Process Algebra with Feedback

J.C.M. Baeten[1] J.A. Bergstra[2,3] Gh. Ştefănescu[4]

[1]*Department of Computing Science, Eindhoven University of Technology P.O.Box 513, 5600 MB Eindhoven, The Netherlands*

[2]*Programming Research Group, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*

[3]*Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands*

[4]*Institute of Mathematics of the Romanian Academy P.O.Box 1-764, 70700 Bucharest, Romania*

E-mail: `josb@win.tue.nl` - `janb@fwi.uva.nl` - `ghstef@imar.ro`

Abstract

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 $\rangle\!\!-\!\!\langle : V \times V \to V$. 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*.

2.1 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. Note that we have no cardinality restrictions on graphs, so also infinite graphs are allowed. Our syntax, however, will only generate finite graphs (but infinite executions will be possible).

2.2 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.

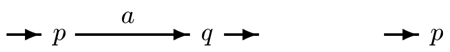

FIGURE 1. Constants

2. Empty graph. We have a constant $\boldsymbol{\emptyset}$ that stands for the empty graph, i.e. $\boldsymbol{\emptyset} = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$. Abusing notation, we usually write $\emptyset$ instead of $\boldsymbol{\emptyset}$. 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 $p \to g$ 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 $p \to \emptyset$ 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 $g \to p$ 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

$(p \to a \to p) + (q \to b \to q) + (p \to c \to q)$ in fig. 2. We picture both p and q twice, in order to emphasise the different roles of entries and exits.

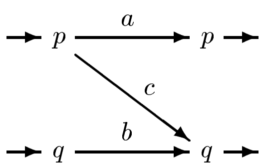

FIGURE 2. Alternative composition.

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 $((p \to a \to p) + (p \to b \to q)) \cdot (p \to c \to q)$ 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 $p \in V$, $g \cdot (p \to \emptyset) = g \cdot \emptyset$, and this graph has no exits.

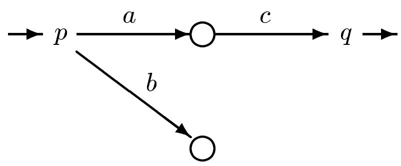

FIGURE 3. Sequential composition.

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 $\to (g)$ 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 $((p \to a \to s) + (s \to b \to s) + (s \to c \to q)){\uparrow}_s^s$ in fig. 4.

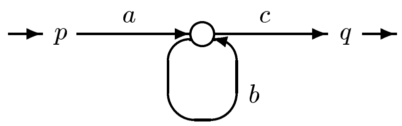

FIGURE 4. Feedback.

10. Pre-feedback. Process graph $g\underline{\uparrow}^q_p$ is just like $g\uparrow^q_p$, 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. $g \bullet^q_p h$ 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*, $g \leftrightarroweq h$, 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 $\mathcal{G}(A, V)/\leftrightarroweq$. 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.
Let P be the set of process expressions given by the constants and operators introduced in section 2. We can give also give semantics for elements of P by means of SOS rules. In table 1 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 $\leftrightarroweq_{\text{sos}}$) on these transition systems yields a congruence.

We can formulate the following theorem, where $[\![.]\!]$ is the interpretation defined in section 2:

3.3 Theorem. For all closed terms u, v: $u \leftrightarroweq_{\text{sos}} v \iff [\![u]\!] \leftrightarroweq [\![v]\!]$.

Proof. In section 2, we gave definitions that allow to associate a graph to each term. Conversely, it is straightforward to associate a term to each finite acyclic graph. In order to prove the theorem, we also need to associate a term with every interior state. However, there is no unique term associated, but for each port name, there is a term with this port name as entry. Thus, if s is a node in a graph g, we write $(g)_s$ for the subgraph from node s, and $term(p, (g)_s)$ for the term, obtained by associating entry p to node s. Then, the theorem follows easily after proving the following statements:

- graph g has edge $p \xrightarrow{a} t$, with $p \in I(g)$, $t \in S(g) \implies$ $term(g) \xrightarrow{p.a} term(r, (g)_t)$, for each $r \in V$
- graph g has edge $p \xrightarrow{a} q$, with $p \in I(g)$, $q \in O(g) \implies$ $term(g) \xrightarrow{p.a} q$
- graph g has edge $s \xrightarrow{a} t$, with $s, t \in S(g) \implies$ $term(p, (g)_s) \xrightarrow{p.a} term(r, (g)_t)$, for each $p, r \in V$
- graph g has edge $s \xrightarrow{a} q$, with $s \in S(g)$, $q \in O(g) \implies$ $term(p, (g)_s) \xrightarrow{p.a} q$, for each $p \in V$
- graph g has $p \in I(g) \implies p \in i(term(g))$
- term u has $u \xrightarrow{p.a} v \implies$ there is $r \in i(v)$ such that $graph(u)$ has edge $p \xrightarrow{a} t$ with $v = term(r, (g)_t)$

