Process Algebra with Iteration and Nesting

JAN A. BERGSTRA*,†, INGE BETHKE†,‡ AND ALBAN PONSE*

*University of Amsterdam, Programming Research Group, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
†Department of Philosophy, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
‡CWI, Department of Software Technology, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

We introduce iteration in process algebra by means of (the original, binary version of) Kleene's star operation: x^*y is the process that chooses between x and y, and upon termination of x has this choice again. We add this operation to a whole range of process algebra axiom systems, starting from BPA (Basic Process Algebra). In the case of the most complex system under consideration, ACP_τ, every regular process can be defined with handshaking (two-party communication) and auxiliary actions. Next we introduce nesting in process algebra: $x^\sharp y$ is defined by the equation $x^\sharp y = x(x^\sharp y)x + y$. We show that * and $^\sharp$ are not interdefinable in most of the axiom systems we regard. The extension with $^\sharp$, and the extension with * and $^\sharp$ of the systems considered also give a genuine hierarchy in expressivity. Finally, it is argued that each finitely branching, computable graph can be defined in ACP_τ extended with * and $^\sharp$, and using handshaking and auxiliary actions.

While writing this paper, we heard of the news that Stephen C. Kleene has died in January 1994. We acknowledge the substantial influence he has had on our work.

1. INTRODUCTION

Systems of recursion equations play a fundamental role in process algebra as a means to specify or to analyze infinite behaviour. The purpose of this paper is to introduce two operations that give sufficient expressive power to study infinite behaviour in process algebra without such systems.

In 1956, Kleene introduced in [16] the binary operation * for describing 'regular events'. 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). Kleene 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.

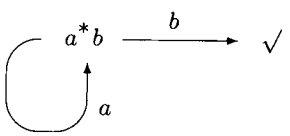
In [11], Copi, Elgot and Wright showed interest in the results in [16]. However, they judged in particular Kleene's theorems on analysis* and synthesis† obscured both by the complexity of his basic concepts and by the nature of the elements used in his nets. These authors introduced simpler and stronger 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 * operation

> "[...] because the operation Kleene uses seems "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 in [16], we introduce x^*y in process algebra with defining equation

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

So x^*y is the process that chooses between x and y, and upon termination of x has this choice again. For example, the process a^*b for actions a and b can be depicted by



where $\surd$ is a symbol expressing (successful) termination.

The operation * can be added to, for example, the following process algebra axiom systems: BPA, PA, ACP, ACP_τ (see [1] and [6]). In each case many models are known that allow to solve the equation $P = x \cdot P + y$. Most of 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.

*Theorem 5, stating that finite automata model regular events.
†Theorem 3, stating that each regular event can be described by a finite automaton ('nerve net').

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

$$x^*\epsilon = x \cdot (x^*\epsilon) + \epsilon.$$

According to [19] (inspired by [13]), the *unary* operation $\_^*\epsilon$ is a plausible candidate for the unary version of Kleene's star operation in process algebra. Note that the unary operation $\_^*\epsilon$ cannot be used in a setting without having ϵ available as a separate process (adopting Kleene's defining equation). Moreover, with ϵ the interdefinability of the unary and the binary star, noted in [11], is preserved.

Obvious as ϵ may be (being a unit for $\cdot$), its introduction is non-trivial 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 study versions of the theory, both with and without ϵ. Just for this reason we propose the star operation with its (original) defining equation given by Kleene in [16].

In [17], 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 and raised the questions of a complete axiomatization, and what structural properties of (finite) transition systems characterize star behaviours.

In [20], Troeger introduces a process specification language with iteration, writing y *wh* x (y *while* x) for x^*y. This work contains an interesting axiom for *:

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

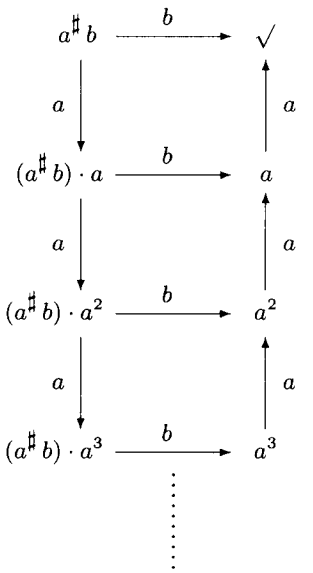
(see further Section 3 for axioms on *). However, Troeger concentrates on semantical characterization results for the restriction of the language obtained by excluding (explicit) internal choice.‡ Therefore, the main results in [20] refer to a different setting.

We call the operation * BKS (Binary Kleene Star), in order to give self-explanatory naming for other, related iteration operations and to avoid confusion about the arity of what is nowadays known as 'Kleene star'.

In this paper we also introduce the non-regular operation $^\sharp$ (*sharp*), which we call the Nesting Operation (NO). This operation is defined by

$$x^\sharp y = x \cdot (x^\sharp y) \cdot x + y.$$

So $x^\sharp y$ is the process that chooses between x and y, and upon termination of x has the choice to perform x again, or to perform y and upon termination of y repeat x the number of times it has already been performed. As an example, the process $a^\sharp b$ for actions a and b can be illustrated by



The main results in this paper concern the expressivity of ACP systems enriched with * and/or $^\sharp$. First, any regular process can be specified in ACP_τ with *, using only 'handshaking' (i.e. two-party communication). Secondly, it turns out that every finitely branching, computable graph modulo rooted τ (or *weak*) bisimulation can be expressed in ACP_τ with * and $^\sharp$ restricting to handshaking. A basic reference for a proof of this second result is [3], where it is shown that each finitely branching, computable graph is recursively definable by a finite guarded specification over ACP_τ.

Many of the results on iteration in this paper have been documented in the report [10] which is superseded by the present paper. Furthermore, we notice that $\mathrm{ACP}_\tau^\sharp$ was introduced in [8] as an abbreviation for ACP_τ extended with a particular group of often used axioms and proof rules. This abbreviation we now consider obsolete.

2. AXIOMS AND TRANSITION RULES

In this section we introduce various process algebra axiom systems, associated operational semantics using bisimulations, and recursive specifications. For a detailed introduction to these matters, see e.g. [1].

2.1. The systems $\mathrm{BPA}(A) - \mathrm{ACP}_\tau(A, \gamma)$

Let $\mathcal{P}$ be the sort of processes under consideration, containing a finite set A of actions $a, b, \ldots$. Let furthermore $\gamma : A \times A \to A$ be a partial function that is *commutative* and *associative* (i.e., $\gamma(a, \gamma(b, c)) = \gamma(\gamma(a, b), c)$). We write $\gamma(a, b) \downarrow$ if $\gamma(a, b)$ is defined, and $\gamma(a, b) \uparrow$ otherwise. We further write $\gamma(a, b, c)$ instead of $\gamma(a, \gamma(b, c))$. The function γ defines *communication actions* and models the simultaneous execution of actions. In the case that $\gamma(a, b, c) \uparrow$ for all $a, b, c \in A$, we speak of *handshaking* (two-party communication, see [9]).

‡I.e. excluding expressions such as $a \cdot P + a \cdot Q$ or $(a \cdot P \,\|\, b \cdot Q) + a \cdot R$.

