Process algebra with feedback

Eindhoven University of Technology

Department of Mathematics and Computing Science

Process Algebra with Feedback

by

J.C.M. Baeten, J.A. Bergstra and
Gh. Ştefănescu

94/30

Computing Science Note 94/30
Eindhoven, July 1994

COMPUTING SCIENCE NOTES

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

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Process Algebra with Feedback

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

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
janb@fwi.uva.nl

Gh. Ştefănescu
Institute of Mathematics of the Romanian Academy
P.O.Box 1-764, 70700 Bucharest, Romania
ghstef@imar.ro

We consider process graphs over a set of pins, i.e. with multiple entries and exits. On process graphs modulo bisimulation, we can define all standard process algebra operators plus the feedback operator from flowchart theory. We provide a complete axiomatisation for finite processes. Considering the one-point pin structure, we get back standard process algebra.

1980 Mathematics Subject Classification (1985 revision): 68Q55, 68Q10, 68Q45.
1987 CR Categories: F.1.2, D.3.1, F.3.1, D.1.3.
Key words & Phrases: process algebra, feedback, pin.
Note: The research of the first two authors was supported in part by ESPRIT basic research action 7166, CONCUR2.

1. INTRODUCTION.

Semantics of process theory is often given in terms of graphs. The process graphs considered usually have exactly one entry and exactly one exit. In [BES94], this was adapted to allow for multiple entries and multiple exits. The resulting model was used to model key constructs of ACP and of the algebra of flownomials [STE90], especially the feedback construct. Feedback is a looping or iteration construct used in e.g. flowcharts. Another iteration construct is Kleene star, that was extensively studied in the setting of ACP in [BEBP94]. Here, we generalize and extend the results of [BES94].

We consider process graphs that have interior states and so-called *pins*, connections to the environment (a pin is an external connection of a chip, the name is used for the external connections of a Petri net in [EXS92]). A pin can be an entry, an exit, or both. We obtain a full theory of ACP with feedback operator, which generalises ACP in the sense that it weakens the axioms (e.g., in general, parallel composition is neither commutative nor associative). But if we add structure to the set of pins, we obtain an algebra of which the original ACP is a subalgebra of a reduced model.

In the paper, we first formulate a graph model (graphs that can have several pins) for the full syntax of ACP. Pin names are used to distinguish different entries or exits. Then, we divide out bisimulation, so that we obtain a bisimulation model. For this bisimulation model, we give a complete axiomatisation of finite processes, and we analyse the algebra obtained. Then, we illustrate the expressive power of the general formalism, and sketch the extension of the theory with silent steps and abstraction.

More about feedback and flowchart theories can be found in [BAWM94], [BAR92], [BLE93], [CAS90, 92], [MIL94], [STE87a, 87b], [STA92].

2. PROCESS GRAPHS OVER A SET OF PINS.

We introduce process graphs over a set of edge labels and a set of pin names. Edge labels are, as usual, taken from a finite set of primitive actions A. This set A is a parameter of the theory. Pin names are taken from a set V, also a parameter of the theory. In order to be able to define parallel composition, we need a pairing construct on V. Therefore, V must be closed under pairing, i.e. there is a coding $)\!(: V \times V \to V$. In the general theory, we assume nothing further about the set V. Later on, we consider special cases, by taking specific instances of V. We call a set V that is closed under pairing and maybe has additional structure a *pin structure*.

3.1 DEFINITION. We introduce a set of process graphs over A and V, $\mathcal{G}(A,V)$, as follows. A *process graph* g is a quadruple $\langle S, \to, I, O\rangle$ where

- S is the set of *(interior) states*. We abstract from the names of the states, i.e. we consider process graphs modulo isomorphism of states. Always, states are disjoint from pins.
- $I \subseteq V$ is the set of *input* or *entry* pins
- $O \subseteq V$, the set of *output* or *exit* pins
- $\to \subseteq (I \cup S) \times A \times (O \cup S)$ is the *transition relation*

We write $s \xrightarrow{a} t$ for $\langle s,a,t\rangle \in \to$. We refer to the four components of a process graph g by $S(g)$, $\to(g)$, $I(g)$ and $O(g)$, respectively. $I(g) \cup O(g)$ is the set of pins of g.

3.2 DEFINITION. We define several operators on process graphs (modulo isomorphism of states). We postpone the treatment of parallel composition to a later section. Here, we define atomic actions, alternative composition, sequential composition and feedback, together with some auxiliary operators.

First, we consider the constants.

1. Atomic actions. For each $a \in A$ and $p,q \in V$ we have a constant $p \to a \to q$. In [BES94], we find the notation a_p^q for $p \to a \to q$. The notation $p \to a \to q$ is already used in [BER89]. The interpretation of this process is the graph with no states, one edge, $p \xrightarrow{a} q$, and $I = \{p\}$, $O = \{q\}$. We picture this process on the left hand side of fig. 1. We indicate an entry pin by a small unlabeled incoming arrow, an exit pin by a small outgoing unlabeled arrow. Note that it is possible that $p=q$, but nevertheless, also in this case the intuition is that action a can only be executed once.

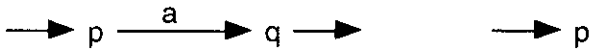

FIGURE 1. Constants.

2. Empty graph. We have a constant $\emptyset$ that stands for the empty graph, i.e. $\emptyset = \langle \emptyset,\emptyset,\emptyset,\emptyset \rangle$. Abusing notation, we usually write $\emptyset$ instead of $\emptyset$. The notation $\emptyset$ was also used in [BES94]. This process behaves as the deadlocked process δ with respect to alternative and sequential composition, but not with respect to parallel composition.

Next, we consider four operators used to manipulate pins.

3. Entry operator. Let a process graph g be given, and let $p \in V$. Now $p \rightarrow g$ has states $S(g)$, each edge $s \xrightarrow{a} t$ in g with $s \in I(g)$ is replaced by an edge $p \xrightarrow{a} t$, and furthermore has as only entry p and has the same exits as g. As an example, we depict graph $p \rightarrow \emptyset$ on the right hand side of fig. 1. This process behaves like ACP's δ with respect to sequential and parallel composition, but not, in general, with respect to alternative composition.

4. Exit operator. Let a process graph g be given, and let $p \in V$. Now $g \rightarrow p$ has states $S(g)$, each edge $s \xrightarrow{a} t$ in g with $t \in O(g)$ is replaced by an edge $s \xrightarrow{a} p$, and furthermore has the same entries as g and has as only exit p.

5. Initialisation operator. Let a process graph g be given, and let $p \in V$. $p \gg g$ has the same edges and exits as g. The set of entries becomes $I(g) \cap \{p\}$. States are all states of g, plus in addition other entries than p (if they exist).

6. Exiting operator. Let a process graph g be given, and let $p \in V$. $g \gg p$ has the same edges and entries as g. The set of exits becomes $O(g) \cap \{p\}$. States are all states of g, plus in addition other exits than p (if they exist).

Now, we consider alternative and sequential composition.

7. Alternative composition. Let process graphs g,h be given. Assume that the set of states of g is disjoint from the set of states of h (since we consider process graphs modulo state isomorphism, we can always ensure that this is the case). $g+h$ is obtained by taking the union of the states, the edges, the entries and the exits. As an example, we show $(p \rightarrow a \rightarrow p) + (q \rightarrow b \rightarrow q) + (p \rightarrow c \rightarrow q)$ in fig. 2. We picture both p and q twice, in order to emphasise the different roles of entries and exits.

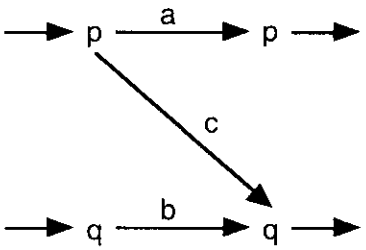

FIGURE 2. Alternative composition.