$(p \to a \to q) \xrightarrow{p.a} q$	$p \in i(p \to a \to q)$	$\dfrac{p \in i(x)}{p \in i(x+y),\ p \in i(y+x)}$
$\dfrac{x \xrightarrow{p.a} x'}{x+y \xrightarrow{p.a} x',\ y+x \xrightarrow{p.a} x'}$		$\dfrac{x \xrightarrow{p.a} q}{x+y \xrightarrow{p.a} q,\ y+x \xrightarrow{p.a} q}$
$\dfrac{x \xrightarrow{p.a} x'}{x \cdot y \xrightarrow{p.a} x' \cdot y}$	$\dfrac{x \xrightarrow{p.a} q}{x \cdot y \xrightarrow{p.a} (r \to (q \gg y))}$	$\dfrac{p \in i(x)}{p \in i(x \cdot y)}$
$\dfrac{x \xrightarrow{r.a} x'}{(p \to x) \xrightarrow{p.a} x'}$	$\dfrac{x \xrightarrow{r.a} q}{(p \to x) \xrightarrow{p.a} q}$	$p \in i(p \to x)$
$\dfrac{x \xrightarrow{r.a} x'}{(x \to p) \xrightarrow{r.a} (x' \to p)}$	$\dfrac{x \xrightarrow{r.a} q}{(x \to p) \xrightarrow{r.a} p}$	$\dfrac{q \in i(x)}{q \in i(x \to p)}$
$\dfrac{x \xrightarrow{p.a} x'}{(p \gg x) \xrightarrow{p.a} x'}$	$\dfrac{x \xrightarrow{p.a} q}{(p \gg x) \xrightarrow{p.a} q}$	$\dfrac{p \in i(x)}{p \in i(p \gg x)}$
$\dfrac{x \xrightarrow{p.a} x'}{(x \gg q) \xrightarrow{p.a} (x' \gg q)}$	$\dfrac{x \xrightarrow{p.a} q}{(x \gg q) \xrightarrow{p.a} q}$	$\dfrac{p \in i(x)}{p \in i(x \gg q)}$
$\dfrac{x \xrightarrow{p.a} x'}{x \bullet^r_s y \xrightarrow{p.a} x' \bullet^r_s y}$	$\dfrac{x \xrightarrow{p.a} s}{x \bullet^r_s y \xrightarrow{p.a} (q \to (r \gg y))}$	$\dfrac{x \xrightarrow{p.a} q,\ q \neq s}{x \bullet^r_s y \xrightarrow{p.a} q}$
$\dfrac{x \xrightarrow{p.a} x'}{x\underline{\uparrow}^r_s \xrightarrow{p.a} x' \bullet^r_s (x\underline{\uparrow}^r_s)}$	$\dfrac{x \xrightarrow{p.a} s}{x\underline{\uparrow}^r_s \xrightarrow{p.a} (q \to (r \gg (x\underline{\uparrow}^r_s)))}$	$\dfrac{x \xrightarrow{p.a} q,\ q \neq s}{x\underline{\uparrow}^r_s \xrightarrow{p.a} q}$
$\dfrac{x \xrightarrow{p.a} x',\ p \neq r}{x\uparrow^r_s \xrightarrow{p.a} x' \bullet^r_s (x\underline{\uparrow}^r_s)}$	$\dfrac{x \xrightarrow{p.a} s,\ p \neq r}{x\uparrow^r_s \xrightarrow{p.a} (q \to (r \gg (x\underline{\uparrow}^r_s)))}$	$\dfrac{\begin{array}{r} p \neq r \\ x \xrightarrow{p.a} q,\ \ q \neq s \end{array}}{x\uparrow^r_s \xrightarrow{p.a} q}$
$\dfrac{p \in i(x)}{p \in i(x \bullet^r_s y)}$	$\dfrac{p \in i(x)}{p \in i(x\underline{\uparrow}^r_s)}$	$\dfrac{p \in i(x),\ p \neq r}{p \in i(x\uparrow^r_s)}$

TABLE 1. Structured operational semantics.

- term u has $u \xrightarrow{p.a} q \implies graph(u)$ has edge $p \xrightarrow{a} q$ with q an exit
- term u has $p \in i(u) \implies p \in I(graph(u))$.

□

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:

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

Functions:

$\to : V \times P \to P$	entry operator
$\to : P \times V \to P$	exit operator
$\gg\ : V \times P \to P$	initialisation operator
$\gg\ : P \times V \to P$	exiting operator
$\bullet : P \times V \times V \times P \to P$	linking composition
$\underline{\uparrow} : P \times V \times V \to P$	pre-feedback
$\uparrow : P \times V \times V \to P$	feedback

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

4.2 Axioms.

First of all, we present the axioms of BPA_δ, replacing δ by $\emptyset$. 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
$X + \emptyset = X$	A6$\emptyset$
$\emptyset \cdot X = \emptyset$	A7$\emptyset$

TABLE 2. $\mathrm{BPA}_\emptyset$.