TABLE 2. Axioms of Standard Concurrency (SC), with $a \in A$

SC1	$(x \between y) \between z$	$=$	$x \between (y \parallel z)$
SC2	$(x \mid ay) \between z$	$=$	$x \mid (ay \between z)$
SC3	$x \mid y$	$=$	$y \mid x$
SC4	$x \parallel y$	$=$	$y \parallel x$
SC5	$x \mid (y \mid z)$	$=$	$(x \mid y) \mid z$
SC6	$x \parallel (y \parallel z)$	$=$	$(x \parallel y) \parallel z$

The actions in A and the communication function γ can be regarded as parameters of the theory $\mathrm{ACP}_\tau(A, \gamma)$. In Table 1 the signature and axioms of the system $\mathrm{ACP}_\tau(A, \gamma)$ are collected. We take $\cdot$ to be the operation that binds strongest, and $+$ the one that binds weakest. As usual in algebra, we often write xy instead of $x \cdot y$. Furthermore, for $n > 0$ we define x^{n+1} as $x \cdot x^n$, and x^1 as x.

The following five subsystems of $\mathrm{ACP}_\tau(A, \gamma)$ have been introduced in [6, 7]:

$\mathrm{BPA}(A)$. The signature of $\mathrm{BPA}(A)$ contains the elements of A, and the operations $+$ and $\cdot$. The axiom system $\mathrm{BPA}(A)$ contains the axioms A1–A5.

$\mathrm{BPA}_\delta(A)$. The signature of $\mathrm{BPA}_\delta(A)$ is the signature of $\mathrm{BPA}(A)$ extended with δ. We often write A_δ for $A \cup \{\delta\}$. The axioms of $\mathrm{BPA}_\delta(A)$ are A1–A7.

$\mathrm{PA}(A)$. The signature of $\mathrm{PA}(A)$ contains the elements of A, the operations $+$, $\cdot$, $\between$ and the *free merge* $\parallel$. The axioms of $\mathrm{PA}(A)$ are A1–A5, the axiom

(M1) $x \parallel y = x \between y + y \between x$

and the axioms CM2–CM4 (in the case of $\mathrm{PA}(A)$ these are referred to as M2–M4). So $\mathrm{PA}(A)$ processes do not communicate.

$\mathrm{PA}_\delta(A)$. The signature of $\mathrm{PA}_\delta(A)$ is the signature of $\mathrm{PA}(A)$ extended with δ. The axioms of $\mathrm{PA}(A)$ are A1–A7 and M1–M4.

$\mathrm{ACP}(A, \gamma)$. The signature and axioms of $\mathrm{ACP}(A, \gamma)$ are defined by all axioms in the left-hand side column of Table 1.

Standard Concurrency. Standard Concurrency (SC) consists of the axioms given in Table 2 and provides the axiomatic support for reasoning with the parallel operators (see [9]). The axiom system given in Table 1 allows one to derive the SC identities for closed instances by structural induction and the axioms CF1, CF2 (see [7]).

2.2. Transition rules and bisimulation semantics

We define a structural operational semantics, and relate processes to transition systems, providing an operational semantics in the style of Plotkin [18]. Then we define bisimilarity as an equivalence between transition systems. We use this operational semantics and bisimilarity in the proofs of some of the classification results in Sections 3 and 4. Observe that various notions are defined relative to our specific choice of the operational semantics.

A *transition system* is a tuple $\langle S, L, T, s \rangle$, where

S is a set of *states*,
L is a set of *labels*,
$T \subseteq (S \times L \times S) \cup (S \times \mathrm{L})$ is a *transition relation*,
$s \in S$ is the *initial state*.

Let $E(A)$ be one of the process algebra axiom systems $\mathrm{BPA}(A) - \mathrm{ACP}(A, \gamma)$, and let $\mathcal{P}(E(A))$ represent all processes over $E(A)$. As the set of states S we take $\mathcal{P}(E(A))$. As labels we take the atomic actions in A. The transition relation T contains transitions

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

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

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

(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 3 define the transition relation T. The signature and parameters of $E(A)$ (possibly including a communication function γ) determine which rules are appropriate. For example, the last four rules for $\parallel$ are not relevant for $\mathrm{PA}(A)$. Note that the state δ has no outgoing transitions. Finally, the transition system related to a process P has P itself as initial state. If $E(A)$ is fixed, we often write simply P for the transition system

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

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

$$P \mathcal{R} Q \;\&\; P \xrightarrow{a} P' \text{ implies } \exists Q' \cdot Q \xrightarrow{a} Q' \;\&\; P' \mathcal{R} Q', \text{ and}$$

$$P \mathcal{R} Q \text{ implies } (P \xrightarrow{a} \surd \iff Q \xrightarrow{a} \surd).$$

Two states P and Q are called *bisimilar*, notation

$$P \leftrightarroweq Q,$$

if there exists a bisimulation $\mathcal{R}$ with $P \mathcal{R} Q$.

TABLE 1. Signature and axioms of $\mathrm{ACP}_\tau(A,\gamma)$ where $a,b \in A_\delta$, $H, I \subseteq A$

sorts:	A	(a given, finite set of actions),
	$\mathcal{P}$	(the set of processes; $A \subseteq \mathcal{P}$),
operations:	$+ : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$	(alternative composition or sum),
	$\cdot : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$	(sequential composition or product),
	$\parallel : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$	(parallel composition or merge),
	$\lfloor\!\lfloor : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$	(left merge),
	$\mid : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$	(communication merge, $\gamma : A \times A \to A$ is given),
	$\partial_H : \mathcal{P} \to \mathcal{P}$	(encapsulation, $H \subseteq A$),
	$\tau_I : \mathcal{P} \to \mathcal{P}$	(abstraction, $I \subseteq A$),
constants:	$\delta \in \mathcal{P} \setminus A$	(deadlock or inaction),
	$\tau \in \mathcal{P} \setminus A$	(silent or internal action).

(A1)	$x+y$	$=$	$y+x$		(T1)	$x\tau$	$=$	x
(A2)	$x+(y+z)$	$=$	$(x+y)+z$		(T2)	$\tau x + x$	$=$	τx
(A3)	$x+x$	$=$	x		(T3)	$a(\tau x + y)$	$=$	$a(\tau x + y) + ax$
(A4)	$(x+y)z$	$=$	$xz+yz$					
(A5)	$(xy)z$	$=$	$x(yz)$					
(A6)	$x+\delta$	$=$	x					
(A7)	δx	$=$	δ					
(CF1)	$a \mid b$	$=$	$\gamma(a,b)$	if $\gamma(a,b)\downarrow$				
(CF2)	$a \mid b$	$=$	δ	otherwise				
(CM1)	$x \parallel y$	$=$	$x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x \mid y$					
(CM2)	$a \lfloor\!\lfloor x$	$=$	ax		(TM1)	$\tau \lfloor\!\lfloor x$	$=$	τx
(CM3)	$ax \lfloor\!\lfloor y$	$=$	$a(x \parallel y)$		(TM2)	$\tau x \lfloor\!\lfloor y$	$=$	$\tau(x \parallel y)$
(CM4)	$(x+y) \lfloor\!\lfloor z$	$=$	$x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$		(TC1)	$\tau \mid x$	$=$	δ
(CM5)	$ax \mid b$	$=$	$(a \mid b)x$		(TC2)	$x \mid \tau$	$=$	δ
(CM6)	$a \mid bx$	$=$	$(a \mid b)x$		(TC3)	$\tau x \mid y$	$=$	$x \mid y$
(CM7)	$ax \mid by$	$=$	$(a \mid b)(x \parallel y)$		(TC4)	$x \mid \tau y$	$=$	$x \mid y$
(CM8)	$(x+y) \mid z$	$=$	$x \mid z + y \mid z$					
(CM9)	$x \mid (y+z)$	$=$	$x \mid y + x \mid z$					
					(DT)	$\partial_H(\tau)$	$=$	τ
					(TI1)	$\tau_I(\tau)$	$=$	τ
(D1)	$\partial_H(a)$	$=$	a	if $a \notin H$	(TI2)	$\tau_I(a)$	$=$	a if $a \notin I$
(D2)	$\partial_H(a)$	$=$	δ	if $a \in H$	(TI3)	$\tau_I(a)$	$=$	τ if $a \in I$
(D3)	$\partial_H(x+y)$	$=$	$\partial_H(x) + \partial_H(y)$		(TI4)	$\tau_I(x+y)$	$=$	$\tau_I(x) + \tau_I(y)$
(D4)	$\partial_H(xy)$	$=$	$\partial_H(x) \cdot \partial_H(y)$		(TI5)	$\tau_I(xy)$	$=$	$\tau_I(x) \cdot \tau_I(y)$