8. Sequential composition. Let process graphs g,h be given. Again assume that the set of states of g is disjoint from the set of states of h. We need to match exits of g with entries of h, but we need to distinguish these from entries of g and exits of h. In order to achieve this we take for each $p \in O(g) \cup I(h)$ a fresh state p^* (i.e. a name not appearing as the name of any state of g or h). The set of interior states of $g \cdot h$ is $S(g) \cup S(h) \cup \{p^* : p \in O(g) \cup I(h)\}$, the set of edges is obtained from the union of the edge sets of g and h by replacing each edge in g of the form $s \xrightarrow{a} p$ with $p \in O(h)$ by an edge $s \xrightarrow{a} p^*$ and replacing each edge in h of the form $p \xrightarrow{a} s$ with $p \in I(h)$ by an edge $p^* \xrightarrow{a} s$. The set of entries is I(g), the set of exits O(h). As an example, we show $((p \rightarrow a \rightarrow p) + (p \rightarrow b \rightarrow q)) \cdot (p \rightarrow c \rightarrow q)$ in fig. 3. Interior states are pictured as circles, their names are left out, since they do not matter (we consider graph isomorphism classes). Note that for all g and all $p \in V$, $g \cdot (p \rightarrow \varnothing) = g \cdot \varnothing$, and this graph has no exits.

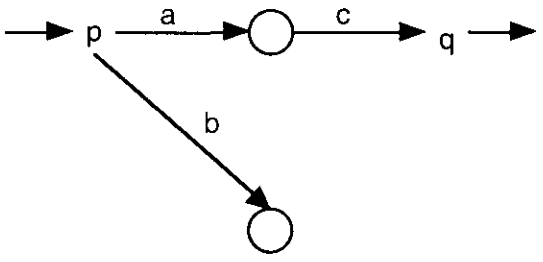

FIGURE 3. Sequential composition.

Finally, we consider feedback. We also define two auxiliary operators, used in the axiomatisation of feedback.

9. Feedback. Let process graph g be given, and let $p,q \in V$. The set of states of $g \uparrow_p^q$ is $S(g) \cup \{r\}$, for r a fresh name, the set of edges is obtained from $\rightarrow(g)$ by replacing each edge $q \xrightarrow{a} t$ in g by an edge $r \xrightarrow{a} t$, and replacing each edge $v \xrightarrow{a} p$ by an edge $v \xrightarrow{a} r$. Further, the entry set is I(g) - {q}, the exit set is O(g) - {p}. As an example, we show $((p \rightarrow a \rightarrow s) + (s \rightarrow b \rightarrow s) + (s \rightarrow c \rightarrow q)) \uparrow_s^s$ in fig. 4.

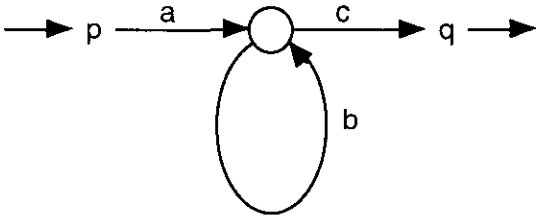

FIGURE 4. Feedback.

10. Pre-feedback. Process graph $g \downarrow_p^q$ is just like $g \uparrow_p^q$ except that we keep all edges $q \xrightarrow{a} t$, and the entry set is I(g).

11. Linking composition. Let process graphs g,h be given, and let $p,q \in V$ be given. We can assume that the state set of g is disjoint from states of h. $g \bullet_p^q h$ has as interior state set $S(g) \cup S(h) \cup \{r\}$, for r a fresh name. The set of edges is obtained by replacing each edge $q \xrightarrow{a} t$ in h by an edge $r \xrightarrow{a} t$, and

replacing each edge $v \xrightarrow{a} p$ in g by an edge $v \xrightarrow{a} r$. Furthermore, all edges $s \xrightarrow{a} t$ in h with $s \in I(h)$, $s \neq q$ are omitted. The graph has as entries I(g) and has as exits $(O(g) - \{p\}) \cup O(h)$.

3. BISIMULATION.

We look at the familiar notion of bisimulation in the present setting. To this end, consider the following definition.

3.1 DEFINITION. We define the familiar notion of bisimulation on process graphs with pins. Let $g,h \in \mathcal{G}(A,V)$ be given with the same entries, i.e. I(g) = I(h). A relation R between states of g and states of h is called a *bisimulation* if:

1. if $p \in I(g)$ and $p \xrightarrow{a} t$ in g, where $t \in S(g)$, then there is $v \in S(h)$ with $p \xrightarrow{a} v$ in h and R(t, v);
2. if $p \in I(g)$ and $p \xrightarrow{a} q$ in g, where $q \in O(g)$, then $q \in O(h)$ and $p \xrightarrow{a} q$ in h;
3. if $p \in I(h)$ and $p \xrightarrow{a} v$ in h, where $v \in S(h)$, then there is $t \in S(g)$ with $p \xrightarrow{a} t$ in g and R(t, v);
4. if $p \in I(h)$ and $p \xrightarrow{a} q$ in h, where $q \in O(h)$, then $q \in O(g)$ and $p \xrightarrow{a} q$ in g;
5. if R(s, t) and $s \xrightarrow{a} s'$ in g, where $s' \in S(g)$, then there is $t' \in S(h)$ such that $t \xrightarrow{a} t'$ in h and R(s', t');
6. if R(s, t) and $s \xrightarrow{a} q$ in g, where $q \in O(g)$, then $q \in O(h)$ and $t \xrightarrow{a} q$ in h;
7. if R(s, t) and $t \xrightarrow{a} t'$ in h, where $t' \in S(h)$, then there is $s' \in S(g)$ such that $s \xrightarrow{a} s'$ in g and R(s', t');
8. if R(s, t) and $t \xrightarrow{a} q$ in h, where $q \in O(h)$, then $q \in O(g)$ and $t \xrightarrow{a} q$ in g.

We say g,h are *bisimilar*, $g \leftrightarrows h$, if I(g) = I(h) and there is an bisimulation between g and h. Note that, different from [BES94], bisimulating processes do not need to have the same exits.

As usual, bisimulation is an equivalence relation on process graphs. We can divide out this equivalence, and obtain the algebras $\mathcal{G}(A,V)/\leftrightarrows$. We will also find that bisimulation is a congruence for all operators defined in section 2, and so we can define these operators on these algebras.

3.2 STRUCTURED OPERATIONAL SEMANTICS.

We can also give the semantics by means of SOS rules. In the following table we have $a,b,c \in A$, $p,q,r,s \in V$, $x,x',y,y' \in P$. We define a transition relation as a subset of $P \times (V \times A) \times P$ and terminating transitions as a subset of $P \times (V \times A) \times V$. Moreover, entry set membership is a subset of $V \times P$, that holds of p and x whenever p is an entry of x, we write $p \in i(x)$. Considering the terminating transitions and entry set membership as predicates, we find that the rules satisfy the *path* format of [BAV93]. Thus, the standard definition of bisimulation (call this $\leftrightarrows_{SOS}$) on these transition system yields a congruence. It is not hard to prove the following theorem, for all closed terms s,t (where $[\![.]\!]$ is the interpretation defined in section 2):

3.3 THEOREM: For all closed terms s,t: $s \leftrightarrows_{SOS} t \Leftrightarrow [\![s]\!] \leftrightarrows [\![t]\!]$.