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

$$
\begin{array}{l}
p \to (q \to a \to r) = p \to a \to r \\
p \to (q \to \emptyset) = p \to \emptyset \\
(p \to a \to q) \to r = p \to a \to r \\
\emptyset \to p = \emptyset \\
(p \to \emptyset) \to q = p \to \emptyset \\
p \to (X + Y) = p \to X + p \to Y \\
(X + Y) \to p = X \to p + Y \to p \\
p \to X \cdot Y = (p \to X) \cdot Y \\
X \cdot Y \to p = X \cdot (Y \to p) \\
X \cdot (p \to \emptyset) = X \cdot \emptyset
\end{array}
$$

TABLE 3. Entries and exits.

Next, the initialisation and exiting operators are axiomatised in table 4. The axioms of tables 2-4 are the axioms of $\mathrm{BPA_{pin}}(A, V)$. 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 $(p \to a \to q) \cdot (\Sigma_{i \leq n}(q \to b_i \to r_i) \cdot x_i)$ by $(p \to a) \cdot (\Sigma_{i \leq n}(b_i \to r_i) \cdot x_i)$.

$$
\begin{array}{ll}
p \gg \emptyset = \emptyset & \\
p \gg (p \to a \to q) = p \to a \to q & \\
p \gg (q \to a \to r) = \emptyset & \text{if } p \neq q \\
p \gg (p \to \emptyset) = p \to \emptyset & \\
p \gg (q \to \emptyset) = \emptyset & \text{if } p \neq q \\
p \gg (X + Y) = p \gg X + p \gg Y & \\
p \gg X \cdot Y = (p \gg X) \cdot Y & \\
& \\
(p \to a \to q) \cdot X = (p \to a \to q) \cdot (q \gg X) & \\
(p \to a \to q) \cdot X = (p \to a \to r) \cdot (r \to (q \gg X)) & \\
& \\
\emptyset \gg p = \emptyset & \\
(p \to a \to q) \gg q = p \to a \to q & \\
(p \to a \to q) \gg r = (p \to a \to q) \cdot \emptyset & \text{if } q \neq r \\
(p \to \emptyset) \gg q = p \to \emptyset & \\
(X + Y) \gg p = X \gg p + Y \gg p & \\
X \cdot Y \gg p = X \cdot (Y \gg p) &
\end{array}
$$

TABLE 4. Initialisation, exiting.

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

$$
\begin{array}{|l|}
\hline
\emptyset \bullet^r_q Y = \emptyset \\
(p \to a \to q) \bullet^r_q Y = (p \to a \to r) \cdot Y \\
(p \to a \to s) \bullet^r_q Y = p \to a \to s \quad \text{if } s \neq q \\
(p \to \emptyset) \bullet^r_q Y = p \to \emptyset \\
(X + Y) \bullet^q_p Z = X \bullet^q_p Z + Y \bullet^q_p Z \\
(X \cdot Y) \bullet^q_p Z = X \cdot (Y \bullet^q_p Z) \\
\\
X \underline{\upharpoonright}^q_p = X \bullet^q_p (X \underline{\upharpoonright}^q_p) \\
X \upharpoonright^q_p = \Sigma_{r \in i(X) - \{q\}} r \gg (X \underline{\upharpoonright}^q_p) \\
\hline
\end{array}
$$

TABLE 5. Linking and feedback.

By convention, a sum over an empty set is equal to $\emptyset$. 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 $\upharpoonright_{\mathrm{pin}}$ (A, V). When these are clear from the context, we usually omit the parameters A, V, and talk about $\mathrm{BPA}_{\mathrm{pin}}$, BPA $\upharpoonright_{\mathrm{pin}}$.

$$
\begin{array}{|l|}
\hline
i(\emptyset) = \emptyset \\
i(p \to a \to q) = \{p\} \\
i(p \to \emptyset) = \{p\} \\
i(X + Y) = i(X) \cup i(Y) \\
i(X \cdot Y) = i(X) \\
\hline
\end{array}
$$

TABLE 6. Entry set.

5 Theory

5.1 Definition. The set of basic (BPA $\upharpoonright_{\mathrm{pin}}$) terms is defined inductively:
i. $\emptyset$ is a basic term, and for all $p \in V$ $p \to \emptyset$ 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 $\mathrm{BPA_{pin}}$ terms s there is a basic term t such that $\mathrm{BPA_{pin}} \vdash s = t$.