TABLE 3. Transition rules for BPA(A) – ACP(A, γ), where $a, b \in A$, $H \subseteq A$

$a \in A$	$a \xrightarrow{a} \surd$			
$+$	$\dfrac{x \xrightarrow{a} x'}{x+y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x+y \xrightarrow{a} y'}$	$\dfrac{x \xrightarrow{a} \surd}{x+y \xrightarrow{a} \surd}$	$\dfrac{y \xrightarrow{a} \surd}{x+y \xrightarrow{a} \surd}$
$\cdot$	$\dfrac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$	$\dfrac{x \xrightarrow{a} \surd}{x \cdot y \xrightarrow{a} y}$		
$\parallel$	$\dfrac{x \xrightarrow{a} x'}{x \parallel y \xrightarrow{a} x' \parallel y}$	$\dfrac{y \xrightarrow{a} y'}{x \parallel y \xrightarrow{a} x \parallel y'}$	$\dfrac{x \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} y}$	$\dfrac{y \xrightarrow{a} \surd}{x \parallel y \xrightarrow{a} x}$
	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \parallel y \xrightarrow{\gamma(a,b)} x' \parallel y'}$ if $\gamma(a,b)\downarrow$		$\dfrac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} y}{x \parallel y \xrightarrow{\gamma(a,b)} \surd}$ if $\gamma(a,b)\downarrow$	
	$\dfrac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} y'}{x \parallel y \xrightarrow{\gamma(a,b)} y'}$ if $\gamma(a,b)\downarrow$		$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} \surd}{x \parallel y \xrightarrow{\gamma(a,b)} x'}$ if $\gamma(a,b)\downarrow$	
$\between$	$\dfrac{x \xrightarrow{a} x'}{x \between y \xrightarrow{a} x' \parallel y}$	$\dfrac{x \xrightarrow{a} \surd}{x \between y \xrightarrow{a} y}$		
$\mid$ provided $\gamma(a,b)\downarrow$	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{x \mid y \xrightarrow{\gamma(a,b)} x' \parallel y'}$	$\dfrac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} y'}{x \mid y \xrightarrow{\gamma(a,b)} y'}$	$\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} \surd}{x \mid y \xrightarrow{\gamma(a,b)} x'}$	$\dfrac{x \xrightarrow{a} \surd \quad y \xrightarrow{b} y}{x \mid y \xrightarrow{\gamma(a,b)} \surd}$
∂_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 $a \notin H$	

THEOREM 2.1

1. (Soundness) *Bisimilarity is a congruence relation with respect to all operators defined (for a 'modern' proof see [2]), and all axioms presented are valid in bisimulation semantics.*
2. (Completeness) *Up to and including* ACP(A, γ) *all axiom systems are complete with respect to bisimilarity between the processes that can be expressed (without recursion, see [1]).*

In the sequel we use the following notions based on our operational semantics:

1. An action $a \in A$ is an *exit* of P if $P \xrightarrow{a} \surd$.
2. P is a *successor of* Q if $Q \xrightarrow{a} P$ for some $a \in A$.
3. Let the relation $\xrightarrow{w}\!\!\!\!\rightarrow$ for $w \in A^*$ with ϵ denoting the empty string be defined as follows:

   (a) $P \xrightarrow{\epsilon}\!\!\!\!\rightarrow P$,

   (b) $\dfrac{P \xrightarrow{w}\!\!\!\!\rightarrow Q \quad Q \xrightarrow{a} R}{P \xrightarrow{wa}\!\!\!\!\rightarrow R}$,

   (c) $\dfrac{P \xrightarrow{w}\!\!\!\!\rightarrow Q \quad Q \xrightarrow{a} \surd}{P \xrightarrow{wa}\!\!\!\!\rightarrow \surd}$.

   Then Q is a *proper substate* of P if $P \xrightarrow{w}\!\!\!\!\rightarrow Q$ for some $w \in A^* \setminus \{\epsilon\}$.
4. P is a *state of* Q if either $P \equiv Q$, or P is a proper substate of Q.
5. If $P \xrightarrow{w}\!\!\!\!\rightarrow \surd$, then w is called a *trace* of P.
   If $P \xrightarrow{w}\!\!\!\!\rightarrow P_0 \xrightarrow{w_0}\!\!\!\!\rightarrow \ldots P_i \xrightarrow{w_i}\!\!\!\!\rightarrow P_{i+1} \ldots$ for $i \in \mathbb{N}$, then $ww_0 \ldots w_i \ldots$ is called an *infinite trace* of P, notation $P \xrightarrow{ww_0 \ldots w_i \ldots}\!\!\!\!\rightarrow$.

τ-bisimilarity. Transition rules for ACP$_\tau(A, \gamma)$ can be obtained by extending the rules in Table 3 to a ranging over $A_\tau \stackrel{\text{def}}{=} A \cup \{\tau\}$, and including the rules defining τ_I transitions displayed in Table 4 below.

Let $P, P' \in \mathcal{P}$. We define the binary relation $\Longrightarrow$ by $P \Longrightarrow P'$ if either $P \equiv P'$ or $P \xrightarrow{\tau} P'' \Longrightarrow P'$ for some $P'' \in \mathcal{P}$, and the unary relation $P \Longrightarrow \surd$ if $P \Longrightarrow P' \xrightarrow{\tau} \surd$. Furthermore, we define $P \overset{a}{\Longrightarrow} P'$ if $P \Longrightarrow Q \xrightarrow{a} Q' \Longrightarrow P'$, and $P \overset{a}{\Longrightarrow} \surd$ if either $P \Longrightarrow Q \xrightarrow{a} Q' \Longrightarrow \surd$ or $P \Longrightarrow Q \xrightarrow{a} \surd$ for some $Q, Q' \in \mathcal{P}$. Observe that the relations $\overset{\tau}{\Longrightarrow}$ and $\Longrightarrow$ coincide (as do the $\surd$-variants of both).

Now a *τ-bisimulation* is a binary, symmetric relation $\mathcal{R} \subseteq (\mathcal{P} \cup \{\surd\}) \times (\mathcal{P} \cup \{\surd\})$ that satisfies

$$P \mathcal{R} Q \ \ \& \ \ P \xrightarrow{a} P' \text{ implies}$$
$$(a = \tau \ \ \& \ \ P' \mathcal{R} Q), \text{ or}$$
$$\exists Q' \,.\, Q \overset{a}{\Longrightarrow} Q' \ \ \& \ \ P' \mathcal{R} Q', \text{ where } Q' \text{ may be } \surd,$$

$$P \mathcal{R} Q \ \ \& \ \ P \xrightarrow{a} \surd \text{ implies}$$
$$Q \overset{a}{\Longrightarrow} \surd \text{ or } (a = \tau \ \ \& \ \ Q \equiv \surd).$$

Two $\mathrm{ACP}_\tau(A, \gamma)$ processes P and Q are called *rooted τ-bisimilar* if there exists a τ-bisimulation $\mathcal{R}$ with $P \mathcal{R} Q$ that moreover satisfies

$$P \xrightarrow{\tau} P' \text{ implies}$$
$$\exists Q' \,.\, Q \xrightarrow{\tau} Q' \ \ \& \ \ P' \mathcal{R} Q' \text{ where } P', Q' \text{ may be } \surd.$$

Also $\mathrm{ACP}_\tau(A, \gamma)$ is sound and complete with respect to rooted τ-bisimilarity between the processes that can be expressed (without recursion, this follows from [7]).

