) \cdot R_b + \sum_{d \in D} r_3(d(1-b)) \cdot s_4(d) \cdot s_5(1-b)$$

for $b = 0,1$.

$ABP = t_0 \circ \partial_H(S \parallel K \parallel L \parallel R)$,

where $H = \{r_k(x), s_k(x) : k \in \{2,3,5,6\}, x \in D \cup (D \times B) \cup B \cup \{\perp\}\}$ is the set of communication actions, $I = \{c_k(x) : k \in \{2,3,5,6\}, x \in D \cup (D \times B) \cup B \cup \{\perp\}\}$ is the set of internal actions, and $t_1$ is the pre-abstraction operator (from [BAB88]), that renames all internal actions into $t$.

$ABP$ is a partial process specifying the alternating bit protocol. Constructing the transition system for this process, we can derive the following recursive specification for $ABP$:

$$X = \sum_{d \in D} r_1(d) \cdot Y_d$$

$$Y_d = t \cdot t \cdot t \cdot t \cdot t \cdot Y_d \parallel t \cdot s_4(d) \cdot Z$$

for $d \in D$.

$$Z = t \cdot t \cdot t \cdot t \cdot Z \parallel t \cdot X$$.

Adding more non-determinism, we can simplify this specification considerably. Consider the following recursive specification, and let process $BUF1$ be a solution for this specification:

$$X' = t \cdot X' \parallel \sum_{d \in D} r_1(d) \cdot Y'_d$$

$$Y'_d = t \cdot Y'_d \parallel s_4(d) \cdot X'$$

for $d \in D$.

By considering the p-graphs of these processes, we can derive $ABP \leq BUF1$ holds in our model. It is an open question, how to derive this inequality on the basis of algebraic calculations.
3.6 FIFO QUEUES.

BUF1 is a general specification of a one-place buffer. It has many different realisations. We think that every one-place buffer described will be a realisation of this process. We can generalise even further, and present (without further comment) a specification (with input port 1 and output port 2) such that every unbounded or bounded buffer or queue will be a realisation of it. The variables are parametrised by sequences of data elements from $D$.

$$Q_e = Q_e^t \uplus Q_e^r \uplus Q_e^t$$

$(\epsilon$ empty sequence)$

$$Q_{ad} = Q_{ad}^t \uplus Q_{ad}^r \uplus Q_{ad}^t \uplus Q_{ad}^s \uplus Q_{ad}^{sr} \uplus Q_{ad}^{st} \uplus Q_{ad}^{srt}$$

$(\sigma \in D^*, d \in D)$

$$Q_{\sigma}^t = t \cdot Q_{\sigma}$$

$(\sigma \in D^*)$

$$Q_{\sigma}^r = \sum_{d \in D} r_1(d) \cdot Q_{d\sigma}$$

$(\sigma \in D^*)$

$$Q_{\sigma}^t = t \cdot Q_{\sigma} + \sum_{d \in D} r_1(d) \cdot Q_{d\sigma}$$

$(\sigma \in D^*)$

$$Q_{ad}^s = s_2(d) \cdot Q_{\sigma}$$

$(\sigma \in D^*, d \in D)$

$$Q_{ad}^{sr} = s_2(d) \cdot Q_{\sigma} + \sum_{e \in D} r_1(e) \cdot Q_{e\sigma d}$$

$(\sigma \in D^*, d \in D)$

$$Q_{ad}^{st} = s_2(d) \cdot Q_{\sigma} + t \cdot Q_{\sigma d}$$

$(\sigma \in D^*, d \in D)$

$$Q_{ad}^{srt} = s_2(d) \cdot Q_{\sigma} + t \cdot Q_{\sigma d} + \sum_{e \in D} r_1(e) \cdot Q_{e\sigma d}$$

$(\sigma \in D^*, d \in D)$.

4. PROCESS ALGEBRA WITH COLLECTING CHOICE.

In a setting where we also have collecting sum $\sqcup$, some things change. The main difference is, that all axioms that involve a duplication of a variable turn into inequalities.

4.1 BASIC PROCESS ALGEBRA.

We start with BPA, so we have a (finite) set of constants $A = \{a, b, c, \ldots\}$, a special constant $\delta \in A$, and binary operators $+, \sqcup, \uplus, \cdot$. Axioms A4, PAC3,4,5 are not valid anymore. The axioms in table 9 replace the ones of BPA in tables 1 and 2. Notice that the inequality version of PAC3, $X \leq X \uplus X$, is a direct consequence of PAC6.
Next, we show the axioms for the collecting alternative composition in table 10. This operator distributes both ways over sequential composition. This is an important difference with dynamic and static choice. Replacing all + operators by \(\sqcup\) operators gives us the traces of a process. Thus, we do not view trace theory as a homomorphic image of a bisimulation based process algebra, but rather obtain a set of traces by using static choice only. Note that dynamic and partial choice distribute over \(\sqcup\) but not the other way around, thus the static choice has precedence (comes before) other forms of choice. In TCSP there is distribution both ways. We think that this two-way distribution is incompatible with the bisimulation model, and forces the use of failure semantics. Therefore, we do not include the other distributive law here.

