Process Algebra with Iteration

Jan A. Bergstra[1,2] Inge Bethke[1] Alban Ponse[1]

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

[2] *Utrecht University, Department of Philosophy*
Heidelberglaan 2, 3584 CS Utrecht, The Netherlands

E-mail: `janb@fwi.uva.nl - inge@fwi.uva.nl - alban@fwi.uva.nl`

Abstract

We introduce iteration in process algebra by means of (the binary version of) Kleene's star operator: x^*y is the process that chooses between x and y, and upon termination of x has this choice again. It is proved that adding respectively interleaving, communication and abstraction operators increases expressivity up to the regular processes. However, if the distinction between (successful) termination and deadlock is dropped, ACP (the Algebra of Communicating Processes, [BK84b]) with * is expressive up to the regular processes. Finally, some attention is paid to other iteration operators and fairness issues, and some open questions are formulated.

Key words & Phrases: process algebra, iteration, Kleene star.
1987 CR Categories: F.1.1, F.1.2, F.3.2, F.4.3, I.1.0.

Note: An earlier version of this work was presented at the REX Symposium, Noordwijkerhout, June 1–4, 1993.

1 Introduction

In 1956, Kleene introduced in [Kle56] the binary operator * for describing 'regular events'. The material for that paper was drawn from a "Research Memorandum" written in 1951 by the same author (not commonly accessible), which was a reaction on the "fundamental paper on the logical analysis of nervous activity" by McCulloch and Pitts [MP43]. Kleene regarded the "McCulloch-Pitts nerve nets" as (finite) automata modelling regular events, but judged the analysis on arbitrary nerve nets (containing circles) in [MP43] obscure, so proceeded independently there. He defined *regular expressions*, and gave algebraic transformation rules for these, notably

$$E^*F = F \vee E(E^*F)$$

(E^*F being the *iterate* of E on F). He also noted the correspondence with the conventions of algebra, treating $E \vee F$ as analogous to $E + F$, and EF as the product of E and F.

In [CEW58], Copi, Elgot and Wright showed interest in the results in [Kle56]. However, they judged in particular Kleene's theorems on analysis[1] and synthesis[2] obscured both by the complexity of his basic concepts and by the nature of the elements used in his nets. They introduced simpler and stronger nets (in a sense weakening Kleene's synthesis result, but stating that this "brings the essential nature of the result into sharper focus"), and simpler (equally named) operations. In particular they introduced a unary * operator

[1] Theorem 5, stating that finite automata model regular events.
[2] Theorem 3, stating that each regular event can be described by a nerve net.

> "[...] because Kleene's star operator is "essentially" singulary and because the singulary operation simplifies the algebra of regular events. It should be noted that the singulary and binary star operations are interdefinable."

Following KLEENE, we define x^*y to denote the process P that satisfies

$$P = x \cdot P + y.$$

The operator * can be added to e.g. the following process algebra axiom systems: BPA, PA, ACP, ACP_τ (see [BK84b, BW90]). In each case many models are known that allow to solve the equation $P = x \cdot P + y$. Often these models satisfy the scheme RSP (Recursive Specification Principle, a fixed point principle) which allows one to infer that such P is unique as well. We call the operator * SEI (Single Exit Iteration), as to give self-explanatory naming for other, related iteration operators and to avoid confusion about the arity of "Kleene's star operator".

Taking $y = \epsilon$ (the *empty process*, or **skip**), one obtains $x^*\epsilon$ which satisfies

$$x^*\epsilon = x \cdot (x^*\epsilon) + \epsilon. \tag{1}$$

According to [Pon92] (inspired by [Fer92]), the *unary* operator $\_^*\epsilon$ is a plausible candidate for the unary version of Kleene's star operator in process algebra. Moreover, taking $x = \delta$ (*inaction* or *deadlock*) in (1) implies that $\delta^*\epsilon = \epsilon$ (by the law $\delta \cdot x = \delta$), and hence that $\_^*\epsilon$ cannot be used in a setting without having ϵ available as a separate process (once δ is accepted as one). So with δ and ϵ, the interdefinability of the unary and the binary star, noted in [CEW58], is preserved.

Obvious as ϵ may be (being a unit for $\cdot$), its introduction is nontrivial because at the same time it must be a unit for $\|$ (merge) as well. In the design of BPA, PA, ACP and related axiom systems it has proven useful to both study versions of the theory with and without ϵ. Just for this reason we propose the star operator with its (original) defining equation

$$x^*y = x \cdot (x^*y) + y \tag{2}$$

given by KLEENE in [Kle56].

In [Mil84], MILNER paid some attention to 'star behaviours', i.e., bisimulation congruence classes that are representable by *regular expressions* (in the modern sense, with unary *). He showed that a simple regular behaviour is not a star behaviour (in Section 3.2 we further comment on this) and raised the questions of a complete axiomatisation, and what structural properties of (finite) transition systems characterise star behaviours. In [Tro93], TROEGER introduces a process specification language with (single exit) iteration, writing $y \;\; wh \;\; x$ (y *while* x) for x^*y. This work contains an interesting axiom for SEI:

$$\text{(SEI3)} \qquad x^*(y \cdot ((x+y)^*z) + z) = (x+y)^*z$$

(see further Section 3.1 for axioms on SEI). However, TROEGER concentrates on semantical characterisation results for the restriction of the language obtained by excluding (explicit) internal choice[3]. Therefore, the main results in [Tro93] refer to a different setting.

Plan of this paper. In Section 2 we introduce some well-known process algebra axiom systems. Next SOS-transition-systems (Structural Operational Semantics) and strong and weak bisimulation equivalence are introduced; these serve as a technical means to prove some of the classification results reported on. In Section 2.4 we define *regular* processes, a class of recursively definable processes containing all processes definable with * and the other process algebra operators.

[3] I.e., excluding expressions such as $a \cdot P + a \cdot Q$ or $(a \cdot P \parallel b \cdot Q) + a \cdot R$.

In Section 3 we formally introduce the iteration operator SEI, and formulate some laws for this operator. Then we classify the expressivity of the process algebra axiom systems BPA, PA (plus their δ versions), ACP and ACP$_\tau$ extended with SEI. We show that these extensions form a genuine hierarchy up to the class of regular processes, among which the fact that *termination-free* regular processes can be expressed in ACP with SEI and binary communication. Here the notion 'termination-free' comprises only one type of "termination", contrary to the usual view in ACP where (successful) termination is distinguished from deadlock. This section ends with the introduction of *flat iteration*: x^*y is a 'flat iterative' process if x represents a sum of atomic actions. For such processes, most of our hierarchy collapses.

In Section 4 we digress a little. First we consider some other iteration operators. Furthermore, we provide some equations for SEI in the presence of MILNER's silent action τ ([Mil80, Mil89]) and the abstraction operator of ACP$_\tau$ ([BK85]), capturing various notions of *fairness*. Finally, some attention is paid to the open questions raised throughout the paper.

2 Axioms, behavioural equivalences and regularity

2.1 Axiom systems: BPA – ACP

In this section the process algebra axiom systems BPA – ACP are shortly introduced. For more detailed background and results we refer to [BK84a, BW90].

We start from a non-empty set A of actions $a, b, a_1, ...$. The axioms in Table 1 constitute the axiom system BPA(A). It axiomatises the binary infix operators $+$ (alternative composition) and $\cdot$ (sequential composition, we mostly suppress the symbol $\cdot$ and take sequential composition to be most binding). A BPA(A) *process* is represented by an expression over this signature. The axioms of BPA(A) describe the basic identities between processes. The operator $+$ is commutative, associative and idempotent (A1 – A3). The operator $\cdot$ right distributes over $+$ and is associative (A4, A5). Note that left distributivity of $\cdot$ over $+$ is absent.

(A1)	$x + y$	$=$	$y + x$
(A2)	$x + (y + z)$	$=$	$(x + y) + z$
(A3)	$x + x$	$=$	x
(A4)	$(x + y)z$	$=$	$xz + yz$
(A5)	$(xy)z$	$=$	$x(yz)$

Table 1: The axioms of BPA(A).

The predefined process δ (*inaction* or *deadlock*) cannot perform any activity and prevents subsequent processes from being executed. The system BPA$_\delta$(A) is obtained by extending BPA(A) with the axioms A6 and A7 given in Table 2. Let in the following A_δ be short for $A \cup \{\delta\}$.

(A6)	$x + \delta$	$=$	x
(A7)	δx	$=$	δ

Table 2: The axioms A6, A7 of BPA$_\delta$(A).

The systems BPA(A) and BPA$_\delta$(A) can be extended to PA(A) and PA(A)$_\delta$ respectively by adding the axioms M1 – M4 from Table 3. These axioms define the two operators $\|$ (free merge) and the auxiliary operator $\mathbin{\|\!\!\lfloor}$ (left merge) which both have equal binding power stronger than + and less than $\cdot$.

(M1)	$x \parallel y$	$=$	$x \mathbin{\|\!\!\lfloor} y + y \mathbin{\|\!\!\lfloor} x$
(M2)	$a \mathbin{\|\!\!\lfloor} x$	$=$	ax
(M3)	$ax \mathbin{\|\!\!\lfloor} y$	$=$	$a(x \parallel y)$
(M4)	$(x+y) \mathbin{\|\!\!\lfloor} z$	$=$	$x \mathbin{\|\!\!\lfloor} z + y \mathbin{\|\!\!\lfloor} z$

Table 3: The axioms M1 – M4 of PA(A) ($a \in A$) and PA(A)$_\delta$ ($a \in A_\delta$).