TABLE 4. Transition rules for τ_I, where $a \in A_\tau$, $I \subseteq A$

$$\frac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')} \text{ if } a \in I \qquad \frac{x \xrightarrow{a} \surd}{\tau_I(x) \xrightarrow{\tau} \surd} \text{ if } a \in I$$

$$\frac{x \xrightarrow{a} x'}{\tau_I(x) \xrightarrow{a} \tau_I(x')} \text{ if } a \notin I \qquad \frac{x \xrightarrow{a} \surd}{\tau_I(x) \xrightarrow{a} \surd} \text{ if } a \notin I$$

2.3. Recursion

Though the operations * and $^\sharp$ introduce recursion into the axiom systems to be provided, we use recursive specifications and associated notions and rules to prove some of our results.

Definition 2.1 *A* (recursive) specification $E = \{E_j \mid j \in J\}$ *is a set of equations in the signature of* $\mathrm{ACP}_\tau(A, \gamma)$ *with variables* $\{X_j \mid j \in J\}$ *for some index set J such that equation E_j has the form $X_j = T_j$, where T_j is a finite $\mathrm{ACP}_\tau(A, \gamma)$-term (with finitely many variables X_i for i ranging over J). Given some process semantics, processes P_j $(j \in J)$ are a solution of E if substitution of P_j for X_j in the equations of E yields equations that hold in this semantics.*

Specifications need not define processes in any reasonable semantics, a clear example being $\{X = X\}$.

Definition 2.2 *Let P be an expression containing a variable X. An occurrence of X in P is* τ-guarded *if P has a subexpression $a \cdot Q$, where $a \in A_\tau$ and Q contains this occurrence of X.*

We call a specification $E = \{X_j = T_j \mid j \in J\}$ τ-guarded *if by substituting T_j expressions for X_j occurrences in the right-hand sides a finite number of times, one can obtain the situation that every occurrence of every variable X_j is τ-guarded.*

We claim that τ-guardedness and another restriction depending on our operational semantics are sufficient to guarantee a *unique* solution (per equation). We discuss this restriction below and first introduce the following convention in order to define transition rules in a simple way. If a specification has solutions in terms of transition systems (and some notion of equality over these), then these solutions are referred to by the variable names declared in E.

Adopting the convention above, we give in Table 5 the transition rules for recursive specifications.

TABLE 5. Transition rules for a recursive specification E, where $a \in A_\tau$

$$\text{For } X_j = T_j \in E: \quad \frac{T_j \xrightarrow{a} x'}{X_j \xrightarrow{a} x'} \qquad \frac{T_j \xrightarrow{a} \surd}{X_j \xrightarrow{a} \surd}$$

If for instance $E = \{X = aX + b\}$, then $X \xrightarrow{a} X$ by the first transition rule and $aX + b \xrightarrow{a} X$, and $X \xrightarrow{b} \surd$ can be derived using the second rule.

Definition 2.3 *A specification $E = \{X_j = T_j \mid j \in J\}$ is* τ-convergent *or* τ-founded *if no X_j has an infinite τ-trace, i.e., if $X_j \xrightarrow{w} P$ then not $P \xrightarrow{\tau^\omega}$.*

E is τ-divergent *if E is not τ-convergent.*

Let $E = \{E_j \mid j \in J\}$ be a τ-convergent specification, and I be some index set disjunct from J. The specification $F = \{Y_i = T_i \mid i \in I\} \cup E$ is τ-semi-convergent *if $Y_i = T_i(\overline{X})$ for all $i \in I$, where all T_i are expressions over the signature of $\mathrm{ACP}_\tau(A, \gamma)$ possibly containing variables from $\{X_j \mid j \in J\}$, but not containing variables Y_i.*

Now recursive specifications that are both τ-guarded and τ-semi-convergent, are assumed to define processes as solutions of each of their variables (this can be seen as a model dependent analogon of RDP, the Recursive Definition Principle, see, e.g. [3]). Note that the notion τ(-semi)-convergence depends on the semantics we use, i.e., on the definition of transition systems.

For reasoning with recursive specifications, we introduce the fixed point rule RSP (Recursive Specification Principle, discussed in for instance [3, 5]). This rule

states that every specification E has at most one solution per variable:

$$\text{(RSP)} \quad \frac{E(x,-) \quad E(y,-)}{x = y}$$

We claim that RSP is a sound rule with respect to the bisimulation semantics introduced thus far (now including transition systems identified by recursion variables) *provided* E is both τ-guarded *and* τ-convergent. In this case E has a unique solution (per variable). So the example specification $\{X = X\}$, or for another example $\{Y = \tau \cdot Y\}$ cannot be used in RSP in a sound way.

Typically, specifications that are both τ-guarded *and* τ-convergent can be used to prove that some τ-semi-convergent specification defines a certain process.

Definition 2.4 *A process P_1 is* regular *over A_δ if P_1 is a solution for X_1 from a finite specification*

$$\{X_i = \sum_{j=1}^{n}(\alpha_{i,j} \cdot X_j) + \beta_i \mid i = 1, ..., n\}$$

where $\alpha_{i,j}$ and β_i are finite sums of actions or δ.

Observe that a specification defining a regular process is both τ-guarded and τ-convergent. As an example, the specification $E = \{X = aX + b\}$ defines a regular process.

3. ITERATION

In Section 1 we introduced Kleene's star operation * in process algebra, and called this operation BKS (Binary Kleene Star) with defining axiom

$$x^*y = x(x^*y) + y.$$

In this section we add * to the axiom systems introduced, and formulate various expressivity results. Finally, we introduce a 'fair iteration rule' for expressing a notion of fairness.

3.1. Axioms and transition rules

In Table 6 we introduce the axiom system BPA$^*(A)$, obtained by adding the BKS axioms (BKS1) – (BKS3) to BPA(A).

The third iteration axiom BKS3 stems from [20], where it is used for a slightly different process specification formalism. Some typical BPA$^*(A)$ identities are

$x^*((x+y)^*z) = (x+y)^*z$ (substitute $x+y$ for y in BKS3, and apply BKS1, A2, A3), and

$x^*(x^*y) = x^*y$ (apply BKS1, A3 and BKS3).

The systems BPA$^*_\delta(A)$ – PA$^*_\delta(A)$ are defined by inclusion of the BKS axioms (BKS1) – (BKS3).

In Table 7 we give some more axioms for BKS. The system ACP$^*(A,\gamma)$ is defined by inclusion of (BKS1) – (BKS4), and the system ACP$^*_\tau(A,\gamma)$ by inclusion of the BKS axioms (BKS1) – (BKS5).

TABLE 6. The axiom system BPA$^*(A)$

(A1)	$x+y$	$=$	$y+x$
(A2)	$x+(y+z)$	$=$	$(x+y)+z$
(A3)	$x+x$	$=$	x
(A4)	$(x+y)z$	$=$	$xz+yz$
(A5)	$(xy)z$	$=$	$x(yz)$
(BKS1)	$x\cdot(x^*y)+y$	$=$	x^*y
(BKS2)	$x^*(y\cdot z)$	$=$	$(x^*y)\cdot z$
(BKS3)	$x^*(y\cdot((x+y)^*z)+z)$	$=$	$(x+y)^*z$

TABLE 7. Some more axioms for BKS

(BKS4)	$\partial_H(x^*y)$	$=$	$\partial_H(x)^*\partial_H(y)$
(BKS5)	$\tau_I(x^*y)$	$=$	$\tau_I(x)^*\tau_I(y)$

Transition rules. Given our description of BKS, the transition rules for this operation defined in Table 8 are quite obvious.