As for partial choice, \(X \sqcup \delta = X\) does not hold. This is because the choice has already been made, and so deadlock cannot be avoided if the wrong choice has been made. CAC8 expresses the idea that realisation reduces collecting non-determinism.

The axioms in tables 9 and 10 together give the theory BPA_{\sqcup}.
If a process $P$ can be written without the $\sqcup$ operator, we call $P$ connected. Thus, in section 2 we treated the theory of the connected processes. Consistent with the terminology in section 3, if $P$ can be written without $\sqcup$ and $\sqcap$, we call $P$ total.

4.2 SEMANTICS.
In a setting with collecting sum, our semantic domain now consists of nonempty sets of p-graphs. Let $\mathbb{PPG}$ be the set of nonempty sets of p-graphs from $\mathbb{PG}$. The interpretation of the constants now becomes the singleton set consisting of the earlier interpretation.

For collecting sum, we simply put $[P \sqcup Q] = [P] \cup [Q]$. For all other binary operators $\sqcap$ (so $\sqcap \in \{+, \cdot, \sqcup\}$) we put $[P \sqcap Q] = \{g \sqcap h : g \in [P], h \in [Q]\}$.

We define partial bisimulation on $\mathbb{PPG}$ as follows: $G \leftrightarrow H$ iff $\forall g \in G \exists h \in H \ g \leftrightarrow h$.

We can still obtain soundness as before. For basic terms, we need an extra layer for collecting sum at the top.

4.3 DEFINITION. We define an extended set of basic terms $B_{\sqcup}$ by adding the following clauses to definition 2.9.

i. $B \subseteq B_{\sqcup}$

ii. $t, s \in B_{\sqcup} \Rightarrow t \sqcup s \in B_{\sqcup}$

An example of an extended basic term: $((a \cdot (b \sqcup (c + d))) + b) \sqcup (a \cdot (b \sqcup c))$.

4.4 THEOREM: Let $S$ be a closed term. Then there is a basic term $t$ such that $\text{BPA}_{\sqcup} \vdash t = S$.

SKETCH OF PROOF: By term rewrite analysis. We do this in two stages. First, we rewrite using the rules in table 11 below. This gives us a $\sqcup$-normal form, i.e. a term where all $\sqcup$-operators appear at the top level only (i.e. there is no operator other than $\sqcup$ above $\sqcup$ in the parse tree). Then, on the subterms below the $\sqcup$ operators, we use the rewrite rules of table 5, in 2.10.

| $(X \sqcup Y) \cdot Z \rightarrow X \cdot Z \sqcup Y \cdot Z$ | RCAC4 |
| $X \cdot (Y \sqcup Z) \rightarrow X \cdot Y \sqcup X \cdot Z$ | RCAC5 |
| $(X \sqcup Y) + Z \rightarrow (X + Z) \sqcup (Y + Z)$ | RCAC6 |
| $X + (Y \sqcup Z) \rightarrow (X + Y) \sqcup (X + Z)$ | RCAC6' |
| $(X \sqcup Y) \sqcap Z \rightarrow (X \sqcap Z) \sqcup (Y \sqcap Z)$ | RPAC6 |
| $X \sqcap (Y \sqcup Z) \rightarrow (X \sqcap Y) \sqcup (X \sqcap Z)$ | RPAC6' |

TABLE 11. Rewrite system.

4.5 LEMMA: For all closed $\text{BPA}_{\sqcup}$-terms $t$ we have $\text{BPA}_{\sqcup} \vdash t \leq t + t$.

PROOF: By 4.4, we may suppose $t \in B_{\sqcup}$. We use induction on the structure of basic terms. All cases except the following are handled in lemma 2.5. Suppose the lemma holds for $x, y$, then:

$$x \sqcup y \leq x \sqcup y \sqcup (x + y) \sqcup (y + x) \leq (x + x) \sqcup (y + x) \sqcup (x + y) \sqcup (y + y) = (x \sqcup (x \sqcup y)) \sqcup (y + (x \sqcup y)) = (x \sqcup y) + (x \sqcup y).$$
4.6 THEOREM (COMPLETENESS). Let $s, t$ be closed terms. Then $[s] \leftrightarrow [t] \Rightarrow \text{BPA}_U \vdash s \leq t$.