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 $(p \to \emptyset)\cdot X = p \to \emptyset$. Then, we prove the term rewriting system is terminating by using the lexicographic path ordering (many examples of this procedure can be found in [BaV95]). Then, elimination follows since all normal forms are basic terms. Note that we can also eliminate linking composition. □

5.3 Theorem (Soundness Theorem). $\mathcal{G}(A,V)/\leftrightarrows\ \models \mathrm{BPA}\!\uparrow_{\mathrm{pin}}$.

Proof. In each case, it is straightforward to find a bisimulation relation in the SOS model. For example, to prove the validity of $X \cdot Y \to p = X \cdot (Y \to p)$, relate each term to itself, and each term of the form $u \cdot v \to p$ to $u \cdot (v \to p)$. To do the proof in the graph model, we need to talk about graphs more explicitly, as in [Vra91]. This requires much overhead from graph theory, and we refrain from doing this here. □

5.4 Theorem (Conservativity Theorem). $\mathrm{BPA}\!\uparrow_{\mathrm{pin}}$ over a set of actions A is a conservative extension of BPA_δ (with $\emptyset$ instead of δ) over the set of actions $\{p \to a \to q : a \in A, p, q \in V\} \cup \{p \to \emptyset : p \in V\}$.

Sketch of proof. Since all our SOS rules are in path format, operational conservativity follows immediately (see [Ver94]). Then the (equational) conservativity follows by a result of [Ver94] since the axiomatisation of BPA_δ is sound and complete and the axiomatisation of $\mathrm{BPA}\!\uparrow_{\mathrm{pin}}$ is sound (theorem 5.3). □

5.5 Theorem (Completeness Theorem). The axiomatisation of $\mathrm{BPA_{pin}}$ (so without iterative construct!) is complete for the model $\mathcal{G}(A,V)/\leftrightarrows$.

Sketch of proof. By the general result of [Ver94]. In addition to the ingredients of the previous proof, all we need is the elimination theorem. □

5.6 Definition. On $\mathrm{BPA}\!\uparrow_{\mathrm{pin}}$, we can define projection operators π_n as follows ($n \geq 0$):

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

TABLE 7. Projection.

5.7 Theorem (Projection theorem). Let t be a closed BPA$\uparrow_{\text{pin}}$ term, and let $n \geq 0$. Then $\pi_n(t)$ can be written as a basic term.

Proof. As in the standard theory, see [BaW90]. □

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$\uparrow_{\text{pin}}$ 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 BPA$_{\text{pin}}$ term t (so without iterative constructs!) such that $[\![t]\!] \leftrightarrows g$. To write this as a BPA$\uparrow_{\text{pin}}$ term (even up to isomorphism) is easy: take fresh names r, s and consider $((p \to a \to r) + (q \to a \to s) + (r \to b \to s) + (r \to c \to p) + (s \to c \to q))\uparrow_r^r\uparrow_s^s$.

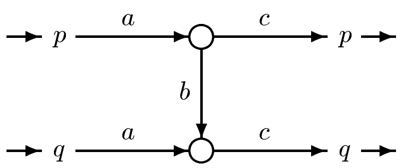

FIGURE 5.

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. Left global state pairing. 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. Right global state pairing. Let a process graph g be given, and let $p \in V$. $g \mid\!\!-\!\!\langle\, p$ 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. Parallel composition. 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 $g \parallel h$ 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 $(p \to a \to p) \parallel (q \to b \to q)$, where $\gamma(a, b) = c$, in fig. 6.

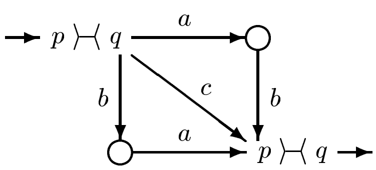

FIGURE 6.

4. Left merge. The graph of $g \mathbin{\|\!\_} h$ 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 $g \mathbin{\_\!\|} h$ 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.

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

$$\frac{x \xrightarrow{r.a} x'}{x \vdash\!\prec p \xrightarrow{r \succ\!\prec p.a} x' \vdash\!\prec p} \qquad \frac{x \xrightarrow{r.a} q}{x \vdash\!\prec p \xrightarrow{r \succ\!\prec p.a} q \succ\!\prec p} \qquad \frac{q \in i(x)}{q \succ\!\prec p \in i(x \vdash\!\prec p)}$$

$$\frac{p \in i(x),\ q \in i(y)}{p \succ\!\prec q \in i(x \Box y)} \qquad (\Box = \|,\ \lfloor\!\lfloor\,,\ \rfloor\!\rfloor\,,\ |)$$

$$\frac{x \xrightarrow{p.a} x',\ r \in i(y)}{\begin{array}{ll} x \| y \xrightarrow{p \succ\!\prec r.a} x' \| (r \gg y), & y \| x \xrightarrow{r \succ\!\prec p.a} (r \gg y) \| x', \\ x \lfloor\!\lfloor y \xrightarrow{p \succ\!\prec r.a} x' \| (r \gg y), & y \rfloor\!\rfloor x \xrightarrow{r \succ\!\prec p.a} (r \gg y) \| x' \end{array}}$$