TABLE 8. Transition rules for BKS, where $a \in A_\tau$

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

$$\frac{y \xrightarrow{a} y'}{x^*y \xrightarrow{a} y'} \qquad \frac{y \xrightarrow{a} \surd}{x^*y \xrightarrow{a} \surd}$$

Note that with the first two transition rules, a state P can have itself as a *proper* substate, e.g. a^*b by the transition $a^*b \xrightarrow{a} a^*b$.

It can be easily shown that the BKS axioms are valid in bisimulation semantics, and according to [2], bisimilarity is a congruence with respect to *.

In [12], Fokkink and Zantema prove that bisimilarity between BPA$^*(A)$ processes is axiomatized by the axioms of BPA$^*(A)$.

We now introduce some further terminology on a special kind of regular processes, namely the 'cyclic' ones. A *chain* in a binary relation $\mathcal{R}$ over 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 a *cycle* if the P_i are all syntactically distinct and also $(P_n, P_0) \in R$. A *trivial* cycle contains one state P_0 such that $(P_0, P_0) \in \mathcal{R}$. A cycle is *non-trivial* if it is not trivial. A state P is *root-cyclic* if P has a cycle starting at P with the 'successor' relation as $\mathcal{R}$. A state P is *cyclic* if P has a state Q that is root-cyclic. For example, $a(a^*b)+b$ is cyclic: a^*b is a proper substate that is root-cyclic.

LEMMA 3.1 *Let P be a process expression over $\mathrm{ACP}^*_\tau(A,\gamma)$ or one of the smaller signatures. Then P has finitely many states.*

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

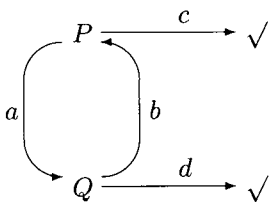
Proof By induction on the maximal depth of the *-nestings in P. The second part of the lemma follows immediately. ■

We end this section by showing that a particular, simple regular process can be specified in $\mathrm{ACP}^*_\tau(A,\gamma)$ *without* its actions being the result of communications.

LEMMA 3.2 *Let the regular process P be defined by*

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

or graphically:



Then P can be expressed in $\mathrm{ACP}^*_\tau(A,\gamma)$ with handshaking (and some auxiliary actions) without its actions being subject to communication.

Proof Assume $H \stackrel{\text{def}}{=} \{k_1,k_2,k_3\} \subseteq A \setminus \{a,b,c,d,i\}$ and let γ be defined by $\gamma(k_1,k_1) = \gamma(k_2,k_3) = \gamma(k_3,k_2) = i$ and undefined otherwise. Consider the process

$$\overline{P} \stackrel{\text{def}}{=} \tau_{\{i\}} \circ \partial_H((a(b+dk_2)+ck_2)R \parallel S)$$

with

$$\begin{aligned} R &= (k_1(a(b+dk_2)+ck_2))^* k_3 \\ S &= {k_1}^*(k_3k_2). \end{aligned}$$

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

$$\begin{aligned} \overline{P} &= a\cdot\tau_{\{i\}} \circ \partial_H((b+dk_2)R \parallel S) + \\ &\quad c\cdot\tau_{\{i\}} \circ \partial_H(k_2R \parallel S) \\ &= a\cdot\tau_{\{i\}} \circ \partial_H((b+dk_2)R \parallel S) + \\ &\quad c\cdot\tau_{\{i\}} \circ \partial_H(R \parallel k_2) \\ &= a\cdot\tau_{\{i\}} \circ \partial_H((b+dk_2)R \parallel S) + c \end{aligned}$$

$$\begin{aligned} &\tau_{\{i\}} \circ \partial_H((b+dk_2)R \parallel S) \\ &\quad = b\cdot\tau_{\{i\}} \circ \partial_H(R \parallel S) + \\ &\qquad d\cdot\tau_{\{i\}} \circ \partial_H(k_2R \parallel S) \\ &\quad = b\cdot\tau_{\{i\}} \circ \partial_H((a(b+dk_2)+ck_2)R \parallel S) + d \\ &\quad = b\cdot\overline{P} + d. \end{aligned}$$

■

3.2. Hierarchy and expressivity results

We have the following hierarchy results:

THEOREM 3.3 *Let $\prec$ mean "less expressive than". Then* $\mathrm{BPA}^*(A) \prec \begin{matrix}\mathrm{BPA}^*_\delta(A)\\ \mathrm{PA}^*(A)\end{matrix} \prec \mathrm{PA}^*_\delta(A) \prec \mathrm{ACP}^*(A,\gamma) \prec \mathrm{ACP}^*_\tau(A,\gamma)$, *provided A contains at least 6 actions.*

Proof All these inequalities are proved for the associated $^{*\sharp}$-systems in Section 4.3, and these proofs can all be easily restricted to the *-axiom systems. ■

We did not attempt to find lower bounds for the number of actions necessary.

Furthermore, we have the following expressivity result for $\mathrm{ACP}^*_\tau(A,\gamma)$.

THEOREM 3.4 *For each regular process P over A_δ there is a finite 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.*

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.:

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

then becomes

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

In the case that $n = 1$, apparently $P_1 = \beta_1$ (as $\alpha_{1,1} = \delta$), and therefore expressible in $\mathrm{BPA}_\delta(A)$.

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

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

and define γ' on $(A' \setminus A) \times (A' \setminus A)$ by $\gamma'(r_i(b), s_i(b)) = c_i(b)$ and undefined otherwise.

For $i = 1,...,n$ we define the following (regular) processes:

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

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

An intuitive explanation:

$r_i(0) \equiv$ 'read instruction to terminate',
$r_i(1) \equiv$ 'read instruction to (re)activate',
$s_i(0) \equiv$ 'instruct the i_{th} process to terminate',
$s_i(1) \equiv$ 'instruct the i_{th} process to (re)activate'.

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

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

Then the $\overline{P}_i$ solve the recursion equations for the P_i: fix i_0, then

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

because $\parallel$ is commutative and associative as a 'process constructor' (by the SC axioms). 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. In Figure 1, we give a schematic picture of Z_1. In this figure, $s_i(1)\cdot Z_1$ for $i \in \{2,\ldots,n\}$ are proper states if $\alpha_{1,i} \neq \delta$ (and non-existent otherwise) and the boldface arrows to the right represent all steps to $s_i(1)\cdot Z_1$, 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 finite extension of A' with (fresh) actions

$$I_1 \cup H_1 \text{ with } I_1 \overset{\text{def}}{=} \{i\}\ ,\ H_1 \overset{\text{def}}{=} \{k_1,k_2,k_3\}$$

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

$$\begin{aligned}
\overline{\overline{P}}_1 &= \tau_{I_1} \circ \partial_{H_1}((r_1(1)\cdot(\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \\
&\quad \beta_1\cdot R\cdot k_2) + r_1(0)\cdot k_2)\cdot Q \parallel S) \\
Q &= (k_1\cdot r_1(1)\cdot(\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \\
&\quad \beta_1\cdot R\cdot k_2))^* k_3 \\
R &= \|_{j\in\{2,\ldots,n\}} s_j(0) \\
S &= {k_1}^*(k_3\cdot k_2).
\end{aligned}$$

Let γ be defined by $\gamma(k_1,k_1) = \gamma(k_2,k_3) = i$ and further undefined on $(A''\setminus A')\times(A''\setminus A')$.