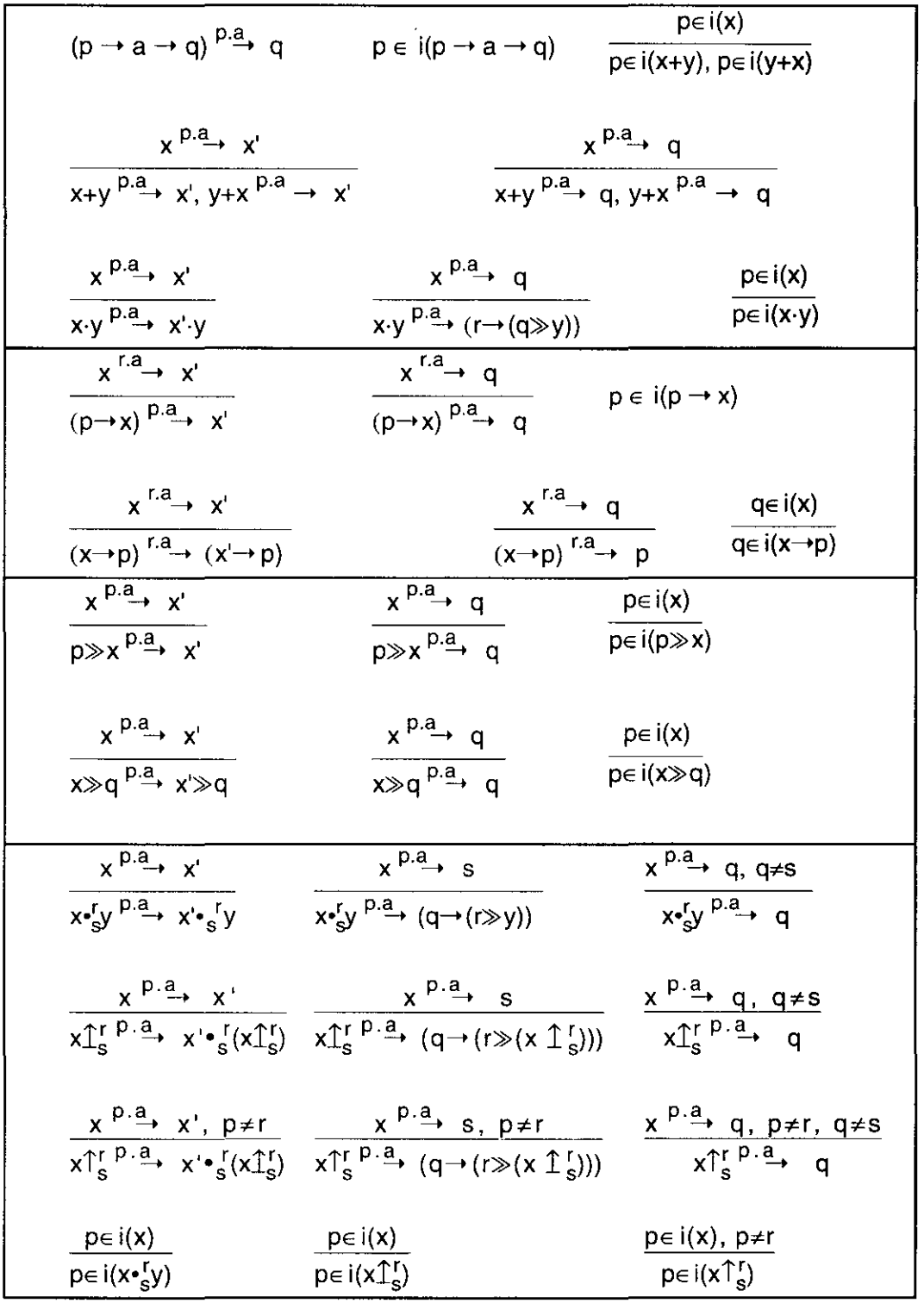

TABLE 1. Structured operational semantics

4. ALGEBRA.

In this section, we axiomatise bisimulation equivalence on process graphs over a set of pins.

4.1 SYNTAX.

Sorts:

P	sort of processes
V	sort of pins

Constants:

$p \rightarrow a \rightarrow q \in P$	atomic action (for each $a \in A$, $p,q \in V$)
$\varnothing \in P$	empty process ($\varnothing \notin A$)

Functions:

$\rightarrow: V \times P \rightarrow P$	entry operator
$\rightarrow: P \times V \rightarrow P$	exit operator
$\gg: V \times P \rightarrow P$	initialisation operator
$\gg: P \times V \rightarrow P$	exiting operator
$\bullet: P \times V \times V \times P \rightarrow P$	linking composition
$\mathbin{\mathbb{1}}: P \times V \times V \rightarrow P$	pre-feedback
$\uparrow: P \times V \times V \rightarrow P$	feedback

The signature without the last three operators is the signature of BPApin(A,V), Basic Process Algebra over a set of actions A and a pin structure V. The full signature is the signature of BPA$\uparrow$pin(A,V), BPA with feedback over actions A and pins V.

4.2 AXIOMS.

First of all, we present the axioms of BPA$_\delta$, replacing δ by $\varnothing$. Always, $X,Y,Z \in P$. Sequential composition always has strongest binding power, alternative composition always has weakest binding power.

$X + Y = Y + X$	A1
$(X+ Y) + Z = X + (Y + Z)$	A2
$X + X = X$	A3
$(X + Y)\cdot Z = X\cdot Z + Y\cdot Z$	A4
$(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z)$	A5
$X + \varnothing = X$	A6$\varnothing$
$\varnothing\cdot X = \varnothing$	A7$\varnothing$

TABLE 2. BPA$_\varnothing$.

Next, entries and exits. Always, $a \in A$, $p,q \in V$.

$$p \to (q \to a \to r) = p \to a \to r$$
$$p \to (q \to \emptyset) = p \to \emptyset$$
$$(p \to a \to q) \to r = p \to a \to r$$
$$\emptyset \to p = \emptyset$$
$$(p \to \emptyset) \to q = p \to \emptyset$$
$$p \to (X + Y) = p \to X + p \to Y$$
$$(X + Y) \to p = X \to p + Y \to p$$
$$p \to X \cdot Y = (p \to X) \cdot Y$$
$$X \cdot Y \to p = X \cdot (Y \to p)$$
$$X \cdot (p \to \emptyset) = X \cdot \emptyset$$

TABLE 3. Entries and exits.

Next, the initialisation and exiting operators.

$$p \gg \emptyset = \emptyset$$
$$p \gg (p \to a \to q) = p \to a \to q$$
$$p \gg (q \to a \to r) = \emptyset \qquad \text{if } p \neq q$$
$$p \gg (p \to \emptyset) = p \to \emptyset$$
$$p \gg (q \to \emptyset) = \emptyset \qquad \text{if } p \neq q$$
$$p \gg (X + Y) = p \gg X + p \gg Y$$
$$p \gg X \cdot Y = (p \gg X) \cdot Y$$

$$(p \to a \to q) \cdot X = (p \to a \to q) \cdot (q \gg X)$$
$$(p \to a \to q) \cdot X = (p \to a \to r) \cdot (r \to (q \gg X))$$

$$\emptyset \gg p = \emptyset$$
$$(p \to a \to q) \gg q = p \to a \to q$$
$$(p \to a \to q) \gg r = (p \to a \to q) \cdot \emptyset \qquad \text{if } q \neq r$$
$$(p \to \emptyset) \gg q = p \to \emptyset$$
$$(X + Y) \gg p = X \gg p + Y \gg p$$
$$X \cdot Y \gg p = X \cdot (Y \gg p)$$

TABLE 4. Initialisation, exiting.

The axioms of tables 2-4 are the axioms of BPApin(A,V). The middle two axioms of table 4 show the nature of sequential composition. The first of the two expresses the idea, that a process following a process that exits in p, can only start in p. The second of the two expresses the idea, that the names of the interior states do not matter. On the basis of this second axiom, we allow ourselves to abbreviate $(p \to a \to q) \cdot (q \to b \to r)$ by $(p \to a \cdot b \to r)$, and similarly abbreviate

$$(p \to a \to q) \cdot \Big(\sum_{i \leq n} (q \to b_i \to r_i) \cdot x_i\Big) \quad \text{by} \quad (p \to a) \cdot \Big(\sum_{i \leq n} (b_i \to r_i) \cdot x_i\Big).$$

Finally, we get to the iteration operators. We use linking and pre-feedback in order to obtain an axiomatisation for feedback.

$$\varnothing \bullet_q^r Y = \varnothing$$
$$(p \to a \to q) \bullet_q^r Y = (p \to a \to r)\cdot Y$$
$$(p \to a \to s) \bullet_q^r Y = p \to a \to s \qquad \text{if } s \neq q$$
$$(p \to \varnothing) \bullet_q^r Y = p \to \varnothing$$
$$(X + Y) \bullet_p^q Z = X \bullet_p^q Z + Y \bullet_p^q Z$$
$$(X\cdot Y) \bullet_p^q Z = X\cdot(Y \bullet_p^q Z)$$
$$X \mathbin{\mathbb{1}}_p^q = X \bullet_p^q (X \mathbin{\mathbb{1}}_p^q)$$
$$X \uparrow_p^q = \sum_{r \in i(X)-\{q\}} r \gg (X \mathbin{\mathbb{1}}_p^q)$$

TABLE 5. Linking and feedback.