Let $\gamma : A \times A \to A$ be a partial function that is *commutative*:

$$\gamma(a,b) = \gamma(b,a) \text{ if } \gamma(a,b) \textit{ defined}, \text{ further denoted as } \gamma(a,b)\downarrow,$$

and *associative*:

$$\gamma(a,\gamma(b,c)) = \gamma(\gamma(a,b),c) \text{ if } \gamma(a,\gamma(b,c))\downarrow, \text{ notation } \gamma(a,b,c).$$

The system ACP(A,γ) can be obtained from BPA$_\delta$(A) by adding the axioms given in Table 4. Here $\|$ (merge) is regarded as an operator that can give rise to *communication actions* (if so defined by γ). Next to the auxiliary operator $\mathbin{\|\!\!\lfloor}$ there is also the auxiliary operator $|$, the communication merge. Finally, for each $H \subseteq A$, there is a unary operator ∂_H. The axioms in Table 4 define all new operators.

In the special case that the communication function γ defines at most *binary* communications, i.e.,

$$\forall a,b,c \in A(\gamma(a,b,c)\uparrow)$$

we speak of *handshaking* [BT84].

(CM1)	$x \parallel y$	$=$	$x \mathbin{\|\!\!\lfloor} y + y \mathbin{\|\!\!\lfloor} x + x \mid y$	(CF1)	$a \mid b$	$=$	$\gamma(a,b)$	if $\gamma(a,b)\downarrow$
(CM2)	$a \mathbin{\|\!\!\lfloor} x$	$=$	ax	(CF2)	$a \mid b$	$=$	δ	otherwise
(CM3)	$ax \mathbin{\|\!\!\lfloor} y$	$=$	$a(x \parallel y)$					
(CM4)	$(x+y) \mathbin{\|\!\!\lfloor} z$	$=$	$x \mathbin{\|\!\!\lfloor} z + y \mathbin{\|\!\!\lfloor} z$	(D1)	$\partial_H(a)$	$=$	a	if $a \notin H$
(CM5)	$ax \mid b$	$=$	$(a \mid b)x$	(D2)	$\partial_H(a)$	$=$	δ	if $a \in H$
(CM6)	$a \mid bx$	$=$	$(a \mid b)x$	(D3)	$\partial_H(x+y)$	$=$	$\partial_H(x) + \partial_H(y)$	
(CM7)	$ax \mid by$	$=$	$(a \mid b)(x \parallel y)$	(D4)	$\partial_H(xy)$	$=$	$\partial_H(x) \cdot \partial_H(y)$	
(CM8)	$(x+y) \mid z$	$=$	$x \mid z + y \mid z$					
(CM9)	$x \mid (y+z)$	$=$	$x \mid y + x \mid z$					

Table 4: The axioms CF1,2, CM1-9 and D1-4 of ACP(A,γ) ($a,b \in A_\delta$).

2.2 SOS and bisimulation semantics

In this section we introduce bisimulation semantics for processes. We define various "SOS notions" (Structural Operational Semantics), and relate processes to "SOS-transition-systems", providing an operational semantics in the style of PLOTKIN [Plo81]. Then we define bisimilarity as an equivalence between SOS-transition-systems. We use SOS notions and bisimilarity in the proofs of some of the classification results in Section 3.3.

An *SOS-transition-system* $\mathcal{A}$ is a tuple $\langle S_{\mathcal{A}}, L_{\mathcal{A}}, \longrightarrow_{\mathcal{A}}, s_{\mathcal{A}}\rangle$, where

- $S_{\mathcal{A}}$ is a set of *states*,
- $L_{\mathcal{A}}$ is a set of *labels*,
- $\longrightarrow_{\mathcal{A}}$ is a *transition relation*,
- $s_{\mathcal{A}} \in S_{\mathcal{A}}$ is the *initial state*.

Let $E(A)$ be one of the process algebra axiom systems introduced before, and let $\mathcal{P}(E(A))$ represent all processes over $E(A)$. As the set of states $S_{\mathcal{A}}$, further also called *SOS-states*, we take $\mathcal{P}(E(A))$. As labels we take the atomic actions in A. The transition relation $\longrightarrow_{E(A)}$ contains transitions

$$\cdot \xrightarrow{\cdot} \cdot \subseteq \mathcal{P}(E(A)) \times A \times \mathcal{P}(E(A)),$$

and for modelling *(successful) termination*, special transitions

$$\cdot \xrightarrow{\cdot} \surd \subseteq \mathcal{P}(E(A)) \times A$$

(pronounce $\surd$ as "tick"). The idea is that for $a \in A$, a transition $P \xrightarrow{a} P'$ expresses that by executing a, the process P can evolve into P'. In this case P' represents what remains to be executed. The transition $P \xrightarrow{a} \surd$ expresses that the process P can terminate (successfully) after executing a. The rules in Table 5 define the transition relation $\longrightarrow_{E(A)}$. The signature and parameters of $E(A)$ (possible including a communication function γ) determine which rules are appropriate. For example, the last two rules for $\|$ are not relevant for PA(A). Note that the SOS-state δ has no outgoing transitions. Finally, the SOS-transition-system related to a process P has initial SOS-state P. If $E(A)$ is fixed, we often write simply P for the SOS-transition-system

$$\langle \mathcal{P}(E(A)), A, \longrightarrow_{E(A)}, P\rangle.$$

In the sequel we use the following notions based on SOS:

1. An action $a \in A$ is an *exit* of P if $P \xrightarrow{a} \surd$.
2. P is an *SOS-successor of* Q if $Q \xrightarrow{a} P$ for some $a \in A$.
3. Let the relation $\twoheadrightarrow^{+}$ be defined as follows
   (a) $P \xrightarrow{a} Q \implies P \twoheadrightarrow^{+} Q$,
   (b) $P \xrightarrow{a} Q \;\&\; Q \twoheadrightarrow^{+} R \implies P \twoheadrightarrow^{+} R$.

   Then P is a *proper SOS-substate* of Q if $P \twoheadrightarrow^{+} Q$.
4. P is an *SOS-state of* Q if either $P \equiv Q$, or P is a proper SOS-substate of Q.

$a \in A$	$a \xrightarrow{a} \surd$			
$+$	$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$	$\dfrac{x \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$	$\dfrac{y \xrightarrow{a} \surd}{x + y \xrightarrow{a} \surd}$
$\cdot$	$\dfrac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$	$\dfrac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$		
$\parallel$	$\dfrac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y}$	$\dfrac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$	$\dfrac{x \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} y}$	$\dfrac{y \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} x'}$
	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \parallel y \xrightarrow{\gamma(a,b)} x' \parallel y'}$ if $\gamma(a,b)\downarrow$		$\dfrac{x \xrightarrow{a} \surd \xleftarrow{b} y}{x \parallel y \xrightarrow{\gamma(a,b)} \surd}$ if $\gamma(a,b)\downarrow$	
$\mathbin{\|\!\!\lfloor}$	$\dfrac{x \xrightarrow{a} x'}{x \mathbin{\|\!\!\lfloor} y \xrightarrow{a} x' \parallel y}$	$\dfrac{x \xrightarrow{a} \surd}{x \mathbin{\|\!\!\lfloor} y \xrightarrow{a} y}$		
$\mid$ provided $\gamma(a,b)\downarrow$	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \mid y \xrightarrow{\gamma(a,b)} x' \parallel y'}$	$\dfrac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} y'}{x \mid y \xrightarrow{\gamma(a,b)} y'}$	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} \surd}{x \mid y \xrightarrow{\gamma(a,b)} x'}$	$\dfrac{x \xrightarrow{a} \surd \xleftarrow{b} y}{x \mid y \xrightarrow{\gamma(a,b)} \surd}$
∂_H	$\dfrac{x \xrightarrow{a} x'}{\partial_H(x) \xrightarrow{a} \partial_H(x')}$ if $a \notin H$		$\dfrac{x \xrightarrow{a} \surd}{\partial_H(x) \xrightarrow{a} \surd}$ if $a \notin H$	

Table 5: Transition rules for BPA(A) – ACP(A, γ), where $a, b \in A$, $H \subseteq A$.

A *bisimulation* is a binary, symmetric relation $\mathcal{R}$ on SOS-states over $E(A)$ that satisfies

$$\begin{array}{rcl} P\mathcal{R}Q \;\;\&\;\; P \xrightarrow{a} P' & \Longrightarrow & \exists Q' \,.\, Q \xrightarrow{a} Q' \;\;\&\;\; P'\mathcal{R}Q' \\ P\mathcal{R}Q & \Longrightarrow & (P \xrightarrow{a} \surd \iff Q \xrightarrow{a} \surd) \end{array}$$

Two SOS-states are called *bisimilar*, notation $P \leftrightarrow Q$, if there exists a bisimulation $\mathcal{R}$ with $P\mathcal{R}Q$.

Theorem 2.2.1.

1. (Soundness) *Bisimilarity is a congruence relation with respect to all operators defined (for a "modern" proof see* [BV93]*), and all axioms presented are valid in bisimulation semantics.*

2. (Completeness) *Up to* ACP(A, γ) *all axiom systems are complete with respect to bisimilarity between the processes that can be expressed* [BW90].

2.3 ACP with abstraction

Considering unobservable actions τ (*silent step*) leads to further extensions of already introduced systems. The system $\mathrm{ACP}(A,\gamma)$, for example, can be extended to $\mathrm{ACP}_\tau(A,\gamma)$ (introduced in [BK85]) by adding the axioms T1-3, TM1,2, TC1-4, D0 and TI0-4 from Table 6.