$$\frac{x \xrightarrow{p.a} q,\ r \in i(y)}{\begin{array}{ll} x \| y \xrightarrow{p \succ\!\prec r.a} q \succ\!\dashv (r \gg y), & y \| x \xrightarrow{r \succ\!\prec p.a} (r \gg y) \vdash\!\prec q, \\ x \lfloor\!\lfloor y \xrightarrow{p \succ\!\prec r.a} q \vdash\!\prec (r \gg y), & y \rfloor\!\rfloor x \xrightarrow{r \succ\!\prec p.a} (r \gg y) \vdash\!\prec q \end{array}}$$

$$\frac{x \xrightarrow{p.a} x',\ y \xrightarrow{r.b} y',\ \gamma(a,b) = c}{x \| y \xrightarrow{p \succ\!\prec r.c} x' \| y',\ x \mid y \xrightarrow{p \succ\!\prec r.c} x' \| y'}$$

$$\frac{x \xrightarrow{p.a} x',\ y \xrightarrow{r.b} s,\ \gamma(a,b) = c}{\begin{array}{ll} x \| y \xrightarrow{p \succ\!\prec r.c} x' \vdash\!\prec s, & x \mid y \xrightarrow{p \succ\!\prec r.c} x' \vdash\!\prec s, \\ y \| x \xrightarrow{r \succ\!\prec p.c} s \succ\!\dashv x', & y \mid x \xrightarrow{p \succ\!\prec r.c} s \succ\!\dashv x' \end{array}}$$

$$\frac{x \xrightarrow{p.a} q,\ y \xrightarrow{r.b} s,\ \gamma(a,b) = c}{x \| y \xrightarrow{p \succ\!\prec r.c} q \succ\!\prec s,\ x \mid y \xrightarrow{p \succ\!\prec r.c} q \succ\!\prec s}$$

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

TABLE 8. Structured operational semantics.

6.3 Algebra.
We repeat the added signature:

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

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

$$
\begin{array}{|l|}
\hline
p \rangle\!\!\dashv \emptyset = \emptyset \\
p \rangle\!\!\dashv (q \to \emptyset) = p \rangle\!\!\langle q \to \emptyset \\
p \rangle\!\!\dashv (q \to a \to r) = p \rangle\!\!\langle q \to a \to p \rangle\!\!\langle r \\
p \rangle\!\!\dashv (X + Y) = p \rangle\!\!\dashv X + p \rangle\!\!\dashv Y \\
p \rangle\!\!\dashv X \cdot Y = (p \rangle\!\!\dashv X) \cdot (p \rangle\!\!\dashv Y) \\
\\
\emptyset \vdash\!\!\langle p = \emptyset \\
(q \to \emptyset) \vdash\!\!\langle p = q \rangle\!\!\langle p \to \emptyset \\
(q \to a \to r) \vdash\!\!\langle p = q \rangle\!\!\langle p \to a \to r \rangle\!\!\langle p \\
(X + Y) \vdash\!\!\langle p = X \vdash\!\!\langle p + Y \vdash\!\!\langle p \\
X \cdot Y \vdash\!\!\langle p = (X \vdash\!\!\langle p) \cdot (Y \vdash\!\!\langle p) \\
\hline
\end{array}
$$

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 $\mathrm{BPA_{pin}}$ plus the axioms in tables 9 and 10 constitute the theory $\mathrm{ACP_{pin}}$. Likewise, $\mathrm{BPA}{\upharpoonleft}_{\mathrm{pin}}$ plus axioms in tables 9 and 10 yields $\mathrm{ACP}{\upharpoonleft}_{\mathrm{pin}}$. On the basis of these axiomatisations, it is easy to obtain axiomatisations for theories without communication, $\mathrm{PA_{pin}}$ and $\mathrm{PA}{\upharpoonleft}_{\mathrm{pin}}$.