By convention, a sum over an empty set is equal to $\varnothing$. The last axiom uses the auxiliary operator i, that was also used in the SOS rules, determining the entry set of a process. It is axiomatised in the following table 6. The complete system of tables 2 through 6 is called BPA$\uparrow$pin(A,V). When these are clear from the context, we usually omit the parameters A,V, and talk about BPApin, BPA$\uparrow$pin.

$$i(\varnothing) = \varnothing$$
$$i(p \to a \to q) = \{p\}$$
$$i(p \to \varnothing) = \{p\}$$
$$i(X + Y) = i(X) \cup i(Y)$$
$$i(X\cdot Y) = i(X)$$

TABLE 6. Entry set.

5. THEORY.

5.1 DEFINITION. The set of basic (BPA$\uparrow$pin) terms is defined inductively:

i. $\varnothing$ is a basic term, and for all $p \in V$ $p \to \varnothing$ is a basic term;

ii. for all $p,q \in V$, $a \in A$ $p \to a \to q$ is a basic term;

iii. if t is a basic term with $i(t) = \{q\}$, and $p \in V$, $a \in A$, then $(p \to a \to q)\cdot t$ is a basic term;

iv. if t,s are basic terms, then $t + s$ is a basic term.

5.2 THEOREM (Elimination Theorem): For all closed BPApin terms s there is a basic term t such that BPApin $\vdash s = t$.

SKETCH OF PROOF: We use term rewrite analysis. Consider the term rewrite system obtained by orienting the following rules from left to right: A3-A7 of BPA_δ, all axioms of table 3 and all axioms of table 4 except the middle two. About the middle two axioms, we leave out the second, and have to make sure that the first is only used once for every sequential composition in a term. In order to achieve this, we mark the $\cdot$ sign on the left hand side, and duplicate all other rules involving sequential composition, by replacing each $\cdot$ by a marked $\cdot$. Basically, the same trick was used in [KLU93]. Further, we add the rule $(p \to \varnothing)\cdot X = p \to \varnothing$. Then, we prove the term rewriting system is terminating by using the lexicographic path ordering. Then, elimination follows since all normal forms are basic terms. Note that we can also eliminate linking composition.

5.3 THEOREM (Soundness Theorem): $\mathcal{G}(A,V)/\leftrightarrows \models BPA\uparrow pin$.
PROOF: Omitted.

5.4 THEOREM (Conservativity Theorem): $BPA\uparrow pin$ over a set of actions A is a conservative extension of BPA_δ (with $\varnothing$ instead of δ) over the set of actions $\{p \to a \to q : a \in A, p,q \in V\} \cup \{p \to \varnothing : p \in V\}$.
SKETCH OF PROOF: Since all our SOS rules are in path format, operational conservativity follows immediately (see [VER93]). Then the (equational) conservativity follows by a result of [VER93] since the axiomatisation of BPA_δ is sound and complete and the axiomatisation of $BPA\uparrow pin$ is sound (theorem 3.5).

5.5 THEOREM (Completeness Theorem): The axiomatisation of BPApin (so without iterative constructs!) is complete for the model $\mathcal{G}(A,V)/\leftrightarrows$.
SKETCH OF PROOF: By the general result of [VER93]. In addition to the ingredients of the previous proof, all we need is the elimination theorem.

5.6 DEFINITION: On $BPA\uparrow pin$, we can define projection operators π_n as follows ($n \geq 0$):

$\pi_n(\varnothing) = \varnothing$
$\pi_0(p \to a \to q) = p \to \varnothing$
$\pi_{n+1}(p \to a \to q) = p \to a \to q$
$\pi_n(p \to \varnothing) = p \to \varnothing$
$\pi_0((p \to a \to q)\cdot X) = p \to \varnothing$
$\pi_{n+1}((p \to a \to q)\cdot X) = (p \to a \to q)\cdot\pi_n(X)$
$\pi_n(X + Y) = \pi_n(X) + \pi_n(Y)$

TABLE 7. Projection.

5.7 THEOREM (Projection theorem): Let t be a closed $BPA\uparrow pin$ term, and let $n \geq 0$. Then $\pi_n(t)$ can be written as a basic term.
PROOF: Omitted.

5.8 THEOREM (Representation theorem): Let g be a regular process (an element of $\mathcal{G}(A,V)$ with finitely many states). Then there is a closed BPA$\uparrow$pin term t such that $[\![t]\!] \leftrightarrows g$.

SKETCH OF PROOF: An example is provided in 5.9 below. The general proof is along the same lines.

5.9 EXAMPLE: Consider the process graph g in fig. 5, with pins $\{p,q\}$ ($p \neq q$). Then there is no closed BPApin term t (so without iterative constructs!) such that $[\![t]\!] \leftrightarrows g$. To write this as a BPA$\uparrow$pin term (even up to isomorphism) is easy: take fresh names r,s and consider $((p \to a \to r) + (q \to a \to s) + (r \to b \to s) + (r \to c \to p) + (s \to c \to q))\uparrow^{r}_{r}\uparrow^{s}_{s}$

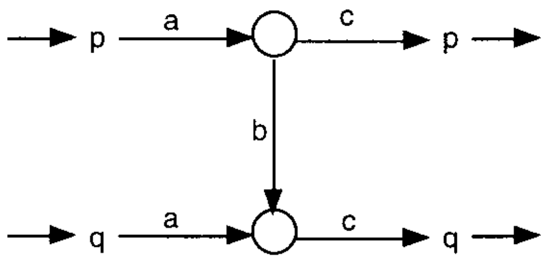

FIGURE 5.

6. PARALLEL COMPOSITION.

In this section, we extend the theory of the previous sections by parallel composition operators. Now we have a third parameter of the theory, the *communication function*. This is a partial, commutative and associative function $\gamma: A \times A \to A$.

6.1 GRAPH MODEL.

We add a number of operators.

1. <u>Left global state pairing</u>. Let a process graph g be given, and let $p \in V$. $p \rangle\!\dashv g$ has states $S(g)$, entries $\{p\} \times I(g)$ and exits $\{p\} \times O(g)$. It has same edges between states as g, and the pins of edges starting from an entry or ending in an exit are paired on the left with p.

2. <u>Right global state pairing</u>. Let a process graph g be given, and let $p \in V$. $g \vdash\!\langle p$ has states $S(g) \times \{p\}$, entries $I(g) \times \{p\}$ and exits $O(g) \times \{p\}$. It has same edges between states as g, and the pins of edges starting from an entry or ending in an exit are paired on the right with p.

3. <u>Parallel composition</u>. Let process graphs g,h be given. Assume that the state sets of g and h are disjoint. Take a set of fresh state names for each exit, i.e. assume there are sets $O^*(g) = \{p^* : p \in O(g)\}$, $O^*(h) = \{p^* : p \in O(h)\}$ of states not occurring elsewhere in g or h (this is needed in case an exit also occurs as an entry). The set of states of $g \| h$ is $((S(g) \cup I(g) \cup O^*(g)) \times (S(h) \cup I(h) \cup O^*(h))) - ((I(g) \times I(h)) \cup (O^*(g) \times O^*(h))$. The transition relation is given by:

a. for each state v in g, and each transition $t \xrightarrow{a} t'$ in h, there is a transition $v \rangle\!\!-\!\!\langle t \xrightarrow{a} v \rangle\!\!-\!\!\langle t'$ (here, we take $v \rangle\!\!-\!\!\langle t \xrightarrow{a} v \rangle\!\!-\!\!\langle t'^*$ if $t' \in O(h)$)

b. for each state t in h, and each transition $v \xrightarrow{a} v'$ in g, there is a transition $v \rangle\!\!-\!\!\langle t \xrightarrow{a} v' \rangle\!\!-\!\!\langle t$ here, we take $v \rangle\!\!-\!\!\langle t \xrightarrow{a} v'^* \rangle\!\!-\!\!\langle t$ if $v' \in O(g)$)

c. for each pair of transitions $v\rangle\!-\!\langle t \xrightarrow{a} v'\rangle\!-\!\langle t$ and $v\rangle\!-\!\langle t \xrightarrow{b} v\rangle\!-\!\langle t'$ such that $\gamma(a,b)$ is defined, say $\gamma(a,b) = c$, there is a transition $v\rangle\!-\!\langle t \xrightarrow{c} v'\rangle\!-\!\langle t'$ (again, v'^* and/or t'^* when needed)

The entries are $I(g) \times I(h)$, the exits are $O(g) \times O(h)$.