PROOF: Omitted.

4.7 ADDITIONAL OPERATORS.

We show the axioms for $\text{ACP}_U$. The ones for $\text{PA}_U$ can be obtained easily from these. We see that the full expansion theorem is no longer valid. The way to expand the merge is to first get all collecting sums to the top level, factor them out, and then use the full expansion theorem on the connected subterms.

<table>
<thead>
<tr>
<th>$a \mid b = b \mid a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(a \mid b) \mid c = a \mid (b \mid c)$</td>
</tr>
<tr>
<td>$a \mid \delta = \delta$</td>
</tr>
<tr>
<td>$X \parallel Y \leq X \parallel Y + Y \parallel X + X \parallel Y$</td>
</tr>
<tr>
<td>$a \parallel X = a \cdot X$</td>
</tr>
<tr>
<td>$a \cdot X \parallel Y = a \cdot (X \parallel Y)$</td>
</tr>
<tr>
<td>$(X \boxdot Y) \parallel Z \leq X \parallel Z \boxdot Y \parallel Z$ for $\boxdot \in {+, \oplus}$</td>
</tr>
<tr>
<td>$(X \cup Y) \parallel Z = X \parallel Z \cup Y \parallel Z$</td>
</tr>
<tr>
<td>$X \parallel (Y \cup Z) = X \parallel Y \cup X \parallel Z$</td>
</tr>
<tr>
<td>$a \cdot X \parallel b = (a \mid b) \cdot X$</td>
</tr>
<tr>
<td>$a \parallel b \cdot X = (a \mid b) \cdot X$</td>
</tr>
<tr>
<td>$a \cdot X \parallel b \cdot Y = (a \mid b) \cdot (X \parallel Y)$</td>
</tr>
<tr>
<td>$(X \boxdot Y) \parallel Z \leq X \parallel Z \boxdot Y \parallel Z$ for $\boxdot \in {+, \oplus}$</td>
</tr>
<tr>
<td>$X \parallel (Y \boxdot Z) \leq X \parallel Y \boxdot X \parallel Z$ for $\boxdot \in {+, \oplus}$</td>
</tr>
<tr>
<td>$(X \cup Y) \parallel Z = X \parallel Z \cup Y \parallel Z$</td>
</tr>
<tr>
<td>$X \parallel (Y \cup Z) = X \parallel Y \cup X \parallel Z$</td>
</tr>
<tr>
<td>$\partial_H(a) = a$ if $a \in H$</td>
</tr>
<tr>
<td>$\partial_H(a) = \delta$ if $a \in H$</td>
</tr>
<tr>
<td>$\partial_H(X \boxdot Y) = \partial_H(X) \boxdot \partial_H(Y)$ for $\boxdot \in {+, \oplus, \cup, \oplus}$</td>
</tr>
</tbody>
</table>

TABLE 12. Axioms for $\text{ACP}_U$.

5. CONCLUSION.

We have extended ACP, in a setting of bisimulation semantics, with two additional choice operators: partial sum and collecting sum. This serves to bridge the gap between ACP and TCSP: we can combine TCSP's distinction between choice and non-determinism with the operational structure of bisimulations.

Further we found that the introduction of partial choice and collecting choice allows us, in some cases, to express safety properties of systems without the use of silent steps and to verify safety
properties in a setting in which no assumption about fairness (or unfairness) has been made. These verifications take place in the model; an algebraic style of verification has not been proposed here. Avoiding silent steps as well as fairness considerations are virtues of CSP that we consider advantages indeed. We think we have extended ACP in such a way that these virtues have become available in the ACP setting as well.

We found that it is the partial sum operator, that extends nicely to a probabilistic sum operator. Thus, the theory presented here forms a possible basis for a theory involving probabilistic choice together with other forms of choice. This allows one to describe systems where we know the probability of some choices, but not of all choices.