(T1)	$x\tau$	$=$	x	(D0)	$\partial_H(\tau)$	$=$	τ	
(T2)	τx	$=$	$\tau x + x$					
(T3)	$a(\tau x + y)$	$=$	$a(\tau x + y) + ax$	(TI0)	$\tau_I(\tau)$	$=$	τ	
				(TI1)	$\tau_I(a)$	$=$	a	if $a \notin I$
(TM1)	$\tau \mathbin{\|\!\!\lfloor} x$	$=$	τx	(TI2)	$\tau_I(a)$	$=$	τ	if $a \in I$
(TM2)	$\tau x \mathbin{\|\!\!\lfloor} y$	$=$	$\tau(x \parallel y)$	(TI3)	$\tau_I(x+y)$	$=$	$\tau_I(x) + \tau_I(y)$	
(TC1)	$\tau \mid x$	$=$	δ	(TI4)	$\tau_I(xy)$	$=$	$\tau_I(x) \cdot \tau_I(y)$	
(TC2)	$x \mid \tau$	$=$	δ					
(TC3)	$\tau x \mid y$	$=$	$x \mid y$					
(TC4)	$x \mid \tau y$	$=$	$x \mid y$					

Table 6: Axioms T1-3, TM1,2, TC1-4, D0 and TI0-4 of $\mathrm{ACP}_\tau(A,\gamma)$ ($a \in A_\delta,\ H, I \subseteq A$).

These axioms describe the interaction between τ and the operators available in $\mathrm{ACP}(A,\gamma)$, and the behaviour of the *abstraction operator* τ_I that hides actions from a certain set $I \subseteq A$.
The transition rules that have to be added to Table 5 are the ones of Table 7 below.

$a \in A \cup \{\tau\}$	$a \xrightarrow{a} \tau$			
saturation	$\dfrac{x \xrightarrow{a} x' \quad x' \xrightarrow{\tau} x''}{x \xrightarrow{a} x''}$		$\dfrac{x \xrightarrow{a} x' \quad x' \xrightarrow{\tau} \surd}{x \xrightarrow{a} \surd}$	
	$\dfrac{x \xrightarrow{\tau} x' \quad x' \xrightarrow{a} x''}{x \xrightarrow{a} x''}$		$\dfrac{x \xrightarrow{\tau} x' \quad x' \xrightarrow{a} \surd}{x \xrightarrow{a} \surd}$	
τ_I	$\dfrac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}$	if $a \in I$	$\dfrac{x \xrightarrow{a} \surd}{\tau_I(x) \xrightarrow{\tau} \surd}$	if $a \in I$
	$\dfrac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{a} \tau_I(x')}$	if $a \notin I$	$\dfrac{x \xrightarrow{a} \surd}{\tau_I(x) \xrightarrow{a} \surd}$	if $a \notin I$

Table 7: Transition rules for τ and τ_I, where $a \in A \cup \{\tau\},\ I \subseteq A$.

Let $P \overset{\tau}{\twoheadrightarrow} P'$ abbreviate zero or more τ-steps from P to P'. A *weak bisimulation* is a binary relation $\mathcal{R}$ on SOS-states over $\mathrm{ACP}_\tau(A,\gamma)$ that satisfies

$$\begin{array}{lcl}
P\mathcal{R}Q \ \ \& \ \ P \xrightarrow{a} P' & \Longrightarrow & (\, a=\tau \ \ \& \ \ P'\mathcal{R}Q\,) \text{ or} \\
 & & \exists Q',Q'',Q''' \,.\, Q \overset{\tau}{\twoheadrightarrow} Q'' \xrightarrow{a} Q''' \overset{\tau}{\twoheadrightarrow} Q' \ \ \& \ \ P'\mathcal{R}Q' \\
P\mathcal{R}Q \ \ \& \ \ Q \xrightarrow{a} Q' & \Longrightarrow & (\, a=\tau \ \ \& \ \ P\mathcal{R}Q'\,) \text{ or} \\
 & & \exists P',P'',P''' \,.\, P \overset{\tau}{\twoheadrightarrow} P'' \xrightarrow{a} P''' \overset{\tau}{\twoheadrightarrow} P' \ \ \& \ \ P'\mathcal{R}Q' \\
P\mathcal{R}Q \ \ \& \ \ P \xrightarrow{a} \surd & \Longrightarrow & (\, a=\tau \ \ \& \ \ Q \overset{\tau}{\twoheadrightarrow} \surd\,) \text{ or} \\
 & & \exists Q',Q'' \,.\, Q \overset{\tau}{\twoheadrightarrow} Q' \xrightarrow{a} Q'' \overset{\tau}{\twoheadrightarrow} \surd \\
P\mathcal{R}Q \ \ \& \ \ Q \xrightarrow{a} \surd & \Longrightarrow & (\, a=\tau \ \ \& \ \ P \overset{\tau}{\twoheadrightarrow} \surd\,) \text{ or} \\
 & & \exists P',P'' \,.\, P \overset{\tau}{\twoheadrightarrow} P' \xrightarrow{a} P'' \overset{\tau}{\twoheadrightarrow} \surd .
\end{array}$$

Two $\mathrm{ACP}_\tau(A,\gamma)$ processes are called *weakly bisimilar*, notation

$$P \leftrightarroweq_{wb} Q$$

if there exists a weak bisimulation $\mathcal{R}$ that satisfies the following condition:

(i) $P\mathcal{R}Q$

(ii) $P \xrightarrow{\tau} P' \quad \Longrightarrow \quad \exists Q',Q'' \,.\, Q \xrightarrow{\tau} Q'' \overset{\tau}{\twoheadrightarrow} Q' \ \ \& \ \ P'\mathcal{R}Q'$

(iii) as in (ii) with P and Q interchanged.

2.4 Regular processes

In this section we introduce *regular* processes over A_δ as the class of (recursively defined) processes that contains the iterative ones over A_δ.

A process P_1 is *regular* over A_δ if P_1 satisfies a finite set of equations of the form

$$P_i = \sum_{j=1}^{n} (\alpha_{i,j} \cdot P_j) + \beta_i$$

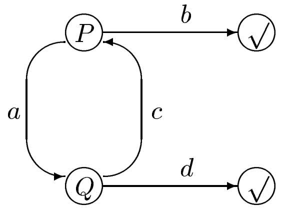
for $i = 1, \ldots, n$, where $\alpha_{i,j}$ and β_i are finite sums of actions or δ. Let $\mathrm{REG}(A_\delta)$ denote the class of regular processes over A_δ.

The regular process P_1 in the representation above is *termination-free* if $\beta_i = \delta$ for all i. The distinction between (successful) termination and inaction/deadlock is reflected in that between 'regular processes' and 'termination-free regular processes' (in our sense).

A simple example of a regular process is $a+b$, for it satisfies $a+b = \delta(a+b)+a+b$ in $\mathrm{BPA}_\delta(A)$. As a second example consider the regular process P defined by

$$\begin{array}{lcl}
P & = & a \cdot Q + b \\
Q & = & c \cdot P + d
\end{array}$$

or in "SOS style":



We use RSP, the Recursive Specification Principle ([BK86, BW90]), for reasoning about equality when regular processes are involved (we do not introduce SOS notions for regular processes, though these are easy to define, see e.g. [BW90]). In the following a short explanation of RSP in the setting with regular processes:

Let the regular process P_1 be defined by equations on $P_1, ..., P_n$ in the format defined above. Now let $E(A)$ be one of the axiom systems introduced, and let processes $Q_1, ..., Q_n$ over $E(A)$ be such that in $E(A)$ these solve the equations for $P_1, ..., P_n$, respectively. Then RSP states that $Q_i = P_i$ in $E(A)$.

3 Iteration

In Section 1 we introduced Kleene's star operator *, here called SEI (Single Exit Iteration), and its defining axiom

$$\text{(SEI)} \quad x^* y = x(x^* y) + y$$

in process algebra. Given a process algebra axiom system $E(A)$ (possibly with a communication function γ), let $E(A)^*$ be the extension of $E(A)$ with SEI.

3.1 Axioms, Double Exit Iteration and SOS rules

In Table 8 we give some axioms for SEI in BPA$(A)^*$. The particular order of the = arguments refers to possible nice term rewrite properties of BPA$(A)^*$ expressions. The third axiom SEI3 stems from [Tro93], where it is used for a slightly different process specification formalism.

(SEI1)	$x \cdot (x^* y) + y$	$=$	$x^* y$
(SEI2)	$x^*(y \cdot z)$	$=$	$(x^* y) \cdot z$
(SEI3)	$x^*(y \cdot ((x+y)^* z) + z)$	$=$	$(x+y)^* z$

Table 8: Some axioms for SEI.

Some typical consequences are

$$\begin{array}{rrcll} i. & x^*((x+y)^* z) & = & (x+y)^* z & \text{(substitute } x+y \text{ for } y \text{ in SEI3, and apply A2, A3),} \\ ii. & x^*(x^* x) & = & x^*(x(x^* x) + x) & \text{(by SEI1)} \\ & & = & x^*(x((x+x)^* x) + x) & \text{(by A3)} \\ & & = & x^* x & \text{(by SEI3, A3).} \end{array}$$

In a setting with δ and the unary operators ∂_H and τ_I we give additional axioms in Table 9.

(SEI4)	$(x + y \cdot \delta)^* z$	$=$	$x^*(y \cdot \delta + z)$
(SEI5)	$\partial_H(x^* y)$	$=$	$\partial_H(x)^* \partial_H(y)$
(SEI6)	$\tau_I(x^* y)$	$=$	$\tau_I(x)^* \tau_I(y)$

Table 9: Three more axioms for SEI.