Note that the pairing function need not be commutative or associative. Thus, the parallel composition may not be commutative or associative. As an example, we show $(p \rightarrow a \rightarrow p) \parallel (q \rightarrow b \rightarrow q)$, where $\gamma(a,b) = c$, in fig. 6.

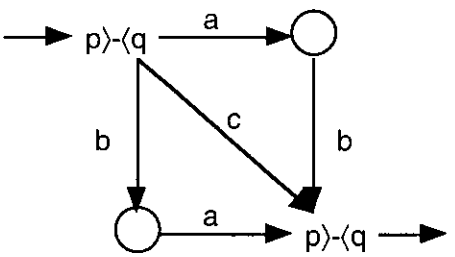

FIGURE 6.

4. Left merge. The graph of $g \between h$ has the same states as the graph of $g \parallel h$, the same pins and the same transitions except that the transitions $v\rangle\!-\!\langle t \xrightarrow{a} v'\rangle\!-\!\langle t'$ with v,t entries, and t' not an entry, are omitted.

5. Right merge. The graph of $g \between h$ has the same states as the graph of $g \parallel h$, the same pins and the same transitions except that the transitions $v\rangle\!-\!\langle t \xrightarrow{a} v'\rangle\!-\!\langle t'$ with v,t entries, and v' not an entry, are omitted.

6. Communication merge. The graph of $g \mid h$ has the same states as the graph of $g \parallel h$, the same pins and the same transitions except that the transitions $v\rangle\!-\!\langle t \xrightarrow{a} v'\rangle\!-\!\langle t'$ with v' or t' an entry are omitted.

7. Encapsulation. Let process graph g be given, and let $H \subseteq A$. The graph of $\partial_H(g)$ has the same states as the graph of g, the same pins and the same transitions except that all transitions $v \xrightarrow{a} t$ with $a \in H$ are omitted.

6.2 STRUCTURED OPERATIONAL SEMANTICS.

We show the SOS rules for the additional operators.

$$\frac{x \xrightarrow{r.a} x'}{p\rangle\!\dashv\!x \xrightarrow{p\rangle\!-\!\langle r.a} p\rangle\!\dashv\!x'} \qquad \frac{x \xrightarrow{r.a} q}{p\rangle\!\dashv\!x \xrightarrow{p\rangle\!-\!\langle r.a} p\rangle\!-\!\langle q} \qquad \frac{q \in i(x)}{p\rangle\!-\!\langle q \in i(p)\rangle\!\dashv\!x)}$$

$$\frac{x \xrightarrow{r.a} x'}{x\vdash\!\langle p \xrightarrow{r\rangle\!-\!\langle p.a} x'\vert\!-\!\langle p} \qquad \frac{x \xrightarrow{r.a} q}{x\vdash\!\langle p \xrightarrow{r\rangle\!-\!\langle p.a} q\rangle\!-\!\langle p} \qquad \frac{q \in i(x)}{q\rangle\!-\!\langle p \in i(x\vert\!-\!\langle p)}$$

$$\frac{p \in i(x),\ q \in i(y)}{p\rangle\!\dashv\langle q \in i(x \square y)} \qquad (\square = \|, \mathbb{L}, \mathbb{L}\!\!\mathbb{L}, |)$$

$$\frac{x \xrightarrow{p.a} x',\ r \in i(y)}{\begin{array}{l} x \| y \xrightarrow{p\rangle\!\dashv\langle r.a} x' \| (r \gg y),\ y \| x \xrightarrow{r\rangle\!\dashv\langle p.a} (r \gg y) \| x', \\ x \mathbb{L} y \xrightarrow{p\rangle\!\dashv\langle r.a} x' \| (r \gg y),\quad y \mathbb{L}\!\!\mathbb{L} x \xrightarrow{r\rangle\!\dashv\langle p.a} (r \gg y) \| x' \end{array}}$$

$$\frac{x \xrightarrow{p.a} q,\ r \in i(y)}{\begin{array}{l} x \| y \xrightarrow{p\rangle\!\dashv\langle r.a} q\rangle\!\dashv\langle (r \gg y),\ y \| x \xrightarrow{r\rangle\!\dashv\langle p.a} (r \gg y) | \dashv\langle q, \\ x \mathbb{L} y \xrightarrow{p\rangle\!\dashv\langle r.a} q\rangle\!\dashv\langle (r \gg y), y \mathbb{L}\!\!\mathbb{L} x \xrightarrow{r\rangle\!\dashv\langle p.a} (r \gg y) | \dashv\langle q \end{array}}$$

$$\frac{x \xrightarrow{p.a} x',\ y \xrightarrow{r.b} y',\ a \mid b = c \neq \delta}{x \| y \xrightarrow{p\rangle\!\dashv\langle r.c} x' \| y',\ x \mid y \xrightarrow{p\rangle\!\dashv\langle r.c} x' \| y'}$$

$$\frac{x \xrightarrow{p.a} x',\ y \xrightarrow{r.b} s,\ a \mid b = c \neq \delta}{\begin{array}{l} x \| y \xrightarrow{p\rangle\!\dashv\langle r.c} x' | \dashv\langle s,\ x \mid y \xrightarrow{p\rangle\!\dashv\langle r.c} x' | \dashv\langle s, \\ y \| x \xrightarrow{r\rangle\!\dashv\langle p.c} s\rangle\!\dashv\langle x',\ y \mid x \xrightarrow{p\rangle\!\dashv\langle r.c} s\rangle\!\dashv\langle x' \end{array}}$$

$$\frac{x \xrightarrow{p.a} q,\ y \xrightarrow{r.b} s,\ a \mid b = c \neq \delta}{x \| y \xrightarrow{p\rangle\!\dashv\langle r.c} q\rangle\!\dashv\langle s,\ x \mid y \xrightarrow{p\rangle\!\dashv\langle r.c} q\rangle\!\dashv\langle s}$$

$$\frac{x \xrightarrow{p.a} x',\ a \notin H}{\partial_H(x) \xrightarrow{p.a} \partial_H(x')} \qquad \frac{x \xrightarrow{p.a} q,\ a \notin H}{\partial_H(x) \xrightarrow{p.a} q} \qquad \frac{p \in i(x)}{p \in i(\partial_H(x))}$$

TABLE 8. Structured operational semantics.

6.3 ALGEBRA.

We repeat the added signature:

$\rangle\!\dashv: V \times P \to P$	left global state pairing
$\dashv\langle: P \times V \to P$	right global state pairing
$\square: P \times P \to P$	standard ACP operator, for $\square \in \{+, \cdot, \|, \mathbb{L}, \mid\}$
$\partial_H: P \to P$	standard ACP encapsulation, for $H \subseteq A$
$\mathbb{L}\!\!\mathbb{L}: P \times P \to P$	right-merge.

The following table shows axioms for the global state pairing operators. We need these in the axiomatisation of parallel composition.

$$p \rangle\!\dashv \varnothing = \varnothing$$
$$p \rangle\!\dashv (q \to \varnothing) = p\rangle\!\!-\!\!\langle q \to \varnothing$$
$$p \rangle\!\dashv (q \to a \to r) = p\rangle\!\!-\!\!\langle q \to a \to p\rangle\!\!-\!\!\langle r$$
$$p \rangle\!\dashv (X + Y) = p \rangle\!\dashv X + p \rangle\!\dashv Y$$
$$p \rangle\!\dashv X \cdot Y = (p \rangle\!\dashv X) \cdot (p \rangle\!\dashv Y)$$

$$\varnothing \vdash\!\langle p = \varnothing$$
$$(q \to \varnothing) \vdash\!\langle p = q\rangle\!\!-\!\!\langle p \to \varnothing$$
$$(q \to a \to r) \vdash\!\langle p = q\rangle\!\!-\!\!\langle p \to a \to r\rangle\!\!-\!\!\langle p$$
$$(X + Y) \vdash\!\langle p = X \vdash\!\langle p + Y \vdash\!\langle p$$
$$X \cdot Y \vdash\!\langle p = (X \vdash\!\langle p) \cdot (Y \vdash\!\langle p)$$

TABLE 9. Global state pairing.