$X \parallel Y = X \mathbin{\|\!\!\lfloor} Y + X \mathbin{\rfloor\!\!\|} Y + X \mid Y$	
$p \to a \mid b \to q = p \to \gamma(a,b) \to q$	if $\gamma(a,b)$ defined
$p \to a \mid b \to q = p \to \emptyset$	if $\gamma(a,b)$ not defined
$\emptyset \mathbin{\|\!\!\lfloor} X = \emptyset$	
$(p \to \emptyset) \mathbin{\|\!\!\lfloor} X = \sum_{r \in i(X)} p \rangle\!\!-\!\!\langle r \to \emptyset$	
$(p \to a \to q) \mathbin{\|\!\!\lfloor} X = (\sum_{r \in i(X)} (p \to a \to q) \vdash\!\!\langle r) \cdot (q \rangle\!\!\dashv X)$	
$(p \to a \to q) \cdot X \mathbin{\|\!\!\lfloor} Y = (\sum_{r \in i(Y)} (p \to a \to q) \vdash\!\!\langle r) \cdot (X \parallel Y)$	
$(X + Y) \mathbin{\|\!\!\lfloor} Z = X \mathbin{\|\!\!\lfloor} Z + Y \mathbin{\|\!\!\lfloor} Z$	
$X \mathbin{\rfloor\!\!\|} \emptyset = \emptyset$	
$X \mathbin{\rfloor\!\!\|} (p \to \emptyset) = \sum_{r \in i(X)} r \rangle\!\!-\!\!\langle p \to \emptyset$	
$X \mathbin{\rfloor\!\!\|} (p \to a \to q) = (\sum_{r \in i(X)} r \rangle\!\!\dashv (p \to a \to q) \cdot (X \vdash\!\!\langle q)$	
$X \mathbin{\rfloor\!\!\|} (p \to a \to q) \cdot Y = (\sum_{r \in i(X)} r \rangle\!\!\dashv (p \to a \to q) \cdot (X \parallel Y)$	
$X \mathbin{\rfloor\!\!\|} (Y + Z) = X \mathbin{\rfloor\!\!\|} Y + X \mathbin{\rfloor\!\!\|} Z$	
$\emptyset \mid X = \emptyset$	
$X \mid \emptyset = \emptyset$	
$(p \to \emptyset) \mid X = \sum_{r \in i(X)} p \rangle\!\!-\!\!\langle r \to \emptyset$	
$X \mid (p \to \emptyset) = \sum_{r \in i(X)} r \rangle\!\!-\!\!\langle p \to \emptyset$	
$(p \to a \to q) \mid (r \to b \to s) = p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle s$	
$(p \to a \to q) \cdot X \mid (r \to b \to s) = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle s) \cdot (X \vdash\!\!\langle s)$	
$(p \to a \to q) \mid (r \to b \to s) \cdot X = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle s) \cdot (q \rangle\!\!\dashv X)$	
$(p \to a \to q) \cdot X \mid (r \to b \to s) \cdot Y = (p \rangle\!\!-\!\!\langle r \to a \mid b \to q \rangle\!\!-\!\!\langle s) \cdot (X \parallel Y)$	
$(X + Y) \mid Z = X \mid Z + Y \mid Z$	
$X \mid (Y + Z) = X \mid Y + X \mid Z$	
$\partial_H(\emptyset) = \emptyset$	
$\partial_H(p \to \emptyset) = p \to \emptyset$	
$\partial_H(p \to a \to q) = p \to a \to q$	if $a \notin H$
$\partial_H(p \to a \to q) = p \to \emptyset$	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 $s \Longrightarrow t$, 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 $t \Longrightarrow t' \xrightarrow{a} t''$ in h and $R(s, t'), R(s', t'')$ *or* there are $t' \in S(h)$, $q \in O(h)$ such that $t \Longrightarrow t' \xrightarrow{a} q$ 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 $t \Longrightarrow t' \xrightarrow{a} t''$ in h and $R(s, t')$, t'' is a q-semi-endpoint *or* there is $t' \in S(h)$ such that $t \Longrightarrow t' \xrightarrow{a} q$ 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 $s \Longrightarrow s' \xrightarrow{a} s''$ in g and $R(s', t), R(s'', t')$ *or* there are $s' \in S(g)$, $q \in O(g)$ such that $s \Longrightarrow s' \xrightarrow{a} q$ 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 $s \Longrightarrow s' \xrightarrow{a} s''$ in g and $R(s', t)$, s'' is a q-semi-endpoint *or* there is $s' \in S(g)$ such that $s \Longrightarrow s' \xrightarrow{a} q$ in g and $R(s', t)$.

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

We say g, h are *rooted branching bisimilar*, $g \leftrightarrows_{\mathrm{rb}} h$, 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 $\mathcal{G}(A \cup \{\tau\}, V)/\leftrightarrows_{\mathrm{rb}}$. 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.

$\tau_I(\emptyset) = \emptyset$	
$\tau_I(p \to \emptyset) = p \to \emptyset$	
$\tau_I(p \to a \to q) = p \to a \to q$	if $a \notin I$
$\tau_I(p \to a \to q) = p \to \tau \to q$	if $a \in I$
$\tau_I(X + Y) = \tau_I(X) + \tau_I(Y)$	
$\tau_I(X \cdot Y) = \tau_I(X) \cdot \tau_I(Y)$	
$X \cdot (p \to \tau \to q) = (X \gg p) \to q$	
$X \cdot ((p \to \tau \to p) \cdot (Y + Z) + Y) = X \cdot (Y + Z)$	

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.