Double Exit Iteration. Let the 4-ary operator ** (DEI, Double Exit Iteration) be defined by

$$\text{(DEI)} \quad (x, y^{**}u, v) = x \cdot (y \cdot (x, y^{**}u, v) + v) + u. \tag{3}$$

The operator ** is used to prove some expressivity results on * in Section 3.3. Observe that $(x, x^{**}u, u) = x^*u$ by double application of (SEI) on x^*u. However, ** is much more expressive than *, for example the regular process P defined by

$$P = a \cdot Q + b$$
$$Q = c \cdot P + d$$

can be represented by $(a, c^{**}b, d)$, whereas this is not so easy with * (see Section 3.3 on expressivity of * systems). Furthermore, taking one of its arguments δ, we can express ** by * in $\text{BPA}_\delta(A)^*$:

1. $(\delta, y^{**}u, v) = u$,
2. $(x, \delta^{**}u, v) = xv + u$.
3. $(x, y^{**}\delta, v) = x((yx)^*v)$,
4. $(x, y^{**}u, \delta) = (xy)^*u$,

In the following it is assumed that the τ_I and ∂_H operators also distribute over ** (cf. the axioms SEI5 and SEI6).

Remark 3.1.1. In the paper [HY76], Hashiguchi and Yoo introduce a star operator *of degree two*, also denoted by **. Their operator really extends the 'regularity' of regular expressions, witness the expressibility of the language $\{a^nb^n \mid n > 0\}$ by $(a, ab, b)^{**}$. However, we judge the nature of their operation ** and main result, i.e., the class of 'regular ** languages' coincides with the class of 'completely linearly nested context-free languages', sufficiently different to avoid confusion, and stick to our use of the notation **. **End Remark.**

SOS rules. Given our description of SEI, the transition rules for * given in Table 10 are quite obvious. It also follows immediately how transition rules for ** can be consistently defined. However, for the latter we have no employ.

$*$	$\dfrac{x \xrightarrow{a} x'}{x^*y \xrightarrow{a} x' \cdot (x^*y)}$	$\dfrac{x \xrightarrow{a} \surd}{x^*y \xrightarrow{a} x^*y}$	$\dfrac{y \xrightarrow{a} y'}{x^*y \xrightarrow{a} y'}$	$\dfrac{y \xrightarrow{a} \surd}{x^*y \xrightarrow{a} \surd}$

Table 10: Transition rules for SEI, where $a \in A$.

It can be easily shown that the SEI axioms are valid in bisimulation semantics, and according to [BV93], bisimilarity is a congruence with respect to *. Note that with the first two transition rules, an SOS-state P can have itself as a *proper* SOS-substate, e.g., $a^*b \twoheadrightarrow^+ a^*b$ by $a^*b \xrightarrow{a} a^*b$.

We now introduce some further SOS terminology on a special kind of regular processes, the 'SOS-cyclic' ones.

A *chain* in a binary relation $\mathcal{R}$ over SOS-states is a sequence $(P_0, ..., P_n)$ such that $(P_i, P_{i+1}) \in \mathcal{R}$ for $i < n$. The chain $(P_0, ..., P_n)$ is an *cycle* if the P_i are all syntactically distinct and also $(P_n, P_0) \in R$.

In this case we sometimes write $(P_0, ..., P_n > P_0)$. A cycle $(P_0, ..., P_n)$ is *trivial* if $n = 0$; otherwise it is *nontrivial*.

An SOS-state P is *SOS-root-cyclic* if P has a cycle starting at P with the 'SOS-successor' relation as $\mathcal{R}$. An SOS-state P is *SOS-cyclic* if P has an SOS-state Q that is SOS-root-cyclic.

For example, $a(a^*b) + b$ is SOS-cyclic: a^*b is a proper SOS-substate that is SOS-root-cyclic.

Lemma 3.1.2. *Let P be a process expression over* $\mathrm{ACP}_\tau(A,\gamma)^*$ *or one of the smaller signatures. Then P has finitely many SOS-states.*

Furthermore, if Q is (weakly) bisimilar with P, then either none or both P and Q are SOS-cyclic.

Proof. By induction on the maximal depth of the * nestings in P. The point is that the only transition rule that adds 'complexity' is the leftmost one for *, and the number of SOS-successors is in this case finitely bounded by the nature of the rules for the $\cdot$, the sequential composition operator.

The second part of the lemma follows immediately. □

We end this section with two open questions (see further Section 4.3):

1. *Is* $\mathrm{BPA}(A)$ *extended with the axioms in Table 8 a complete axiomatisation of bisimilarity between* $\mathrm{BPA}(A)^*$ *processes? And if not, what axioms or rules should be added?*
2. *What axioms or rules should be added to Table 8 and* SEI4 *to obtain a complete axiomatisation of bisimilarity between* $\mathrm{BPA}_\delta(A)^*$ *processes?*

3.2 Hierarchy results on Single Exit Iteration

In this section we prove

$$\mathrm{BPA}(A)^* \prec \begin{array}{c} \mathrm{BPA}_\delta(A)^* \\ \\ \mathrm{PA}(A)^* \end{array} \prec \mathrm{PA}_\delta(A)^* \prec \mathrm{ACP}(A,\gamma)^* \prec \mathrm{ACP}_\tau(A,\gamma)^*$$

where $\prec$ means "less expressive than". We prove all $\prec$ results with help of SOS notions and bisimilarity. The rightmost result $\mathrm{ACP}(A,\gamma)^* \prec \mathrm{ACP}_\tau(A,\gamma)^*$ contains a modest claim on the parameters of the theories involved: $|A| \geq 8$ and γ restricted to handshaking. For the other hierarchy results we do not make any claims on a minimal cardinality of A, as to give easy proofs.

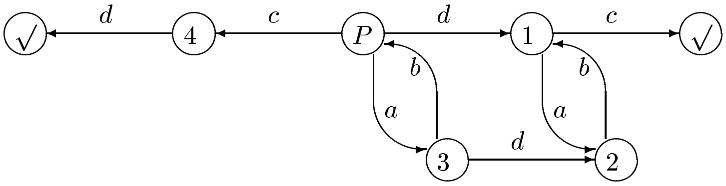
In [Mil84], Milner already has a result implying that $\mathrm{BPA}_{\delta\epsilon}(A)^* \prec \mathrm{REG}(A_\delta)$. He proved that F defined by

$$\begin{array}{rcl} F & = & a_1G + a_2H \\ G & = & b_1H + b_2F \\ H & = & c_1F + c_2G \end{array}$$

cannot be a 'star behaviour'. Here ϵ refers to the 'empty process', and * to the unary Kleene star.

Theorem 3.2.1. $\mathrm{BPA}(A)^* \prec \mathrm{PA}(A)^*$, *and (even without use of δ)* $\mathrm{BPA}_\delta(A)^* \prec \mathrm{PA}_\delta(A)^*$.

Proof. Consider the $\mathrm{PA}(A)^*$ process $P = (ab)^*c \parallel d$, which can graphically be characterised as follows (duplicating the $\surd$ states):



with

1	abbreviates	$(ab)^*c$
2	abbreviates	$b((ab)^*c)$
3	abbreviates	$b((ab)^*c) \parallel d$
4	abbreviates	d

Suppose P can be specified in $\mathrm{BPA}_\delta(A)^*$. As P is SOS-root-cyclic over $\mathrm{PA}(A)^*$, we can assume that P can be represented by a (syntactically) minimal SOS-root-cyclic process over $\mathrm{BPA}_\delta(A)^*$, say Q. By the soundness of $\mathrm{PA}_\delta(A)^*$ (see Theorem 2.2.1), Q is bisimilar with P. We show by induction on the structure of $\mathrm{BPA}_\delta(A)^*$ processes that such Q cannot exist, contradicting the supposition.

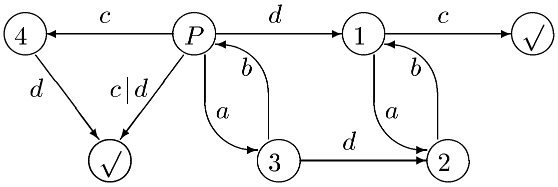
$Q \in A_\delta$ or $Q \equiv Q_1 + Q_2$. Then Q is not SOS-root-cyclic.

$Q \equiv Q_1 \cdot Q_2$. Then necessarily Q_1 has an initial d step and an initial c step. As Q can terminate in states bisimilar with 1 and 4, it follows that $Q_2 \leftrightarrow 1 \leftrightarrow 4$. But obviously $1 \not\leftrightarrow 4$.

$Q \equiv {Q_1}^* Q_2$. Then necessarily Q_1 has no initial c or d step: after one of these termination is possible, so repetition of one of these would otherwise also be possible, which is contradictory. By minimality, Q_1 must have an initial a step. Again, the subsequent d step must be necessarily from Q_2, so Q_2 can terminate with dbc. But this 'termination path' is not possible from (SOS-state) Q. □

Theorem 3.2.2. $\mathrm{PA}_\delta(A)^* \prec \mathrm{ACP}(A,\gamma)^*$, *even without use of* δ *and of* ∂_H.