Now we have the ingredients that together allow to axiomatise parallel composition. As parallel composition is not in general commutative, we need both a left-merge and a right-merge operator. Recall that we have a partial, commutative and associative communication function γ on A. The axioms of BPApin plus the axioms in tables 9 and 10 constitute the theory ACPpin. Likewise, BPA↑pin plus axioms in tables 9 and 10 yields ACP↑pin. On the basis of these axiomatisations, it is easy to obtain axiomatisations for theories without communication, PApin and PA↑pin.

$$X \| Y = X \mathbin{|\!\lfloor} Y + X \mathbin{\rfloor\!|} Y + X \mid Y$$
$$p \to a \mid b \to q = p \to \gamma(a,b) \to q \qquad \text{if } \gamma(a,b) \text{ defined}$$
$$p \to a \mid b \to q = p \to \varnothing \qquad \text{if } \gamma(a,b) \text{ not defined}$$

$$\varnothing \mathbin{|\!\lfloor} X = \varnothing$$
$$(p \to \varnothing) \mathbin{|\!\lfloor} X = \sum_{r \in i(X)} p\rangle\!\!-\!\!\langle r \to \varnothing$$
$$(p \to a \to q) \mathbin{|\!\lfloor} X = (\sum_{r \in i(X)} (p \to a \to q) \vdash\!\langle r) \cdot (q \rangle\!\dashv X)$$
$$(p \to a \to q) \cdot X \mathbin{|\!\lfloor} Y = (\sum_{r \in i(Y)} (p \to a \to q) \vdash\!\langle r) \cdot (X \| Y)$$
$$(X + Y) \mathbin{|\!\lfloor} Z = X \mathbin{|\!\lfloor} Z + Y \mathbin{|\!\lfloor} Z$$

$$X \mathbin{\rfloor\!|} \varnothing = \varnothing$$
$$X \mathbin{\rfloor\!|} (p \to \varnothing) = \sum_{r \in i(X)} r\rangle\!\!-\!\!\langle p \to \varnothing$$
$$X \mathbin{\rfloor\!|} (p \to a \to q) = (\sum_{r \in i(X)} r \rangle\!\dashv (p \to a \to q) \cdot (X \vdash\!\langle q)$$
$$X \mathbin{\rfloor\!|} (p \to a \to q) \cdot Y = (\sum_{r \in i(X)} r \rangle\!\dashv (p \to a \to q) \cdot (X \| Y)$$
$$X \mathbin{\rfloor\!|} (Y + Z) = X \mathbin{\rfloor\!|} Y + X \mathbin{\rfloor\!|} Z$$

$$\varnothing \mid X = \varnothing$$
$$X \mid \varnothing = \varnothing$$
$$(p \to \varnothing) \mid X = \sum_{r \in i(X)} p \rangle\!\!-\!\!\langle r \to \varnothing$$
$$X \mid (p \to \varnothing) = \sum_{r \in i(X)} r \rangle\!\!-\!\!\langle p \to \varnothing$$
$$(p \to a \to q) \mid (r \to b \to s) = p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle s$$
$$(p \to a \to q) \cdot X \mid (r \to b \to s) = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle (s) \cdot (X \vdash\!\langle s)$$
$$(p \to a \to q) \mid (r \to b \to s) \cdot X = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle (s) \cdot (q \rangle\!\!\dashv X)$$
$$(p \to a \to q) \cdot X \mid (r \to b \to s) \cdot Y = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle (s) \cdot (X \parallel Y)$$
$$(X + Y) \mid Z = X \mid Z + Y \mid Z$$
$$X \mid (Y + Z) = X \mid Y + X \mid Z$$

$$\partial_H(\varnothing) = \varnothing$$
$$\partial_H(p \to \varnothing) = p \to \varnothing$$
$$\partial_H(p \to a \to q) = p \to a \to q \qquad \text{if } a \notin H$$
$$\partial_H(p \to a \to q) = p \to \varnothing \qquad \text{if } a \in H$$
$$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$$
$$\partial_H(X \cdot Y) = \partial_H(X) \cdot \partial_H(Y)$$

TABLE 10. Parallel composition and encapsulation.

For the extensions defined in this section, we can obtain results similar to the results in section 5.

7. ABSTRACTION.

In this section we define abstraction and branching bisimulation for process algebra over a set of pins.

7.1 PROCESS GRAPHS.

Suppose we have a special constant τ, and we consider process graphs over $A \cup \{\tau\}$ and V. Again, we consider process graphs modulo state isomorphism. On an element of this domain of process graphs, the abstraction operator τ_I is simply defined by relabeling all edges with labels from $I \subseteq A$ by τ. We put $s \Rightarrow t$, for nodes s,t of a process graph, if t can be reached from s by doing a number of τ-steps (0 or more). The notion of branching bisimulation is defined as follows:

7.2 DEFINITION. Let $g,h \in \mathcal{G}(A \cup \{\tau\}, V)$ be given with $I(g) = I(h)$. A relation R between states of g and states of h is called a *rooted branching bisimulation* if, for all $a \in A \cup \{\tau\}$:

1. if $p \in I(g)$ and $p \xrightarrow{a} t$ in g, where $t \in S(g)$, then there is $v \in S(h)$ with $p \xrightarrow{a} v$ in h and $R(t, v)$;
2. if $p \in I(g)$ and $p \xrightarrow{a} q$ in g, where $q \in O(g)$, then $q \in O(h)$ and $p \xrightarrow{a} q$ in h;
3. if $p \in I(h)$ and $p \xrightarrow{a} v$ in h, where $v \in S(h)$, then there is $t \in S(g)$ with $p \xrightarrow{a} t$ in g and $R(t, v)$;
4. if $p \in I(h)$ and $p \xrightarrow{a} q$ in h, where $q \in O(h)$, then $q \in O(g)$ and $p \xrightarrow{a} q$ in g;

5. if $R(s, t)$ and $s \xrightarrow{a} s'$ in g, where $s' \in S(g)$, then *either* $a=\tau$ and $R(s',t)$ *or* there are $t', t'' \in S(h)$ such that $t \Rightarrow t' \xrightarrow{a} t''$ in h and $R(s, t')$, $R(s',t'')$ *or* there are $t' \in S(h)$, $q \in O(h)$ such that $t \Rightarrow t' \xrightarrow{a} q$ in h and $R(s, t')$ and s' is a q-semi-endpoint;

6. if $R(s, t)$ and $s \xrightarrow{a} q$ in g, where $q \in O(g)$, then $q \in O(h)$ and *either* $a=\tau$ and t is a q-semi-endpoint*or* there are $t', t'' \in S(h)$ such that $t \Rightarrow t' \xrightarrow{a} t''$ in h and $R(s, t')$, t'' is a q-semi-endpoint *or* there is $t' \in S(h)$ such that $t \Rightarrow t' \xrightarrow{a} q$ in h and $R(s, t')$;

7. if $R(s, t)$ and $t \xrightarrow{a} t'$ in h, where $t' \in S(h)$, then *either* $a=\tau$ and $R(s,t')$ *or* there are $s', s'' \in S(g)$ such that $s \Rightarrow s' \xrightarrow{a} s''$ in g and $R(s', t)$, $R(s'',t')$ *or* there are $s' \in S(g)$, $q \in O(g)$ such that $s \Rightarrow s' \xrightarrow{a} q$ in g and $R(s', t)$ and t' is a q-semi-endpoint;

8. if $R(s, t)$ and $t \xrightarrow{a} q$ in h, where $q \in O(h)$, then $q \in O(g)$ and *either* $a=\tau$ and s is a q-semi-endpoint*or* there are $s', s'' \in S(g)$ such that $s \Rightarrow s' \xrightarrow{a} s''$ in g and $R(s', t)$, s'' is a q-semi-endpoint *or* there is $s' \in S(g)$ such that $s \Rightarrow s' \xrightarrow{a} q$ in g and $R(s', t)$.

Here, we say that an interior node t is a q-*semi-endpoint* ($q \in V$) if $t \Rightarrow q$ and $t \xrightarrow{a} s$ implies $a=\tau$ and s is a q-(semi-)endpoint.

We say g,h are *rooted branching bisimilar*, $g \leftrightarrows_{rb} h$, if $I(g) = I(h)$ and there is a rooted branching bisimulation between g and h. Rooted branching bisimulation, as defined here, is an equivalence relation on process graphs. We can divide out this equivalence, and obtain the algebras $\mathcal{G}(A\cup\{\tau\},V)/\leftrightarrows_{rb}$. We will also find that rooted branching bisimulation is a congruence for all operators defined in sections 2 and 6.