With RSP one can easily show in $\mathrm{ACP}_\tau^*(A'',\gamma'')$ that $\overline{\overline{P}}_1 = Z_1$ (cf. Lemma 3.2):

$$\begin{aligned}
\overline{\overline{P}}_1 &= r_1(1)\cdot\tau_{I_1}\circ\partial_{H_1}((\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \\
&\quad \beta_1\cdot R\cdot k_2)\cdot Q \parallel S) + \\
&\quad r_1(0)\cdot\tau_{I_1}\circ\partial_{H_1}(k_2\cdot Q \parallel S) \\
&= r_1(1)\cdot\tau_{I_1}\circ\partial_{H_1}((\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \\
&\quad \beta_1\cdot R\cdot k_2)\cdot Q \parallel S) + \\
&\quad r_1(0)\cdot\tau_{I_1}\circ\partial_{H_1}(Q \parallel k_2) \\
&= r_1(1)\cdot\tau_{I_1}\circ\partial_{H_1}((\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \\
&\quad \beta_1\cdot R\cdot k_2)\cdot Q \parallel S) + r_1(0)
\end{aligned}$$

$$\begin{aligned}
&\tau_{I_1}\circ\partial_{H_1}((\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \beta_1\cdot R\cdot k_2)\cdot Q \parallel S) \\
&= \textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1))\cdot\tau_{I_1}\circ\partial_{H_1}(Q \parallel S) + \\
&\quad \beta_1\cdot R\cdot\tau_{I_1}\circ\partial_{H_1}(k_2\cdot Q \parallel S) \\
&= \textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1))\cdot\tau_{I_1}\circ\partial_{H_1}((r_1(1) \\
&\quad (\textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)) + \beta_1\cdot R\cdot k_2) + \\
&\quad r_1(0)\cdot k_2)\cdot Q \parallel S) + \beta_1\cdot R \\
&= \textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1))\cdot\overline{\overline{P}}_1 + \beta_1\cdot R \\
&= \textstyle\sum_{j=2}^n(\alpha_{1,j}\cdot s_j(1)\cdot\overline{\overline{P}}_1) + \beta_1\cdot R.
\end{aligned}$$

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

3.3. Fairness

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 τ-bisimulation, a general solution is provided by *Koomen's Fair Abstraction Rule* (KFAR), introduced in [6]. For each n and each 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 the 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'

(BKS5) $\quad \tau_I(x^*y) = \tau_I(x)^*\tau_I(y)$

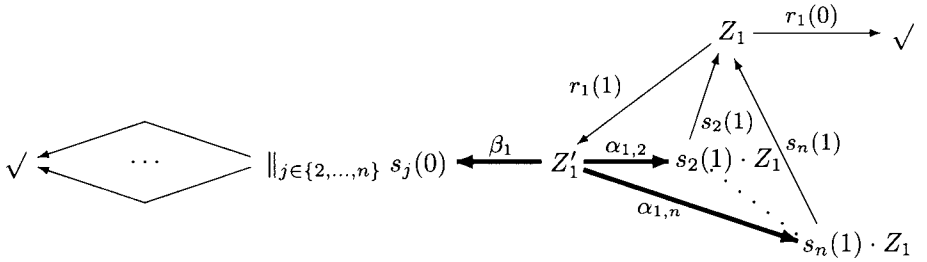


FIGURE 1. A schematic picture of Z_1

(see Table 7), we can even represent KFAR_1 simply by

$$(\mathrm{FIR}_1) \qquad \tau^* x = \tau \cdot x$$

(taking x for $\tau_I(y)$), where FIR abbreviates Fair Iteration Rule.

4. ITERATION AND NESTING

In Section 1 we introduced the nesting operation NO, notation $^\sharp$, with defining axiom

$$x^\sharp y = x(x^\sharp y)x + y.$$

In this section we prove the non-interdefinability of * and $^\sharp$ in most of the axiom systems introduced. Then we define extensions with $^\sharp$ of the axiom systems introduced thus far. We come up with several expressivity results, the last one of which concerns the modelling of a Turing machine in $\mathrm{ACP}^*_\tau(A,\gamma)$ with $^\sharp$ and handshaking. As a consequence, the expressivity result that each finitely branching, computable graph can be expressed by a finite, guarded specification in $\mathrm{ACP}_\tau(A,\gamma)$, formulated in [3], can also be expressed in the setting with * and $^\sharp$ by a single term.

4.1. Axioms and transition rules

In Table 9 we introduce the axiom system $\mathrm{BPA}^\sharp(A)$, obtained by adding the NO axioms (NO1) – (NO3) to $\mathrm{BPA}(A)$ (cf. the system $\mathrm{BPA}^*(A)$ given in Table 6).
A typical $\mathrm{BPA}^\sharp(A)$ identity is

$$x^\sharp(x^\sharp y) = x^\sharp y$$

(take $y = x$ in (NO3)).

The systems $\mathrm{BPA}^\sharp_\delta(A) - \mathrm{PA}^\sharp_\delta(A)$ are defined by inclusion of the NO axioms (NO1) – (NO3).

In Table 10 we give some more axioms for NO. The system $\mathrm{ACP}^\sharp(A,\gamma)$ is defined by inclusion of (NO1) – (NO4), and the system $\mathrm{ACP}^\sharp_\tau(A,\gamma)$ by inclusion of the NO axioms (NO1) – (NO5).

Given our description of NO, the transition rules defined in Table 11 are quite obvious.

It can be easily shown that the axioms in Table 9 and Table 10 are valid in bisimulation semantics ((NO7) only in rooted τ-bisimulation semantics), and according to [2], bisimilarity is a congruence with respect to $^\sharp$.

TABLE 9. The axiom system $\mathrm{BPA}^\sharp(A)$

(A1)	$x+y$	$=$	$y+x$
(A2)	$x+(y+z)$	$=$	$(x+y)+z$
(A3)	$x+x$	$=$	x
(A4)	$(x+y)z$	$=$	$xz+yz$
(A5)	$(xy)z$	$=$	$x(yz)$
(NO1)	$x\cdot(x^\sharp y)\cdot x+y$	$=$	$x^\sharp y$
(NO2)	$(x^\sharp y)x$	$=$	$x^\sharp(yx)$
(NO3)	$(x+y)^\sharp(x\cdot((x+y)^\sharp z)\cdot(x+y)+z)$	$=$	$(x+y)^\sharp z$

TABLE 10. Some more axioms for NO

(NO4)	$\partial_H(x^\sharp y)$	$=$	$\partial_H(x)^\sharp\partial_H(y)$
(NO5)	$\tau_I(x^\sharp y)$	$=$	$\tau_I(x)^\sharp\tau_I(y)$
(NO6)	$x^\sharp\delta$	$=$	$x^*\delta$
(NO7)	$\tau^\sharp x$	$=$	$\tau^* x$

TABLE 11. Transition rules for NO, where $a \in A_\tau$

$$\frac{x \xrightarrow{a} x'}{x^\sharp y \xrightarrow{a} x'\cdot((x^\sharp y)\cdot x)} \qquad \frac{x \xrightarrow{a} \surd}{x^\sharp y \xrightarrow{a} (x^\sharp y)\cdot x}$$

$$\frac{y \xrightarrow{a} y'}{x^\sharp y \xrightarrow{a} y'} \qquad \frac{y \xrightarrow{a} \surd}{x^\sharp y \xrightarrow{a} \surd}$$

REMARK 4.1 The axiom (NO2) is one of the identities[§]

$$((x^n)^\sharp y)\cdot x = (x^n)^\sharp (yx) \qquad (n>0).$$

These identities are valid in bisimulation semantics, and for $n > 1$ not derivable in $\mathrm{BPA}^\sharp(A)$. Their validity follows easily from a variant of RSP, say $\mathrm{RSP}^\sharp$:

$$x = yxy + z \implies x = y^\sharp z$$

which is sound in a setting without τ and τ_I operations. Here is a proof of the identities above:

$$\begin{aligned}((x^n)^\sharp y)x &= ((x^n)((x^n)^\sharp y)(x^n)+y)x\\ &= (x^n)((x^n)^\sharp y)(x^{n+1})+yx\\ &= (x^n)(((x^n)^\sharp y)x)(x^n)+yx\\ &\overset{\mathrm{RSP}^\sharp}{=} (x^n)^\sharp(yx).\end{aligned}$$

The validity of NO3 also follows easily with $\mathrm{RSP}^\sharp$.