REFERENCES.
| Computing Science Notes | Department of Mathematics and Computing Science  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In this series appeared:</strong></td>
<td><strong>Eindhoven University of Technology</strong></td>
</tr>
</tbody>
</table>
| 91/02 R.P. Nederpelt  
H.C.M. de Swart | Implication. A survey of the different logical analyses "if...,then...", p. 26. |
| 91/03 J.P. Katoen  
L.A.M. Schoenmakers | Parallel Programs for the Recognition of P-invariant Segments, p. 16. |
| 91/04 E. v.d. Sluis  
| 91/05 D. de Reus | An Implementation Model for GOOD, p. 18. |
| 91/06 K.M. van Hee | SPECIFICATIEMETHODEN, een overzicht, p. 20. |
| 91/07 E.Poll | CPO-models for second order lambda calculus with recursive types and subtyping, p. 49. |
| 91/10 R.C.Backhouse  
P.J. de Bruin  
P. Hoogendijk  
G. Malcolm  
E. Voermans  
| 91/11 R.C. Backhouse  
P.J. de Bruin  
G. Malcolm  
E. Voermans  
J. van der Woude | Relational Catamorphism, p. 31. |
| 91/12 E. van der Sluis | A parallel local search algorithm for the travelling salesman problem, p. 12. |
| 91/14 P. Lemmens | The PDB Hypermedia Package. Why and how it was built, p. 63. |
| 91/15 A.T.M. Aerts  
<p>| 91/16 A.J.J.M. Marcelis | An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25. |</p>
<table>
<thead>
<tr>
<th>Page</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>91/18</td>
<td>Rik van Geldrop</td>
<td>Transformational Query Solving, p. 35.</td>
</tr>
<tr>
<td>91/19</td>
<td>Erik Poll</td>
<td>Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.</td>
</tr>
<tr>
<td>91/23</td>
<td>K.M. van Hee, L.J. Somers, M. Voorhoeve</td>
<td>Z and high level Petri nets, p. 16.</td>
</tr>
<tr>
<td>91/24</td>
<td>A.T.M. Aerts, D. de Reus</td>
<td>Formal semantics for BRM with examples, p. 25.</td>
</tr>
<tr>
<td>91/25</td>
<td>P. Zhou, J. Hooman, R. Kuiper</td>
<td>A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and completeness, p. 52.</td>
</tr>
<tr>
<td>91/27</td>
<td>F. de Boer, C. Palamidessi</td>
<td>Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.</td>
</tr>
<tr>
<td>91/28</td>
<td>F. de Boer</td>
<td>A compositional proof system for dynamic process creation, p. 24.</td>
</tr>
<tr>
<td>91/30</td>
<td>J.C.M. Baeten, F.W. Vaandrager</td>
<td>An Algebra for Process Creation, p. 29.</td>
</tr>
<tr>
<td>91/31</td>
<td>H. ten Eikelder</td>
<td>Some algorithms to decide the equivalence of recursive types, p. 26.</td>
</tr>
<tr>
<td>91/33</td>
<td>W. v.d. Aalst</td>
<td>The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.</td>
</tr>
<tr>
<td>91/34</td>
<td>J. Coenen</td>
<td>Specifying fault tolerant programs in deontic logic, p. 15.</td>
</tr>
<tr>
<td>Year/Seq</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>92/01</td>
<td>J. Coenen, J. Zwiers, W.-P. de Roever</td>
<td>A note on compositional refinement, p. 27.</td>
</tr>
<tr>
<td>92/02</td>
<td>J. Coenen, J. Hooman</td>
<td>A compositional semantics for fault tolerant real-time systems, p. 18.</td>
</tr>
<tr>
<td>92/03</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Real space process algebra, p. 42.</td>
</tr>
<tr>
<td>92/05</td>
<td>J.P.H.W.v.d.Eijnde</td>
<td>Conservative fixpoint functions on a graph, p. 25.</td>
</tr>
<tr>
<td>92/06</td>
<td>J.C.M. Baeten, J.A. Bergstra</td>
<td>Discrete time process algebra, p.45.</td>
</tr>
<tr>
<td>92/07</td>
<td>R.P. Nederpelt</td>
<td>The fine-structure of lambda calculus, p. 110.</td>
</tr>
<tr>
<td>92/10</td>
<td>P.M.P. Rambags</td>
<td>Composition and decomposition in a CPN model, p. 55.</td>
</tr>
<tr>
<td>92/13</td>
<td>F. Kamareddine</td>
<td>Set theory and nominalisation, Part II, p.22.</td>
</tr>
<tr>
<td>92/14</td>
<td>J.C.M. Baeten</td>
<td>The total order assumption, p. 10.</td>
</tr>
<tr>
<td>92/15</td>
<td>F. Kamareddine</td>
<td>A system at the cross-roads of functional and logic programming, p.36.</td>
</tr>
<tr>
<td>92/16</td>
<td>R.R. Seljéc</td>
<td>Integrity checking in deductive databases; an exposition, p.32.</td>
</tr>
<tr>
<td>92/17</td>
<td>W.M.P. van der Aalst</td>
<td>Interval timed coloured Petri nets and their analysis, p. 20.</td>
</tr>
<tr>
<td>92/18</td>
<td>R.Nederpelt, F. Kamareddine</td>
<td>A unified approach to Type Theory through a refined lambda-calculus, p. 30.</td>
</tr>
<tr>
<td>92/20</td>
<td>F.Kamareddine</td>
<td>Are Types for Natural Language? P. 32.</td>
</tr>
</tbody>
</table>
92/21 F.Kamareddine
Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.