Proof. Consider the $\mathrm{ACP}(A,\gamma)^*$ process $P = (ab)^*c \parallel d$ and let $c \mid d$ be the only communication action defined. Then P can be depicted as follows:



with

1	abbreviates	$(ab)^*c$
2	abbreviates	$b((ab)^*c)$
3	abbreviates	$b((ab)^*c) \parallel d$
4	abbreviates	d

Suppose P can be specified in $\mathrm{PA}_\delta(A)^*$. As P is SOS-root-cyclic, we can assume that P can be represented by a minimal SOS-root-cyclic $\mathrm{PA}_\delta(A)^*$ process, say Q. By the soundness of $\mathrm{ACP}(A,\gamma)^*$, Q is bisimilar with P. We show by induction on the structure of $\mathrm{PA}_\delta(A)^*$ processes that such Q cannot exist.

$Q \in A_\delta$ or $Q \equiv Q_1 + Q_2$ or $Q \equiv Q_1 \mathbin{|\!\!\lfloor} Q_2$. Then Q is not SOS-root-cyclic.

$Q \equiv Q_1 \cdot Q_2$. Then Q cannot have the exit $c \mid d$.

$Q \equiv {Q_1}^* Q_2$. In this case Q_1 has no initial c, d or $c \mid d$ step: after one of these termination is possible, so repetition of one of these would otherwise also be possible, which is contradictory. By minimality, Q_1 has an initial a step and the subsequent d step is an initial step of Q_2. So Q_2 can terminate with dbc. But this termination behaviour is not possible from Q.

$Q \equiv Q_1 \parallel Q_2$. Then Q cannot have the exit $c \mid d$. □

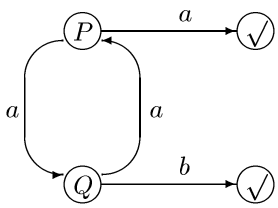
The following lemma relates the expressivity of $\mathrm{ACP}(B,\gamma)^*$ to $\mathrm{REG}(A_\delta)$, where $B \supseteq A$, by focusing on termination properties. We use this result in the proof of our last hierarchy result: $\mathrm{ACP}(A,\gamma)^* \prec \mathrm{ACP}_\tau(A,\gamma)^*$.

Lemma 3.2.3. *Let $B \supseteq A$ and A contain (at least) two actions. Then* $\mathrm{ACP}(B,\gamma)^* \prec \mathrm{REG}(A_\delta)$. *This is independent of the definition of γ.*

Proof. Consider the regular process P defined by

$$\begin{array}{rcl} P & = & a \cdot Q + a \\ Q & = & a \cdot P + b \end{array}$$

or graphically:



Any two $\mathrm{ACP}(B,\gamma)^*$ processes that satisfy the equations for P and Q are both necessarily SOS-cyclic as infinite behaviour is possible (cf. Lemma 3.1.2). Moreover, any of these must contain an SOS-cycle C on which both the exits a and b occur at least once at different SOS-states of C. We show that such 'exit behaviour' cannot be defined in $\mathrm{ACP}(B,\gamma)^*$, thereby proving the claim of the lemma.

Let the ***exit property*** of SOS-*root*-cyclic states over $\mathrm{ACP}(B,\gamma)^*$ be the following:

> "if two different SOS-states A and B of an SOS-root-cyclic state R that are in some cycle C, have exits, say $a_0, ..., a_k$ from A and $b_0, ..., b_n$ from B, then $\{a_0, ..., a_k\} = \{b_0, ..., b_n\}$".

We prove the exit property for $\mathrm{ACP}(B,\gamma)^*$ by induction on the syntactic structure of SOS-root-cyclic states, ranged over by R.

$R \in A_\delta$ or $R \equiv R_1 \Box R_2$ with $\Box \in \{+, \lfloor\!\lfloor, |\}$. Then R is not SOS-root-cyclic.

$R \equiv R_1 \cdot R_2$. There are two types of cycles that can start from R, namely
(i) those that are generated by R_1, i.e., are of the form $(R_1 \cdot R_2, R_1^1 \cdot R_2, ..., R_1^k \cdot R_2)$, and
(ii) those that are defined by R_2 and for which $R_1 \cdot R_2$ is a proper SOS-substate.
We can apply the induction hypothesis to both types. (Note that cycles of type (i) have no exits at all.)

$R \equiv R_1 \parallel R_2$. Let $C = (S_0, ..., S_n > S_0)$ with $S_0 \equiv R$ be an SOS-cycle starting from R. Assume that $n > 0$ (otherwise we are done). Observe that each S_{i+1} is of the form $R_1' \parallel R_2'$ with R_j' an SOS-state of R_j for $j = 1, 2$. Suppose S_i, S_j are two different SOS-states of C. Then the exits at both of these come from communications between their composing SOS-states of R_1 and R_2. Now either one of both these composing states is equivalent with R_1 or R_2, respectively, or both are proper SOS-substates of R_1 and R_2, respectively. In each case the exit property follows by induction.

$R \equiv \partial_H(R_1)$. If R is SOS-root-cyclic, so is R_1, and the 'exit-property' is preserved by ∂_H application: R_1 exits can at most disappear.

$R \equiv {R_1}^* R_2$. For trivial cycles the property follows immediately. If R is the first SOS-state of a nontrivial cycle C, it must be that the proper SOS-substates of C are of the form $R_1'({R_1}^* R_2)$, so have no exits (otherwise R is not SOS-root-cyclic). □

Our last hierarchy result states that $\mathrm{ACP}(A,\gamma)^* \prec \mathrm{ACP}_\tau(A,\gamma)^*$. Of course the comparison between $\mathrm{ACP}(A,\gamma)^*$ and $\mathrm{ACP}_\tau(A,\gamma)^*$ only makes sense for processes that can be represented without τ. We prove this result by a very simple process defined with handshaking, presupposing that A contains at least 8 (different) actions.

Theorem 3.2.4. *If A contains (at least) 8 actions, one can find γ with handshaking such that* $\mathrm{ACP}(A,\gamma)^* \prec \mathrm{ACP}_\tau(A,\gamma)^*$.

Proof. Let $\{a,b,c\} \subseteq A$. In the previous lemma we proved that the regular process P defined by

$$\begin{array}{rcl} P &=& a \cdot Q + a \\ Q &=& a \cdot P + b \end{array}$$

cannot be defined in $\mathrm{ACP}(A,\gamma)^*$ (not even in $\mathrm{ACP}(B,\gamma)^*$ for any extension B of A). Hence we can prove the theorem by showing that P *can* be defined in $\mathrm{ACP}_\tau(A,\gamma)^*$.

Assume $H \stackrel{\text{def}}{=} \{k_1, k_2, ..., k_5\} \subseteq A \setminus \{a,b,c\}$ and let γ be defined by

$$\begin{array}{l} \gamma(k_1,k_1) \stackrel{\text{def}}{=} a \\ \gamma(k_2,k_4) \stackrel{\text{def}}{=} b \\ \gamma(k_3,k_3) \stackrel{\text{def}}{=} a \\ \gamma(k_1,k_2) \stackrel{\text{def}}{=} a \\ \gamma(k_3,k_5) \stackrel{\text{def}}{=} c \end{array}$$

and undefined otherwise. Consider the $\mathrm{ACP}_\tau(A,\gamma)^*$ process $\overline{P}$ defined by

$$\begin{array}{rcl} \overline{P} &=& \tau_{\{c\}} \circ \partial_H(R \parallel S) \\ R &=& (k_1 \cdot k_2)^* k_3 \\ S &=& {k_1}^*(k_3 + k_4 \cdot k_5). \end{array}$$

Then one can derive $\overline{P} = P$ in $\mathrm{ACP}_\tau(A,\gamma)^*$ with RSP:

$$\overline{P} \;=\; a \cdot \tau_{\{c\}} \circ \partial_H(k_2 \cdot R \parallel S) + a$$

$$\begin{array}{rcl} \tau_{\{c\}} \circ \partial_H(k_2 \cdot R \parallel S) &=& a \cdot \overline{P} + b \cdot \tau_{\{c\}} \circ \partial_H(\overline{P} \parallel k_5) \\ &=& a \cdot \overline{P} + b \cdot \tau_{\{c\}}(c) \\ &=& a \cdot \overline{P} + b. \end{array}$$

□

We leave it open whether the result above can be proved with less than 8 actions (for suitable γ with handshaking), and raise the following question:

3. *Under what minimal cardinality of A are the hierarchy results preserved?*

3.3 Expressivity results on regular processes

All results in this section are on the general expressivity of ACP and ACP_τ with *. Of course, in all process algebra systems with * introduced here, one cannot define a process that is *not* regular. We now show that each regular process over A_δ can be defined with *. The proofs of these facts are based on extensions of A with (auxiliary) fresh actions, and appropriate communication functions.

Theorem 3.3.1. *For each regular process P over A_δ we can find an extension B of A such that P can be expressed in* $\mathrm{ACP}_\tau(B,\gamma)^*$ *with ternary communication only, and the actions in A not subject to communication.*

Proof. Let the regular process P_1 be given by

$$P_i = \sum_{j=1}^{n}(\alpha_{i,j} \cdot P_j) + \beta_i$$

where $\alpha_{i,j}$ and β_i are finite sums of actions or δ. We may assume that $a_{j,j} = \delta$ for all $j \in \{1, ..., n\}$, for if not, we can duplicate the identifier P_j, e.g.:

$$\begin{array}{lcl} X & = & aX + bY + c \\ Y & = & dX + e \end{array}$$

then becomes

$$\begin{array}{lcl} X & = & aX' + bY + c \\ X' & = & aX + bY + c \\ Y & = & dX + e. \end{array}$$

In the case that $n = 1$, apparently $P_1 = \beta_1$, a sum of atomic actions (or δ), and therefore even expressible as a $\mathrm{BPA}_\delta(A)$ process.