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

TABLE 13. Fair abstraction rule, simplest case.

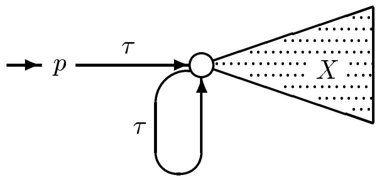

FIGURE 7.

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 $\surd \to a \to \surd$ by a, for each $a \in A$, and we abbreviate $\surd \to \emptyset$ by δ.

8.1 Buffers.

We specify a two element buffer in $\mathrm{BPA}^*_{\mathrm{pin}}(A, V)$. 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 [BeBP94b]):

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

TABLE 14. Binary Kleene star.

$$B = \sum_{d \in D} r_1(d) \cdot (d \gg (B'^*\delta))$$
$$B' = \sum_{d \in D} d \to ((s_2(d) \cdot \sum_{e \in D} r_1(e) \to e) + (\sum_{e \in D} r_1(e) \cdot s_2(d) \to e)).$$

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 $\mathrm{BPA}^*_{\mathrm{pin}}$. 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 [BeBP94a]. 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.

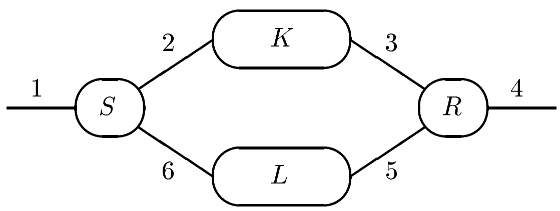

FIGURE 8. ABP.

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

$$L = (\textstyle\sum_{b \in B} r_5(b) \cdot (i \cdot s_6(b) + i \cdot s_6(\bot)))^* \delta$$

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

$$Sb = \textstyle\sum_{d \in D} r_1(d) \cdot s_2(db) \cdot (((r_6(1-b) + r_6(\bot)) \cdot s_2(db))^* r_6(b)) \quad \text{for } b = 0, 1$$

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

$$Rb = (((\textstyle\sum_{d \in D} r_3(db) + r_3(\bot)) \cdot s_5(b))^* (\textstyle\sum_{d \in D} r_3(d(1-b)) \cdot s_4(d))) \cdot s_5(1-b) \quad \text{for } b = 0, 1.$$

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:

$X = (X0 \cdot X1)^* \delta$
$Xb = \sum_{d \in D} r_1(d) \cdot c2(db) \cdot (i \cdot s_3(db) + i \cdot s_3(\perp)) \cdot (d \gg Yb)$ for $b = 0, 1$
$Yb =$
$\sum_{d \in D} d \to (((r_6(1-b) + r_6(\perp)) \cdot c_2(db) \cdot (i \cdot s_3(db) + i \cdot s_3(\perp)))^* r_6(b))$
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 $\surd \to a \to \surd$, and δ for $\surd \to \emptyset$), we have embedded the signature of ACP in the signature of $\mathrm{ACP_{pin}}$. However, we get unwanted equations, as e.g. $a \cdot (b \parallel c) = a \cdot \delta$ since pin names do not match. Therefore, we will divide out the equation $\surd \rightarrowtail\!\!\!\!\leftarrowtail \surd = \surd$.

9.1 Standard Process Algebra.

Suppose the set of pins V contains $\surd$ with $\surd \rightarrowtail\!\!\!\!\leftarrowtail \surd = \surd$, and A is a given finite set. Consider the signature Σ that has constants $\{\surd \to a \to \surd : a \in A\} \cup \{\surd \to \emptyset\}$ and operators $\{+, \cdot, \|, \lfloor\!\lfloor, |\} \cup \{\partial_H : H \subseteq A\}$. Now consider the minimal subalgebra of the Σ-reduct of the initial algebra of $\mathrm{ACP_{pin}}$. We claim that standard ACP is an axiomatisation of this algebra, again substituting a for $\surd \to a \to \surd$, and δ for $\surd \to \emptyset$. Thus, ACP is a subalgebra of a reduced model specification (an SRM specification) of $\mathrm{ACP_{pin}}$. For more information on SRM specifications, we refer to [BaB94].