92/22 R. Nederpelt
A useful lambda notation, p. 17.
F.Kamareddine

92/23 F.Kamareddine
Nominalization, Predication and Type Containment, p. 40.
E.Klein

92/24 M.Codish
Bottom-up Abstract Interpretation of Logic Programs, p. 33.
D.Dams
Eyal Yardeni

92/25 E.Poll
A Programming Logic for F0, p. 15.

92/26 T.H.W.Beelen
A modelling method using MOVIE and SimCon/ExSpect, p. 15.
W.J.J.Stut
P.A.C.Verkoulen

92/27 B. Watson
A taxonomy of keyword pattern matching algorithms, p. 50.
G. Zwaan

93/01 R. van Geldrop
Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
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
Deterministic and randomized local search, p. 78.
J.H.M. Korst
P.J. Zwietering

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

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
<table>
<thead>
<tr>
<th>Volume</th>
<th>Author(s)</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/16</td>
<td>H. Schepers, J. Hooman</td>
<td>A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27</td>
<td></td>
</tr>
<tr>
<td>93/17</td>
<td>D. Alstein, P. van der Stok</td>
<td>Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.</td>
<td></td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.</td>
<td></td>
</tr>
<tr>
<td>93/19</td>
<td>G-J. Houben</td>
<td>The Design of an Online Help Facility for ExSpect, p.21.</td>
<td></td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
<td>A Typechecker for Bijective Pure Type Systems, p. 28.</td>
<td></td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
<td>Relational Algebra and Equational Proofs, p. 23.</td>
<td></td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll and Paula Severi</td>
<td>Pure Type Systems with Definitions, p. 38.</td>
<td></td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
<td>Multi-dimensional Petri nets, p. 25.</td>
<td></td>
</tr>
<tr>
<td>93/27</td>
<td>T. Kloks and D. Kratsch</td>
<td>Finding all minimal separators of a graph, p. 11.</td>
<td></td>
</tr>
<tr>
<td>93/28</td>
<td>F. Kamareddine and R. Nederpelt</td>
<td>A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.</td>
<td></td>
</tr>
<tr>
<td>93/29</td>
<td>R. Post and P. De Bra</td>
<td>GOLD, a Graph Oriented Language for Databases, p. 42.</td>
<td></td>
</tr>
<tr>
<td>93/30</td>
<td>J. Deogun, T. Kloks, D. Kratsch, H. Müller</td>
<td>On Vertex Ranking for Permutation and Other Graphs, p. 11.</td>
<td></td>
</tr>
<tr>
<td>93/31</td>
<td>W. Körver</td>
<td>Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.</td>
<td></td>
</tr>
</tbody>
</table>
93/33  L. Loyens and J. Moonen  
ILIAS, 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.

93/40  P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein  

93/41  A. Bijlsma  
Temporal operators viewed as predicate transformers, p. 11.

93/42  P.M.P. Rambags  
Automatic Verification of Regular Protocols in P/T Nets, p. 23.

93/43  B.W. Watson  
A taxonomy of finite automata construction algorithms, p. 87.

93/44  B.W. Watson  
A taxonomy of finite automata minimization algorithms, p. 23.

93/45  E.J. Luit J.M.M. Martin  
A precise clock synchronization protocol,p.

93/46  T. Kloks D. Kratsch J. Spinrad  

93/47  W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky  

93/48  R. Gerth  
Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01  P. America
       M. van der Kammen
       R.P. Nederpelt
       O.S. van Roosmalen
       H.C.M. de Swart

The object-oriented paradigm, p. 28.

94/02  F. Kamareddine
       R.P. Nederpelt

Canonical typing and Π-conversion, p. 51.

94/03  L.B. Hartman
       K.M. van Hee

Application of Markov Decision Processes to Search
       Problems, p. 21.

94/04  J.C.M. Baeten
       J.A. Bergstra

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

94/05  P. Zhou
       J. Hooman

Formal Specification and Compositional Verification of
       an Atomic Broadcast Protocol, p. 22.

94/06  T. Basten
       T. Kunz
       J. Black
       M. Coffin
       D. Taylor

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

94/07  K.R. Apt
       R. Bol


94/08  O.S. van Roosmalen

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