Assume $n > 1$. We extend A to A' by adding fresh actions from sets H and I:

1. $H \stackrel{\text{def}}{=} \{r_i(b), s_i(b) \mid i \in \{1, ..., n\}, b \in \{0, 1\}\}$, and
2. $I \stackrel{\text{def}}{=} \{c_i(b) \mid i \in \{1, ..., n\}, b \in \{0, 1\}\}$,

and define γ' on $(A' \setminus A) \times (A' \setminus A)$ by $\gamma'(r_i(b), s_i(b)) \stackrel{\text{def}}{=} c_i(b)$ for $i \in \{1, ..., n\}, b \in \{0, 1\}\}$, and undefined otherwise.

For $i = 1, ..., n$ we define the following $\mathrm{ACP}_\tau(A', \gamma')^{**}$ processes:

$$\begin{array}{lcl} Z_i & = & (\ r_i(1)\ ,\ \sum_{j=1}^{n} \alpha_{i,j} \cdot s_j(1)\ \ {}^{**}\ \ r_i(0)\ ,\ \beta_i \cdot (\|_{j \in \{1,\ldots,n\} \setminus \{i\}}\ s_j(0))\ ) \\ Z'_i & = & \sum_{j=1}^{n}(\alpha_{i,j} \cdot s_j(1) \cdot Z_i) + \beta_i \cdot (\|_{j \in \{1,\ldots,n\} \setminus \{i\}}\ s_j(0)) \end{array}$$

where $\|_{j \in \{1\}}\ x_j = x_1$ and $\|_{j \in \{1,\ldots,k+2\}}\ x_j = (\|_{j \in \{1,\ldots,k+1\}}\ x_j) \parallel x_{k+2}$. Note that Z_i, Z'_i are well defined, as $n > 1$.

An intuitive explanation:

$r_i(0) \equiv$ 'read instruction to terminate',

$r_i(1) \equiv$ 'read instruction to (re)activate',

$s_i(0) \equiv$ 'send the instruction that the i_{th} process terminates',

$s_i(1) \equiv$ 'instruct the i_{th} process to (re)activate'.

Let $\overline{P}_i$ be defined by

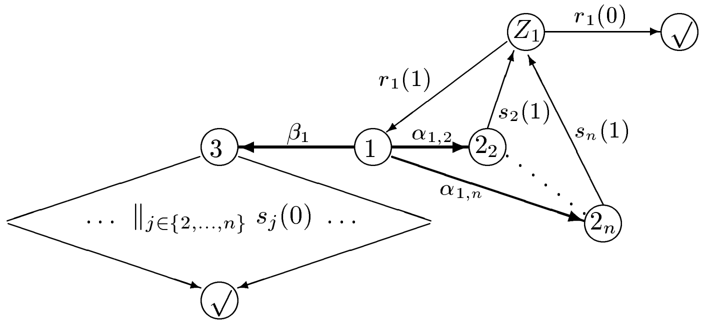
$$\overline{P}_i = \tau_I \circ \partial_H(Z'_i \parallel (\parallel_{j\in\{1,\ldots,n\}\setminus\{i\}} Z_j)).$$

Then the $\overline{P}_i$ solve the recursion equations for the P_i in $\mathrm{ACP}_\tau(A',\gamma')^{**}$: fix i_0, then

$$\begin{array}{rcl}
\overline{P}_{i_0} & = & \tau_I \circ \partial_H(Z'_{i_0} \parallel (\parallel_{k\in\{1,\ldots,n\}\setminus\{i_0\}} Z_k)) \\
& = & \sum_{j=1}^n \alpha_{i_0,j} \cdot \tau_I \circ \partial_H(s_j(1) \cdot Z_{i_0} \parallel (\parallel_{k\in\{1,\ldots,n\}\setminus\{i_0\}} Z_k)) + \\
& & \beta_{i_0} \cdot \tau_I \circ \partial_H((\parallel_{j\in\{1,\ldots,n\}\setminus\{i_0\}} s_j(0)) \parallel (\parallel_{j\in\{1,\ldots,n\}\setminus\{i_0\}} Z_j)) \\
& \overset{(1)}{=} & \sum_{j=1}^n \alpha_{i_0,j} \cdot \tau \cdot \tau_I \circ \partial_H(Z_{i_0} \parallel (\parallel_{k\in\{1,\ldots,n\}\setminus\{i_0,j\}} Z_k) \parallel \\
& & \qquad (\sum_{k=1}^n (\alpha_{j,k} \cdot s_k(1) \cdot Z_j) + \beta_j \cdot (\parallel_{k\in\{1,\ldots,n\}\setminus\{j\}} s_k(0))) \\
& & + \\
& & \beta_{i_0} \cdot (\parallel_{j\in\{1,\ldots,n\}\setminus\{i_0\}} \tau) \\
& = & \sum_{j=1}^n \alpha_{i_0,j} \cdot \overline{P}_j + \beta_{i_0}
\end{array}$$

as $\parallel$ is commutative and associative as a 'process constructor'. Note that in (1) it is essential that $\alpha_{i_0,i_0} = \delta$. With RSP it follows that $\overline{P}_i = P_i$.

Next we show that all Z_i, and therefore all $\overline{P}_i$, can be defined in $\mathrm{ACP}_\tau(B,\gamma)^*$ where B,γ are extensions of A' and γ' respectively. We first give a schematic picture of Z_1:



where 2_i for $i \in \{2,...,n\}$ are proper states if $\alpha_{1,i} \neq \delta$ (and non-existent otherwise) and the boldface arrows to the right represent all steps to 2_i, and the boldface arrow to the left represents all transitions defined by β_1 (provided $\beta_1 \neq \delta$).

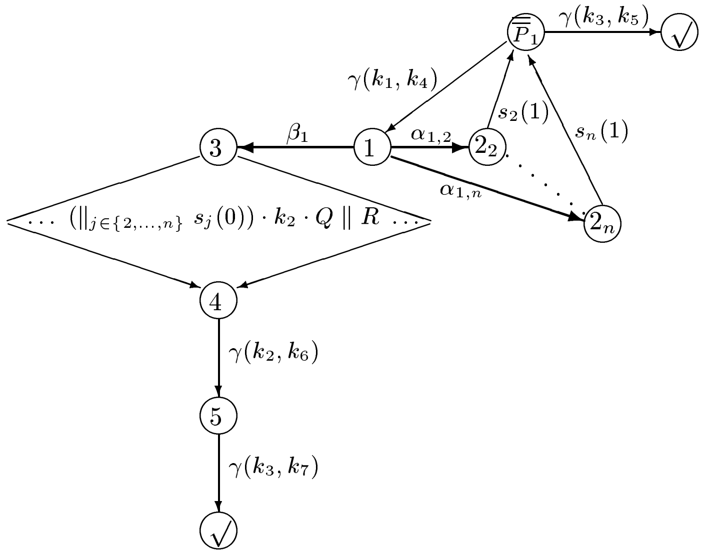
Now let A'' be defined as the extension of A' with (fresh) actions

$$\{i\} \cup \{k_1, k_2, ..., k_7\}$$

and consider the process $\overline{\overline{P}}_1$ defined by

$$\begin{array}{rcl}
\overline{\overline{P}}_1 & = & \partial_{\{k_1,\ldots,k_7\}}(Q \parallel R) \\
Q & = & (k_1 \cdot (\sum_{j=2}^n (\alpha_{1,j} \cdot s_j(1)) + \beta_1 \cdot (\parallel_{j\in\{2,\ldots,n\}} s_j(0)) \cdot k_2))^* k_3 \\
R & = & {k_4}^*(k_5 + k_6 \cdot k_7).
\end{array}$$

In the same style as above, we can depict $\overline{\overline{P}}_1$ schematically as follows, while showing in which cases the extension of γ' to γ'' on $(A'' \setminus A') \times (A'' \setminus A')$ is defined:



Now let γ'' be defined on $(A'' \setminus A') \times (A'' \setminus A')$ by

$$\begin{aligned}
\gamma''(k_3, k_5) &\stackrel{\text{def}}{=} r_1(0)\\
\gamma''(k_2, k_6) &\stackrel{\text{def}}{=} i\\
\gamma''(k_1, k_4) &\stackrel{\text{def}}{=} r_1(1)\\
\gamma''(k_3, k_7) &\stackrel{\text{def}}{=} i
\end{aligned}$$

and further undefined on this part of its domain. With RSP one can easily show in $\mathrm{ACP}_\tau(A'', \gamma'')^*$ that

$$\tau_{\{i\}}(\overline{\overline{P}}_1) = Z_1.$$

In a similar way $Z_2, ..., Z_n$ can be treated, so for some appropriate extension of A to B and of γ'' to γ, one can find a definition of the (arbitrarily chosen) regular process P_1 in $\mathrm{ACP}_\tau(B, \gamma)^*$. $\square$

An obvious open question is whether one can restrict to handshaking in the result above, i.e.,

4. *Can one find for each regular process P over A_δ an extension B of A such that P can be expressed in $\mathrm{ACP}_\tau(B, \gamma)^*$ with handshaking only, and the actions in A not subject to communication.*

Next we show that any regular process P over A_δ can also be defined in $\mathrm{ACP}(B, \gamma)^{**}$ with B a suitable extension of A (see for ** the defining equation (3) in Section 3.1). The proof uses n-ary communication if P can be specified with n equations in a certain restricted format (of course $n \geq 2$ is only of interest, otherwise P could be expressed in $\mathrm{BPA}_\delta(A)^*$).