7.3 Structured operational semantics.

The SOS rules for the abstraction operator are straightforward, and displayed in table 11. On the basis of the previous definition, it is not hard to define rooted branching bisimulation also on transition systems. We obtain the same equivalence as in 3.3.

$$\frac{x \xrightarrow{p.a} x',\ a \in I}{\tau_I(x) \xrightarrow{p.\tau} \tau_I(x')} \qquad \frac{x \xrightarrow{p.a} q,\ a \in I}{\tau_I(x) \xrightarrow{p.\tau} q}$$

$$\frac{x \xrightarrow{p.a} x',\ a \notin I}{\tau_I(x) \xrightarrow{p.a} \tau_I(x')} \qquad \frac{x \xrightarrow{p.a} q,\ a \notin I}{\tau_I(x) \xrightarrow{p.a} q}$$

$$\frac{r \in i(x)}{r \in i(\tau_I(x))}$$

TABLE 11. Structured operational semantics.

7.4 Axiomatisation.

An aximatisation of silent step and abstraction is given in table 12.

$$\tau_I(\varnothing)=\varnothing$$
$$\tau_I(p \rightarrow \varnothing) = p \rightarrow \varnothing$$
$$\tau_I(p \rightarrow a \rightarrow q) = p \rightarrow a \rightarrow q \qquad \text{if } a \notin I$$
$$\tau_I(p \rightarrow a \rightarrow q) = p \rightarrow \tau \rightarrow q \qquad \text{if } a \in I$$
$$\tau_I(X + Y) = \tau_I(X) + \tau_I(Y)$$
$$\tau_I(X \cdot Y) = \tau_I(X) \cdot \tau_I(Y)$$

$$X \cdot (p \rightarrow \tau \rightarrow q) = (X \gg p) \rightarrow q$$
$$X \cdot ((p \rightarrow \tau \rightarrow p) \cdot (Y + Z) + Y) = X \cdot (Y + Z)$$

TABLE 12. Abstraction and τ- laws.

7.5 FAIR ABSTRACTION.

Table 13 shows a law for removing a τ-loop. This law corresponds to the law $KFAR_1^b$ of [BAW90]. The process on the left-hand side of the equation is shown in fig. 7. We have to give an axiomatisation of the set of exits of a process (similar to the set of entries in table 6) in order to turn this into an algebraic law.

$$((p \rightarrow \tau \rightarrow q) + (q \rightarrow \tau \rightarrow q) + (q \gg X))\uparrow_q^q = (p \rightarrow \tau \rightarrow q) \cdot X$$
$$\text{if } p \neq q \text{ and } q \text{ is not an exit of } X$$

TABLE 13. Fair abstraction rule, simplest case.

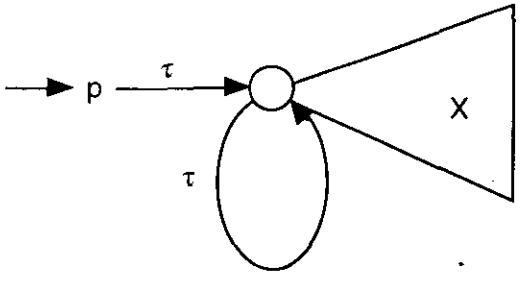

FIGURE 7.

8. EXAMPLES.

In this section, we consider a few simple examples of the use of process algebra over pins. Given a (finite) data set D, we assume the set of pins V satisfies $D \cup \{\surd\} \subseteq V$, where $\surd \notin D$. We use $\surd$ as a default element. We abbreviate $\surd \rightarrow a \rightarrow \surd$ by a, for each $a \in A$, and we abbreviate $\surd \rightarrow \varnothing$ by δ.

8.1 BUFFERS.

We specify a two element buffer in BPA*pin(A, V). An input of $d \in D$ is denoted by action $r_1(d)$, an output of $d \in D$ by $s_2(d)$. The iteration construct *, binary Kleene star, is defined by the usual equation (see [BEBP94]):

$x^*y = x\cdot(x^*y) + y$

TABLE 14. Binary Kleene star.

$$B = \sum_{d\in D} r_1(d)\cdot(d \gg (B' * \delta))$$

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

A two element buffer over a data set of at most two elements can be specified as a closed term over ACP* (without feedback); if there are 3 distinct data elements, this is not possible any more. The above shows that it is possible over BPA*pin. This shows that generalising to processes over pin structures adds expressive power.

8.2 ALTERNATING BIT PROTOCOL.

We consider the well-known example of the Alternating Bit Protocol. We give the specification using the iteration operator, taken from [BEBP93]. There, also a verification can be found.

We assume that elements of D are to be transmitted from sender S to receiver R using unreliable channels K,L. B = {0,1}. We use the standard communication function given by $\gamma(r_k(x), s_k(x)) = c_k(x)$ (see [BAW90]). The communication links are as shown in fig. 8. We have the following specifications.

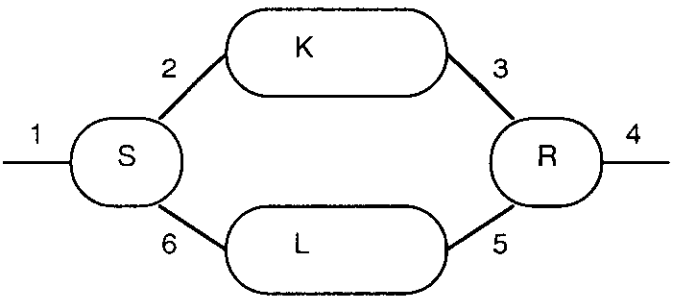

FIGURE 8. ABP.

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

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

$$S = (S0\cdot S1) * \delta$$

$$Sb = \sum_{d\in D} r_1(d)\cdot s_2(db) \cdot \Big(\big((r_6(1\text{-}b)+r_6(\perp))\cdot s_2(db)\big)^*\cdot r_6(b)\Big) \qquad \text{for } b = 0,1.$$

$$R = (R1\cdot R0) * \delta$$

$$Rb = \Big(\big((\sum_{d\in D} r_3(db) + r_3(\perp)) \cdot s5(b) \big) * \big(\sum_{d\in D} r_3(d(1\text{-}b))\cdot s_4(d) \big) \Big) \cdot s_5(1\text{-}b) \qquad \text{for } b=0,1$$

The protocol is now given by $ABP = \partial_H(S \parallel K \parallel L \parallel R)$, with

$H = \{r_k(x), s_k(x) : k \in \{2,3,5,6\}, x \in (D \times B) \cup B \cup \{\bot\}\}$.

Use of pins now allows to give a direct specification of a part of this system. As an example, we consider $\partial_{H2}(S \parallel K)$, with $H2 = \{r_2(x), s_2(x) : x \in D \times B\}$. We claim this expression satisfies the following specification, again using data values as pin names:

$X = (X0 \cdot X1)^* \cdot \delta$

$$Xb = \sum_{d \in D} r_1(d) \cdot c_2(db) \cdot (i \cdot s_3(db) + i \cdot s_3(\bot)) \cdot (d \gg Yb) \qquad \text{for } b = 0,1.$$

$$Yb = \sum_{d \in D} d \rightarrow \Big(\big((r_6(1-b) + r_6(\bot)) \cdot c_2(db) \cdot (i \cdot s_3(db) + i \cdot s_3(\bot)) \big)^* \cdot r_6(b) \Big) \qquad \text{for } b = 0,1.$$

9. SUBALGEBRAS OF REDUCED MODELS.

We consider a specific choice for the pin structure V, and consider some subalgebras. With the abbreviations introduced in the beginning of section 8 (a for $\surd \rightarrow a \rightarrow \surd$, and δ for $\surd \rightarrow \varnothing$), we have embedded the signature of ACP in the signature of ACPpin. However, we get unwanted equation, as e.g. $a \cdot (b \parallel c) = a \cdot \delta$ since pin names do not match. Therefore, we will divide out the equation $\surd \leftarrow\!\!\!\shortmid\!\!\!\rightarrow \surd = \surd$.