From the main result in [4] it follows that bisimilarity between $\mathrm{BPA}^\sharp(A)$ processes is decidable (the associated recursive specifications are 'normed'). It is an open question how $\mathrm{BPA}^\sharp(A)$ should be extended in order to axiomatize bisimilarity.

As $^\sharp$ gives rise to essentially non-regular processes, it cannot be defined in any of the *-systems considered so far. Conversely, the process algebra axiom systems $\mathrm{BPA}^\sharp(A)$ up to and including $\mathrm{ACP}^\sharp(A,\gamma)$ lack *-processes. That is, throughout this part of the hierarchy, * and $^\sharp$ remain non-interdefinable.

THEOREM 4.1

1. *The operation* $^\sharp$ *is not definable in* $\mathrm{ACP}^*_\tau(A,\gamma)$, *and*
2. *The operation* * *is not definable in* $\mathrm{ACP}^\sharp(A,\gamma)$.

Proof Clearly, $^\sharp$ is not definable in terms of *. This follows immediately from Lemma 3.1 and the observation that $a^\sharp b$ has infinitely many non-bisimilar states. To prove the second part of the theorem, we shall show that the basic *-process a^*b is not definable in $\mathrm{ACP}^\sharp(A,\gamma)$.

For the purpose of this proof, we shall introduce the following terminology: we shall say that a process

1. P *has an exit* iff for some $a \in A$, a is an exit of P,
2. P *always has an exit* iff every state Q of P has an exit,
3. P *has a 1-exit* iff P has an exit,
4. P *has an* $n+1$*-exit* iff P has a successor Q which has an n-exit, and
5. P *has arbitrary exits* iff for every $n \in \mathbb{N}$, P has an m-exit, for some $m \in \mathbb{N}$ with $n \le m$.

[§]Personal communication with Fokkink.

Observe that a^*b has arbitrary exits and always has an exit. We shall now show by structural induction that such a process does not exist in $\mathrm{ACP}^\sharp(A,\gamma)$.

The base case is clear: atoms do not have arbitrary exits and δ has no exit at all. For the induction step, we have to distinguish several cases. We shall consider $^\sharp$.

Assume that $P^\sharp Q$ has always and arbitrary exits. Then Q must always have an exit. It follows from the induction hypothesis that Q cannot have arbitrary exits. That is, for some $n+1 \in \mathbb{N}$, Q does not have an $n+1+k$-exit for any $k \in \mathbb{N}$. By the assumption, $P^\sharp Q$ must have such an exit. So, $P^\sharp Q$ has a successor R which has an $n+k$-exit. R cannot be a successor of Q, for otherwise Q would have an $n+1+k$-exit. So either $R \equiv P'\cdot((P^\sharp Q)\cdot P)$ for some successor P' of P, or $R \equiv (P^\sharp Q)\cdot P$. However, in both of the cases, R does not have an exit. This contradicts the assumption that $P^\sharp Q$ always has an exit. ■

However, in $\mathrm{ACP}^\sharp_\tau(A,\gamma)$ things are different:

Proposition 4.1 The process a^*b can be defined in $\mathrm{ACP}^\sharp_\tau(A,\gamma)$ with handshaking.

Proof Let $H \overset{\mathrm{def}}{=} \{a_1,a_2,b_1,b_2,k_1,k_2,k_3,k_4\} \subseteq A \setminus \{a,b\}$. Define

$$P_1 \overset{\mathrm{def}}{=} (a_1k_1)^\sharp(b_1k_2)$$
$$P_2 \overset{\mathrm{def}}{=} (k_3(a_2+b_2))^\sharp k_4$$

and γ by

$$\gamma(a_1,a_2)=a,\quad \gamma(k_1,k_3)=i,\quad \gamma(k_1,a_2)=i,$$
$$\gamma(b_1,b_2)=b,\quad \gamma(k_2,k_4)=i,\quad \gamma(a_1,k_3)=i.$$

and undefined otherwise. Then

$$\tau_{\{i\}}\circ\partial_H(P_1 \parallel (a_2+b_2)P_2)$$

is a solution of X_0 defined by the recursive specification

$$X_j = a\cdot X_{j+1}+b \quad (j\in\mathbb{N}),$$

and

$$\tau_{\{i\}}\circ\partial_H(P_1\cdot(a_1k_1)^{j+1} \parallel (a_2+b_2)P_2\cdot(k_3(a_2+b_2))^{j+1})$$

solves the equation for X_{j+1}. This can be seen as follows:

$$\begin{aligned}&\partial_H(P_1 \parallel (a_2+b_2)P_2)\\ &\quad= \gamma(a_1,a_2)\cdot\partial_H(k_1P_1(a_1k_1) \parallel P_2)+\\ &\qquad\gamma(b_1,b_2)\cdot\partial_H(k_2 \parallel P_2)\\ &\quad= a\cdot\gamma(k_1,k_3)\cdot\partial_H(P_1(a_1k_1) \parallel (a_2+b_2)P_2\cdot\\ &\qquad(k_3(a_2+b_2)))+b\cdot\gamma(k_2,k_4)\\ &\quad= a\cdot i\cdot\partial_H(P_1(a_1k_1) \parallel (a_2+b_2)P_2\cdot\\ &\qquad(k_3(a_2+b_2)))+b\cdot i\end{aligned}$$

and

$$\begin{aligned}
&\partial_H(P_1 \cdot (a_1k_1)^{j+1} \parallel (a_2+b_2)P_2 \cdot (k_3(a_2+b_2))^{j+1}) \\
&= \gamma(a_1,a_2) \cdot \partial_H(k_1P_1(a_1k_1)^{j+2} \parallel \\
&\quad P_2 \cdot (k_3(a_2+b_2))^{j+1}) + \\
&\quad \gamma(b_1,b_2) \cdot \partial_H(k_2 \cdot (a_1k_1)^{j+1} \parallel \\
&\quad P_2 \cdot (k_3(a_2+b_2))^{j+1}) \\
&= a \cdot \gamma(k_1,k_3) \cdot \partial_H(P_1(a_1k_1)^{j+2} \parallel \\
&\quad (a_2+b_2)P_2 \cdot (k_3(a_2+b_2))^{j+2}) + \\
&\quad b \cdot \gamma(k_2,k_4) \cdot \\
&\quad \partial_H((a_1k_1)^{j+1} \parallel (k_3(a_2+b_2))^{j+1}) \\
&= a \cdot i \cdot \partial_H(P_1(a_1k_1)^{j+2} \parallel (a_2+b_2)P_2 \cdot \\
&\quad (k_3(a_2+b_2))^{j+2}) + \\
&\quad b \cdot i \cdot (\gamma(a_1,k_3) \cdot \gamma(k_1,a_2))^{j+1} \\
&= a \cdot i \cdot \partial_H(P_1(a_1k_1)^{j+2} \parallel (a_2+b_2)P_2 \cdot \\
&\quad (k_3(a_2+b_2))^{j+2}) + \\
&\quad b \cdot i \cdot (i \cdot i)^{j+1}.
\end{aligned}$$