Theorem 3.3.2. *For each regular process P over A_δ we can find an extension B of A such that P can be expressed in* $\mathrm{ACP}(B,\gamma)^{**}$ *with the actions in A not subject to communication.*

Proof. Let the regular process P_1 be given by

$$P_j = \sum_{i=1}^{n}(a_{i,j} \cdot P_{f(i,j)}) + \sum_{i=1}^{n} b_{i,j}$$

where $j = 1,...,n$ and $a_{i,j}, b_{i,j} \in A_\delta$, and $f : \{1,...,n\}^2 \to \{1,...,n\}$ satisfies $f(i,j) = j$ in case $a_{i,j} = \delta$. As in the previous proof, we can assume $n > 1$. We extend A to B by adding fresh actions (for $i,j \in \{1,...,n\}$) from a set $H \stackrel{\text{def}}{=} H_1 \cup H_2$:

1. $H_1 \stackrel{\text{def}}{=} \{start_k, stop_k, a_{i,j,k} \mid k \in \{1,...,n\}\}$, and
2. $H_2 \stackrel{\text{def}}{=} \{b_{i,j,V} \mid V \neq \emptyset, V \subseteq \{1,...,n\} \setminus \{j\}\}$,

and define γ on $(B \setminus A) \times (B \setminus A)$ by $\gamma(start_k, a_{i,j,k}) \stackrel{\text{def}}{=} a_{i,j}$,

$$\gamma(b_{i,j,V}, stop_k) \stackrel{\text{def}}{=} \begin{cases} b_{i,j,V\setminus\{k\}} & \text{if } k \in V, |V| > 1, \\ b_{i,j} & \text{if } V = \{k\}, \end{cases}$$

and undefined otherwise.
For $j = 1,...,n$ we define the following $\mathrm{ACP}(B,\gamma)^{**}$ processes:

$$\begin{array}{rcl} Z_j & = & (\ start_j\ ,\ \sum_{i=1}^{n} a_{i,j,f(i,j)}\ \ ^{**}\ \ stop_j\ ,\ \sum_{i=1}^{n} b_{i,j,\{1,...,n\}\setminus\{j\}}\ ) \\ Z'_j & = & (\sum_{i=1}^{n} a_{i,j,f(i,j)}) \cdot Z_j) + \sum_{i=1}^{n} b_{i,j,\{1,...,n\}\setminus\{j\}}. \end{array}$$

Let $\overline{P}_j$ for $j = 1,...,n$ be defined by

$$\overline{P}_j = \partial_H(Z'_j \parallel (\parallel_{i\in\{1,...,n\}\setminus\{j\}} Z_i))$$

It can be easily shown that the $\overline{P}_j$ solve the recursion equations for the P_j in $\mathrm{ACP}(B,\gamma)^{**}$. With RSP it follows that $\overline{P}_j = P_j$. Hence we find a definition of P_1 in $\mathrm{ACP}(B,\gamma)^{**}$. □

For termination-free regular processes we have a stronger result:

Theorem 3.3.3. *For each regular termination-free process P over A_δ we can find an extension B of A such that P can be expressed in* $\mathrm{ACP}(B,\gamma)^{*}$ *with handshaking only, and the actions in A not subject to communication.*

Proof. If all $b_{i,j}$ in the proof of Theorem 3.3.2 above equal δ, then the ** occurrences in the Z_j expressions can be replaced by their * equivalents. Moreover, the only communications then defined are binary. □

So dropping the distinction between (successful) termination and inaction/deadlock, and taking δ as *the* termination state (as is done in CCS [Mil80, Mil89]), implies that any such regular process over A_δ can be expressed in $\mathrm{ACP}(B,\gamma)^{*}$ $(B \supseteq A)$ with handshaking communication only.

3.4 Flat Iteration

A possibly interesting use of * is the restriction to *atomic sums* as left-arguments:

$$\begin{array}{l} a \in A \text{ is an atomic sum, and} \\ \sigma \text{ is an atomic sum} \quad \Longrightarrow \quad \sigma + a \text{ is an atomic sum.} \end{array}$$

In this case we speak of *flat iterative* processes. Let $\mathrm{BPA}(A)^{f*}$ processes be those processes that can be described with $\mathrm{BPA}(A)$ and ^{f*}, where ^{f*} is defined as * (by the axiom SEI), but also requires the left argument to be an atomic sum. Some typical examples are

$$(a+b)^{f*}(c+d) \quad \text{or} \quad (a+b)^{f*}((a+c)^{f*}d).$$

A particular property of flat iterative processes concerns the representation format defined below. Let *flat iterative basic terms* over $\mathrm{BPA}(A)^{f*}$ be defined by the following BNF clause:

$$p \quad ::= \quad a \mid a \cdot p \mid p + p \mid \sigma^{f*} p$$

where $a \in A$ and σ is an atomic sum. Flat iterative basic terms over $\mathrm{BPA}_\delta(A)^{f*}$ are defined by

$$p \quad ::= \quad \delta \mid a \mid a \cdot p \mid p + p \mid \sigma^{f*} p$$

where $a \in A$ and σ is an atomic sum.

Lemma 3.4.1. *Each* $\mathrm{BPA}(A)^{f*}$ [$\mathrm{BPA}_\delta(A)^{f*}$] *process can be proved equal to a flat iterative basic term over* $\mathrm{BPA}(A)^{f*}$ [$\mathrm{BPA}_\delta(A)^{f*}$, *respectively*] *with* SEI1 − SEI3 *from Table 8.*

Proof. By structural induction. □

On the expressivity of $\mathrm{BPA}(A)^{f*}$ and $\mathrm{BPA}_\delta(A)^{f*}$ we have the following result:

Theorem 3.4.2 (*Expressivity of flat iteration*)**.**

1. $\mathrm{BPA}(A)^{f*} \prec \mathrm{BPA}(A)^*$ *and* $\mathrm{BPA}_\delta(A)^{f*} \prec \mathrm{BPA}_\delta(A)^*$,
2. $\mathrm{BPA}(A)^{f*}$ *is as expressive as* $\mathrm{PA}(A)^{f*}$, *and*
3. $\mathrm{BPA}_\delta(A)^{f*}$ *is as expressive as* $\mathrm{ACP}(A,\gamma)^{f*}$.

Proof. The proof of 1 is trivial. Facts 2 and 3 can both be proved rather straightforward. By way of example we sketch part of a proof of 2:

From Lemma 3.4.1 it follows that $\mathrm{BPA}(A)^{f*}$ is as expressive as $\mathrm{PA}(A)^{f*}$ if $P \parallel Q$ and $P ⌊\!⌊ Q$ with P, Q flat iterative basic terms, are expressible in $\mathrm{BPA}(A)^{f*}$. This can be proved by induction on the complexity of flat iterative basic terms, occasionally using RSP. We show one of the inductive steps in which RSP is used; the case for flat iterative basic terms σ^*P and ρ^*Q: first observe that

$$\begin{aligned} (\sigma^*P) ⌊\!⌊ (\rho^*Q) &= (\sigma\cdot(\sigma^*P)+P) ⌊\!⌊ (\rho^*Q) \\ &= \sigma((\sigma^*P)\parallel(\rho^*Q)) + P ⌊\!⌊ (\rho^*Q). \end{aligned}$$

In the same way we find

$$(\rho^* Q) \mathbin{|\!\!\lfloor} (\sigma^* P) \quad = \quad \rho((\sigma^* P) \parallel (\rho^* Q)) + Q \mathbin{|\!\!\lfloor} (\sigma^* P)$$

(having used the commutativity of $\parallel$ in $\mathrm{PA}(A)^*$). By induction it follows that $P \mathbin{|\!\!\lfloor} (\rho^* Q)$ and $Q \mathbin{|\!\!\lfloor} (\sigma^* P)$ can be expressed in $\mathrm{BPA}(A)^{f*}$, say by $\overline{P}$ and $\overline{Q}$, respectively. So in $\mathrm{PA}(A)^{f*}$ we find

$$\begin{aligned}(\sigma^* P) \parallel (\rho^* Q) &= (\sigma^* P) \mathbin{|\!\!\lfloor} (\rho^* Q) + (\rho^* Q) \mathbin{|\!\!\lfloor} (\sigma^* P)\\ &= (\sigma + \rho) \cdot ((\sigma^* P) \parallel (\rho^* Q)) + \overline{P} + \overline{Q}.\end{aligned}$$

With RSP it follows that $(\sigma + \rho)^*(\overline{P} + \overline{Q})$ expresses the same process. So we have

i. $(\sigma^* P) \parallel (\rho^* Q) = (\sigma + \rho)^*(\overline{P} + \overline{Q})$,

ii. $(\sigma^* P) \mathbin{|\!\!\lfloor} (\rho^* Q) = \sigma((\sigma + \rho)^*(\overline{P} + \overline{Q})) + \overline{P}$, and

iii. $(\rho^* Q) \mathbin{|\!\!\lfloor} (\sigma^* P) = \rho((\sigma + \rho)^*(\overline{P} + \overline{Q})) + \overline{Q}$.

□

Note that parts 2 and 3 of the theorem above do not state that any $\mathrm{PA}(A)^*$ [$\mathrm{ACP}(A, \gamma)^*$] process can be *proved* equal to one defined over $\mathrm{PA}(A)^*$ [$\mathrm{ACP}(A, \gamma)^*$, respectively]. If one of the arguments of $\parallel$ contains an SOS-cyclic process, this occurrence of $\parallel$ cannot be eliminated with the axioms provided.

We conjecture that for processes defined over $\mathrm{BPA}(A)^{f*}$ [$\mathrm{BPA}_\delta(A)^{f*}$], bisimilarity is completely characterised by $\mathrm{BPA}(A)$ [$\mathrm{BPA}_\delta(A)$, respectively], and the axioms from Table 8.