9.1 STANDARD PROCESS ALGEBRA.

Suppose the set of pins V contains $\surd$ with $\surd \leftarrow\!\!\!\shortmid\!\!\!\rightarrow \surd = \surd$, and A is a given finite set. Consider the signature Σ that has constants $\{\surd \rightarrow a \rightarrow \surd : a \in A\} \cup \{\surd \rightarrow \varnothing\}$ and operators $\{+, \cdot, \parallel, \between, |\} \cup \{\partial_H : H \subseteq A\}$. Now consider the minimal subalgebra of the Σ-reduct of the initial algebra of ACPpin. We claim that standard ACP is an axiomatisation of this algebra, again substituting a for $\surd \rightarrow a \rightarrow \surd$, and δ for $\surd \rightarrow \varnothing$. Thus, ACP is a subalgebra of a reduced model specification (an SRM specification) of ACPpin. For more information on SRM specifications, we refer to [BAB94].

9.2 ACP WITH $\varnothing$.

Now extend the signature Σ above with the extra constant $\varnothing$. We claim that the axioms in table 15 constitute an SRM specification for this signature ($a,b \in A \cup \{\delta\}$).

10. CONCLUSIONS.

We generalised process algebra to processes with multiple entries and exits, so-called pins. On the resulting model of process graphs modulo bisimulation we can define the feedback operator from flowchart theory, besides all usual operators from process algebra. This gives us more expressive power in the specification of processes parametrised by data. Moreover, we get the orginal ACP back as an SRM specification of the general theory. As a side effect, this introduces a new constant $\varnothing$ into ACP. As future work, we leave the exact determination of the interaction of all the special constants: $\varnothing$,

δ (inaction, [BEK84]), nil (CCS termination [MIL89], [ACH92]), ε (termination option [VRA91]), τ (silent step [MIL89]), Δ (divergence [ACH92]), χ (chaos [BRHR84]), 0 (zero process [BAB90]).

$X + Y = Y + X$	
$(X + Y) + Z = X + (Y + Z)$	
$X + X = X$	
$(X + Y)\cdot Z = X\cdot Z + Y\cdot Z$	
$(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z)$	
$X + \varnothing = X$	$a + \delta = a$
$\varnothing\cdot X = \varnothing$	$a\cdot X + \delta = a\cdot X$
$X\cdot\varnothing = X\cdot\delta$	$\delta\cdot X = \delta$
$a \mid b = \gamma(a, b)$	if $\gamma(a, b)$ defined
$a \mid b = \delta$	otherwise
$X \Vert Y = X \mathbb{L} Y + Y \mathbb{L} X + X \mid Y$	
$X \mathbb{L} \varnothing = \varnothing$	
$\varnothing \mathbb{L} X = \varnothing$	
$a \mathbb{L} (X + \delta) = a\cdot(X + \delta)$	
$a\cdot X \mathbb{L} (Y + \delta) = a\cdot(X \Vert (Y + \delta))$	
$(X + Y) \mathbb{L} Z = X \mathbb{L} Z + Y \mathbb{L} Z$	
$\varnothing \mid X = \varnothing$	
$X \mid \varnothing = \varnothing$	
$a\cdot X \mid b = (a \mid b)\cdot X$	
$a \mid b\cdot X = (a \mid b)\cdot X$	
$a\cdot X \mid b\cdot Y = (a \mid b)\cdot(X \Vert Y)$	
$(X+Y) \mid Z = X \mid Z + Y \mid Z$	
$X \mid (Y+Z) = X \mid Y + X \mid Z$	
$\partial_H(\varnothing) = \varnothing$	
$\partial_H(a) = a$	if $a \notin H$
$\partial_H(a) = \delta$	if $a \in H$
$\partial_H(X + Y) = \partial_H(X) + \partial_H(Y)$	
$\partial_H(X\cdot Y) = \partial_H(X)\cdot\partial_H(Y)$	

TABLE 15. $ACP_\varnothing$.

REFERENCES.

[ACH92] L. ACETO & M. HENNESSY, *Termination, deadlock and divergence*, Journal of the ACM 39 (1), 1992, pp. 147-187.

[BAB90] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with a zero object*, in: Proc. CONCUR 90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), Springer LNCS 458, 1990, pp. 83-98.

[BAB94] J.C.M. BAETEN & J.A. BERGSTRA, *On sequential composition, action prefixes and process*

[BAV93] J.C.M. BAETEN & C. VERHOEF, *A congruence theorem for structured operational semantics with predicates,* in Proc. CONCUR'93, Hildesheim (E. Best, ed.), Springer LNCS 715, 1993, pp. 477-492.

[BAW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press 1990.

[BAWM94] H. BARENDREGT, H. WUPPER & H. MULDER, *Computable processes,* Technical report CSI-R9405, Computing Science Institute, Catholic University of Nijmegen 1994.

[BAR92] M. BARTHA, *An algebraic model of synchronous systems.* Information & Computation, 97, 1992, pp. 97-131.

[BEBP93] J.A. BERGSTRA, I. BETHKE & A. PONSE, *Process algebra with combinators,* report P9319, Programming Research Group, University of Amsterdam 1993.

[BEBP94] J.A. BERGSTRA, I. BETHKE & A. PONSE, *Process algebra with iteration and nesting,* The Computer Journal 37 (4), 1994.

[BEK84] J.A. BERGSTRA & J.W. KLOP, *Process algebra for synchronous communication,* Inf. & Control 60, pp. 109-137.

[BES94] J.A. BERGSTRA & GH. ŞTEFĂNESCU, *Processes with multiple entries and exits modulo isomorphism and modulo bisimulation,* report P9403, Programming Research Group, University of Amsterdam 1994.

[BLE93] S.L. BLOOM & Z. ESIK, *Iteration theories; the equational logic of iterative processes,* Springer Verlag, 1993.

[BRHR84] S.D. BROOKES, C.A.R. HOARE & A.W. ROSCOE, *A theory of communicating sequential processes,* Journal of the ACM 31, 1984, pp. 560-599.

[CAS90] V.E. CĂZĂNESCU & GH. ŞTEFĂNESCU, *Towards a new algebraic foundation of flowchart scheme theory,* Fundamenta Informaticae 13, 1990, pp. 171-210.

[CAS92] V.E. CĂZĂNESCU AND GH. ŞTEFĂNESCU, *A general result of abstarct flowchart schemes with applications to the study of accessibility, reduction and minimization,* Theoretical Computer Science 99, 1992, pp. 1-63.

[EXS92] EXSPECT-TEAM, *ExSpect user manual,* Eindhoven University of Technology 1992.

[KLU93] A.S. KLUSENER, *Models and axioms for a fragment of real time process algebra,* Ph.D. thesis, Eindhoven University of Technology 1993.

[MIL89] R. MILNER, *Communication and concurrency,* Prentice Hall 1989.

[MIL94] R. MILNER, *Action Calculi V: Reflexive molecular forms,* Draft, June 1994.

[STA92] E.W. STARK, *A calculus of dataflow networks,* in: Proc. LICS (Santa Cruz), IEEE, 1992, pp. 125-136.

[STE87a] GH. ŞTEFĂNESCU, *On flowchart theories, part I, the deterministic case,* JCSS 35, 1987, pp. 163-191.

[STE87b] GH. ŞTEFĂNESCU, *On flowchart theories, part II, the non-deterministic case,* TCS 52, 1987, pp. 307-340.

[STE90] GH. ŞTEFĂNESCU, *Feedback theories (a calculus for isomorphism classes of flowchart schemes),* Revue Roumaine de Mathematiques Pures et Appliqué 35, 1990, pp. 73-79.

[VER93] C. VERHOEF, *A general conservative extension theorem in process algebra,* report CSN 93/38, Eindhoven University of Technology 1993. To appear in Proc. PROCOMET'94, IFIP TC2 Working Conference, San Miniato 1994, North-Holland. Conference edition, pp. 144-163.

[VRA91] J.L.M. VRANCKEN, *Studies in process algebra, algebraic specifications and parallelism,* Ph.D. thesis, UvA 1991.

Computing Science Notes

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
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 A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
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.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
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.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.

91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.
92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.

92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
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. Zwietering	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. 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	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
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 Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach

		Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish 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	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.

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	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
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 taxomomy 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	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
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 Marcov Decision Processe 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	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.
94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.

94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.