9.2 ACP with $\emptyset$.

Now extend the signature Σ above with the extra constant $\emptyset$. 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 $\emptyset$ into ACP. As future work, we leave the exact determination of the interaction of all the special constants: $\emptyset$, 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]).

$$
\begin{array}{lll}
X + Y = Y + X & & \\
(X + Y) + Z = X + (Y + Z) & & \\
X + X = X & & \\
(X + Y) \cdot Z = X \cdot Z + Y \cdot Z & & \\
(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) & & \\
X + \emptyset = X & & a + \delta = a \\
\emptyset \cdot X = \emptyset & & \delta \cdot X = \delta \\
 & X \cdot \emptyset = X \cdot \delta & \\
 & & \\
a \mid b = \gamma(a,b) & \text{if } \gamma(a,b) \text{ defined} & \\
a \mid b = \delta & \text{otherwise} & \\
 & & \\
X \parallel Y = X \mathbin{⌊\!⌊} Y + Y \mathbin{⌊\!⌊} X + X \mid Y & & \\
X \mathbin{⌊\!⌊} \emptyset = \emptyset & & \\
\emptyset \mathbin{⌊\!⌊} X = \emptyset & & \\
a \mathbin{⌊\!⌊} (X + \delta) = a \cdot (X + \delta) & & \\
a \cdot X \mathbin{⌊\!⌊} (Y + \delta) = a \cdot (X \parallel (Y + \delta)) & & \\
(X + Y) \mathbin{⌊\!⌊} Z = X \mathbin{⌊\!⌊} Z + Y \mathbin{⌊\!⌊} Z & & \\
\emptyset \mid X = \emptyset & & \\
X \mid \emptyset = \emptyset & & \\
a \cdot X \mid b = (a \mid b) \cdot X & & \\
a \mid b \cdot X = (a \mid b) \cdot X & & \\
a \cdot X \mid b \cdot Y = (a \mid b) \cdot (X \parallel Y) & & \\
(X + Y) \mid Z = X \mid Z + Y \mid Z & & \\
X \mid (Y + Z) = X \mid Y + X \mid Z & & \\
 & & \\
\partial_H(\emptyset) = \emptyset & & \\
\partial_H(a) = a & \text{if } a \notin H & \\
\partial_H(a) = \delta & \text{if } a \in H & \\
\partial_H(X + Y) = \partial_H(X) + \partial_H(Y) & & \\
\partial_H(X \cdot Y) = \partial_H(X) \cdot \partial_H(Y) & &
\end{array}
$$

Table 15. $\mathrm{ACP}_{\emptyset}$.

References

[AcH92] L. Aceto and M. Hennessy, *Termination, deadlock and divergence*, Journal of the ACM 39(1), 1992, pp. 147-187.

[BaB90] J.C.M. Baeten and J.A. Bergstra, *Process algebra with a zero object*, in: Proc. CONCUR'90, Amsterdam (J.C.M. Baeten and J.W. Klop, eds.), LNCS 458, Springer-Verlag, 1990, pp. 83-98.

[BaB94] J.C.M. Baeten and J.A. Bergstra, *On sequential composition, action prefixes and process prefix*, Formal Aspects of Computing 6(3), 1994, pp. 250-268.

[BaV93] J.C.M. Baeten and C. Verhoef, *A congruence theorem for structured operational semantics with predicates*, in Proc. CONCUR'93, Hildesheim (E. Best, ed.), LNCS 715, Springer-Verlag, 1993, pp. 477-492.

[BaV95] J.C.M. Baeten and C. Verhoef, *Concrete process algebra*, to appear in: Handbook of logic in computer science (S. Abramsky, D.M. Gabbay and T.S.E. Maibaum, eds.), Vol. IV, Syntactical Methods, Chapter 2, Oxford University Press, 1995, pp. 149 - 268.

[BaW90] J.C.M. Baeten and W.P. Weijland, *Process algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press 1990.

[BaWM94] H. Barendregt, H. Wupper and 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 and Computation 97, 1992, pp. 97-131.

[BeBP94a] J.A. Bergstra, I. Bethke and A. Ponse, *Process algebra with combinators*, in E. Börger, Y. Gurevich, and K. Meinke, editors, *Proceedings CSL'93, Swansea*, LNCS 832, Springer-Verlag, 1994, pp. 36 - 65.

[BeBP94b] J.A. Bergstra, I. Bethke and A. Ponse, *Process algebra with iteration and nesting*, The Computer Journal 37(4), 1994, pp. 243 - 258.

[BeK84] J.A. Bergstra and J.W. Klop, *Process algebra for synchronous communication*, Inf. and Control 60, 1984, pp. 109-137.

[BeS94] J.A. Bergstra and 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 and Z. Esik, *Iteration theories; the equational logic of iterative processes*, Springer-Verlag, 1993.

[BrHR84] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe, *A theory of communicating sequential processes*, Journal of the ACM 31, 1984, pp. 560-599.

[CaS90] V.E. Căzănescu and 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 abstract 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.

[Ver94] C. Verhoef, *A general conservative extension theorem in process algebra*, in: Programming Concepts, Methods and Calculi (PROCOMET'94), San Miniato (E.-R. Olderog, ed.), IFIP Transactions A-56, North-Holland 1994, pp. 149-168.

[Vra91] J.L.M. Vrancken, *Studies in process algebra, algebraic specifications and parallelism*, Ph.D. thesis, UvA 1991.