4 Discussion

4.1 k-Exit Iteration

In the same way as ** is defined, we can also define the more general notion of k-Exit Iteration, say by an operator ^{*k}. Taking $^{*1} \stackrel{\text{def}}{=} {}^*$ and $^{*2} \stackrel{\text{def}}{=} {}^{**}$, the defining axioms for ^{*k} are given with help of expressions $\phi^k(x_1, ..., x_k, y_1, ..., y_k, z)$:

$$\begin{aligned}\phi^1(x, y, z) &= x \cdot z + y\\ \phi^{k+1}(x_1, ..., x_{k+1}, y_1, ..., y_{k+1}, z) &= \phi^k(x_1, ..., x_k, y_2, ..., y_{k+1}, x_{k+1} \cdot z + y_1)\end{aligned}$$

Then

$$(x_1, ..., x_k{}^{*k} y_1, ..., y_k) \stackrel{\text{def}}{=} \phi^k(x_1, ..., x_k\ ,\ y_1, ..., y_k\ ,\ (x_1, ..., x_k{}^{*k} y_1, ..., y_k)).$$

Let ^{f*k} be the restriction of ^{*k} to atomic sums as left arguments.

Two open questions are:

5. *Can each regular process P over A_δ be specified in* $\mathrm{BPA}_\delta(A)^{*k}$ *for some k?*
6. *Can each regular process P over A_δ be specified in* $\mathrm{BPA}_\delta(A)^{f*k}$ *for some k?*

4.2 Fairness and iteration

Due to the character of τ, one would also want to be able to abstract from infinite sequences of τ steps or *divergence*. Depending on the kind of process semantics one wants to use, different solutions have been found. In the case of weak bisimulation, a general solution is provided by *Koomen's Fair Abstraction Rule* (KFAR), introduced in [BK84b]. For each n and set of equations, there is a version KFAR_n. For example, the rule KFAR_1 reads as follows:

$$\frac{x = ix + y \quad (i \in I)}{\tau_I(x) = \tau \cdot \tau_I(y)}$$

(so the infinite τ sequence induced by ix is reduced to a single τ step). By definition of * we now have an immediate representation of the process in the premise of KFAR_1, namely i^*y. Henceforth we can represent KFAR_1 by the law

$$\tau_I(i^*y) = \tau \cdot \tau_I(y) \qquad (i \in I).$$

Given our 'distribution law' SEI6

$$\tau_I(x^*y) = \tau_I(x)^*\tau_I(y)$$

(see Table 9), we can even represent KFAR_1 simply by

$$\tau^*x = \tau \cdot x$$

(taking x for $\tau_I(y)$).

Divergence is treated differently in other process semantics. In for instance *failure semantics* [BHR84], fair abstraction can be modelled by a modification of KFAR to KFAR^- (see further [BKO87] on failure semantics with fair abstraction). In the recently developed *branching bisimulation semantics* [GW89, BW90], KFAR was modified to KFAR_1^b.

In each of the process semantics mentioned above, it is rather straightforward how the different KFAR versions give rise to their * counterparts, using the axiom SEI6. We call these counterparts *fair iteration rules* (FIRs), and summarise some of them in Table 11 below.

(FIR_1)	τ^*x	$=$	$\tau \cdot x$
(FIR_1^b)	$\tau \cdot \tau^*x$	$=$	$\tau \cdot x$
(FIR_1^-)	$\tau^*\tau$	$=$	τ
(FIR_2)	$(\tau, \tau^{**}x, y)$	$=$	$\tau(x+y)$
(FIR_2^b)	$\tau(\tau, \tau^{**}x, y)$	$=$	$\tau(x+y)$
(FIR_2^-)	$(\tau, \tau^{**}\tau \cdot x, \tau \cdot y)$	$=$	$\tau(\tau \cdot x + \tau \cdot y)$

Table 11: Some Fair Iteration Rules.

4.3 Some questions

We first repeat the open questions scattered throughout the paper:

1. (Section 3.1.) *Is* BPA(A) *extended with the axioms in Table 8 a complete axiomatisation of bisimilarity between* BPA$(A)^*$ *processes? And if not, what axioms or rules should be added?*
2. (Section 3.1.) *What axioms or rules should be added to Table 8 and* SEI4 *(see Table 9) to obtain a complete axiomatisation of bisimilarity between* BPA$_\delta(A)^*$ *processes?*
3. (Section 3.2.) *Under what minimal cardinality of A are the hierarchy results preserved?*
4. (Section 3.3.) *Can one find for each regular process P over A_δ an extension B of A such that P can be expressed in* ACP$_\tau(B,\gamma)^*$ *with handshaking only, and the actions in A not subject to communication?*
5. (Section 4.1.) *Can each regular process P over A_δ be specified in* BPA$_\delta(A)^{*k}$ *for some k?*
6. (Section 4.1.) *Can each regular process P over A_δ be specified in* BPA$_\delta(A)^{f*k}$ *for some k?*

As to questions 1 and 2, a related question is:

1.*a. Does* BPA(A) *extended with the axioms in Table 8 form a confluent term rewriting system modulo permutative reductions (replacing $=$ by $\to$)? And does the extension of* BPA$_\delta(A)$ *with the axioms in Table 8 and the axiom* SEI4 *form such a system?*

On question 2 we further remark that the axioms mentioned there are probably incomplete. As an example, consider the BPA$_\delta(A)^*$ processes $a^*\delta$ and $(aa)^*\delta$. Both processes are in each of their states only capable of performing an a step, and therefore bisimilar. However, it is not clear how their equality can be derived from the axioms mentioned. The following adaptation of RSP may give a solution:

$$(\mathrm{RSP}^*) \quad x = y \cdot x + z \quad \Longrightarrow \quad x = y^* z.$$

With RSP* it follows that $a^*\delta = (aa)^*\delta$, as $a^*\delta = (aa)\cdot(a^*\delta) + \delta$ is easily derivable. However, in the presence of τ, the rule RSP* as such is not sound: for instance $x = \tau \cdot x + \delta$ is satisfied by any process $\tau \cdot P$, but obviously $\tau \cdot P = \tau^*\delta$ is not true in weak bisimulation semantics (and anyhow an undesirable equality). Indeed the rule RSP takes as a general condition the *guardedness* of recursion equations (see for instance [BW90]), excluding the equation $x = \tau \cdot x + \delta$ above. In this paper we did not mention guardedness because RSP is only used in the setting with 'regular' processes, which satisfy guardedness by definition (on syntactical grounds).

The motivation for question 3 is primarily a technical one.

Question 4 reflects the interest in the restriction to binary communication, e.g. *read-write* communication.

An answer to question 5 may relate the expressivity of the parallel, encapsulation and abstraction operators of process algebra with "levels of iteration".

As for question 6, *flat* iteration is probably easier to study than ordinary (single exit) iteration. Furthermore, an approach to the k-exit generalisations of flat iteration may give new insights in complete axiomatisations, or a more general perspective on iteration.

References

[BB88] J.C.M. Baeten and J.A. Bergstra. Global renaming operators in concrete process algebra. *Information and Computation*, 78(3):205–245, 1988.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984.

[BK84a] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In J. Paredaens, editor, *Proceedings 11^{th} ICALP,* Antwerpen, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984.

[BK84b] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[BK86] J.A. Bergstra and J.W. Klop. Verification of an alternating bit protocol by means of process algebra. In W. Bibel and K.P. Jantke, editors, *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31*, pages 9–23, Berlin, 1986. Akademie-Verlag. First appeared as: Report CS-R8404, CWI, Amsterdam, 1984.

[BKO87] J.A. Bergstra, J.W. Klop, and E.-R. Olderog. Failures without chaos: a new process semantics for fair abstraction. In M. Wirsing, editor, *Formal Description of Programming Concepts – III, Proceedings of the 3^{th} IFIP WG 2.2 working conference,* Ebberup 1986, pages 77–103, Amsterdam, 1987. North-Holland.

[BT84] J.A. Bergstra and J.V. Tucker. Top down design and the algebra of communicating processes. *Science of Computer Programming*, 5(2):171–199, 1984.

[BV93] J. C. M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, editor, *Proceedings CONCUR'93* Hildesheim, Springer-Verlag, Berlin, to appear in the LNCS series, 1993. An earlier version appeared as Report CSN 93/05, Eindhoven University of Technology, 1993.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[CEW58] I.M. Copi, C.C. Elgot, and J.B. Wright. Realization of events by logical nets. *Journal of the ACM*, 5:181–196, 1958.

[Fer92] R.T.P. Fernando. Comparative transition system semantics. In E. Börger et al., editors, *Computer Science Logic: Selected Papers from CSL '92.* Springer-Verlag, Berlin, to appear. An earlier version appeared as CWI Report CS-R9222, July 1992.

[GW89] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.

[HY76] K. Hashiguchi and H. Yoo. Extended regular expressions of star degree at most two. *Theoretical Computer Science*, 76:273–284, 1976.

[Kle56] S.C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.

[Mil89] R. Milner. *Communication and concurrency.* Prentice Hall International, 1989.

[MP43] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Pon92] A. Ponse. Process algebra and dynamic logic. In D.J.N. van Eijck and A. Visser, editors, *Logic and the Flow of Information*, MIT Press, to appear. An earlier version appeared as Report P9217, University of Amsterdam, 1992.

[Tro93] D.R. Troeger. Step bisimulation is pomset equivalence on a parallel language without explicit internal choice. *Mathematical Structures in Computer Science*, 3:25–62, 1993.