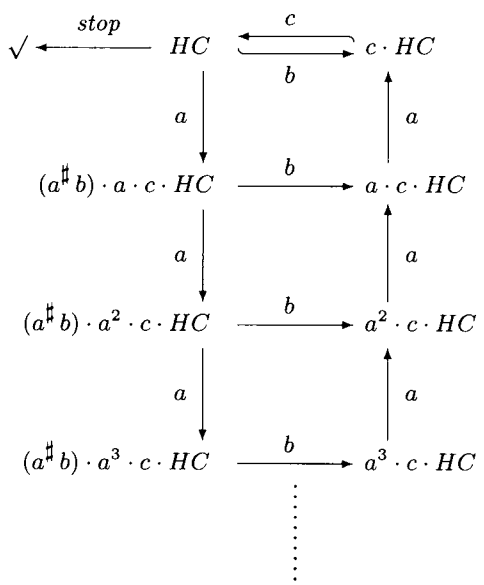
Clearly, a^*b also solves X_j for all $j \in \mathbb{N}$. Hence, by RSP it follows that $\tau_{\{i\}} \circ \partial_H(P_1 \parallel (a_2+b_2)P_2) = a^*b$. ■

We define $\mathrm{BPA}^{*\sharp}(A) - \mathrm{PA}^{*\sharp}_\delta(A)$ by inclusion of (NO1) – (NO3) in the associated * systems. The system $\mathrm{ACP}^{*\sharp}(A,\gamma)$ is defined by inclusion of (NO1) – (NO6) in $\mathrm{ACP}^*(A,\gamma)$, and $\mathrm{ACP}^{*\sharp}_\tau(A,\gamma)$ is defined by inclusion of all the axioms in Table 10 in $\mathrm{ACP}^*_\tau(A,\gamma)$.

An example: the half-counter. We end this section by introducing a basic process that can be specified with $^\sharp$ in $\mathrm{BPA}^{*\sharp}(A)$. The *half-counter* is specified by

$$((a^\sharp b) \cdot c)^* stop$$

for actions a, b, c and *stop*. Let HC abbreviate this expression, then the following transitions can be derived:



The process HC has an 'add mode' characterized by the $(a^\sharp b) \cdot a^n \cdot c \cdot HC$ expressions, and a 'subtract-or-test-zero mode' characterized by the $a^n \cdot c \cdot HC$ expressions above. Change of mode happens by a b step, a c step, or in case of termination in the 'zero state' HC by a *stop* step.

With two half-counters one can specify a *counter* in $\mathrm{ACP}^{*\sharp}_\tau(A,\gamma)+\mathrm{RSP}$, either with or without an option to terminate (i.e., using either *stop*, or δ instead). In the next section we regard a more general case: we use two half-counters to specify a stack over a finite data type.

4.2. The stack

Let $D = \{d_1, ..., d_N\}$ for some $N \in \mathbb{N} \setminus \{0\}$ be a finite set of data elements, ranged over by d. Let furthermore D^* be the set of finite strings over D, ranged over by w, and ϵ denote the empty string. Then the stack S_ϵ over D with termination option is defined by

$$\begin{aligned}
S_\epsilon &= \textstyle\sum_{j=1}^N r(d_j) \cdot S_{d_j} + r(stop) \\
S_{dw} &= \textstyle\sum_{j=1}^N r(d_j) \cdot S_{d_jdw} + s(d) \cdot S_w
\end{aligned}$$

(and a non-terminating stack is obtained by leaving out the $r(stop)$ summand). Here the contents of the stack is represented by the S-index: S_{dw} is the stack that contains dw with d on top. An action $r(d_i)$ (receive d_i) models the push of d_i onto the stack, and an action $s(d_i)$ (send d_i) models a pop of d_i from the stack. The action $r(stop)$ models termination of the (empty) stack. Observe that taking $N = 1$ ($D = \{d_1\}$), the equations above specify a *counter* (the stack contents then models the counter value). We have the following fundamental result for $\mathrm{ACP}^{*\sharp}_\tau(A,\gamma)$.

THEOREM 4.2 *A (terminating) stack over a finite data type D with actions from A can be expressed in $\mathrm{ACP}^{*\sharp}_\tau(B,\gamma)$ with handshaking only, B a finite extension of A and the actions in A not subject to communication.*

Proof We first define the two half-counters that we need in order to specify S_ϵ in $\mathrm{ACP}^{*\sharp}_\tau(B,\gamma)$.

$$P_j = ((\overline{a}_j{}^\sharp \overline{b}_j) \cdot \overline{c}_j)^* \, \overline{stop}_j \quad (j = 1,2),$$

with $\overline{a}_j, \overline{b}_j, \overline{c}_j, \overline{stop}_j \in B \setminus A$. We use the following abbreviations ($n \in \mathbb{N}$):

$$\begin{aligned}
P_j(0) &\stackrel{\text{def}}{=} (\overline{a}_j{}^\sharp \overline{b}_j) \cdot \overline{c}_j \cdot P_j + \overline{stop}_j \\
P_j(n+1) &\stackrel{\text{def}}{=} (\overline{a}_j{}^\sharp \overline{b}_j) \cdot \overline{a}_j^{n+1} \cdot \overline{c}_j \cdot P_j \\
Q_j(0) &\stackrel{\text{def}}{=} \overline{c}_j \cdot P_j \\
Q_j(n+1) &\stackrel{\text{def}}{=} \overline{a}_j^{n+1} \cdot \overline{c}_j \cdot P_j.
\end{aligned}$$

It follows immediately that

1. $P_j = P_j(0) = \overline{a}_j \cdot P_j(1) + \overline{b}_j \cdot Q_j(0) + \overline{stop}_j$,
2. $P_j(n+1) = \overline{a}_j \cdot P_j(n+2) + \overline{b}_j \cdot Q_j(n+1)$,

3. $Q_j(n+1) = \overline{a}_j \cdot Q_j(n)$.

Given $D = \{d_1, ..., d_N\}$ for some $N > 0$, we encode D^* according to the Gödel numbering $\ulcorner\cdot\urcorner : D^* \to \mathbb{N}$ defined by:

$$\begin{aligned} \ulcorner\epsilon\urcorner &\stackrel{\text{def}}{=} 0, \\ \ulcorner d_j w\urcorner &\stackrel{\text{def}}{=} j + N\,\ulcorner w\urcorner. \end{aligned}$$

This coding is a bijection with inverse $decode : \mathbb{N} \to D^*$ defined by:

$$n \bmod N = 0 \Longrightarrow$$
$$decode(n) \stackrel{\text{def}}{=} \begin{cases} \epsilon & \text{if } n = 0, \\ d_N \star decode(\frac{n-N}{N}) & \text{otherwise,} \end{cases}$$

$$n \bmod N > 0 \Longrightarrow$$
$$decode(n) \stackrel{\text{def}}{=} d_{n \bmod N} \star decode(\tfrac{n-(n \bmod N)}{N})$$

where $\star$ denotes concatenation of strings.

So in case $N = 3$, e.g., $\ulcorner d_3 d_1 d_2\urcorner = 24$, and $decode(32) = d_2 d_1 d_3 \in \{d_1, d_2, d_3\}^*$.

Next we define the auxiliary processes with actions $a_j, b_j, c_j, stop_j \in B \setminus A$ that in combination with the P_j can be used to define S_ϵ:

$$X_\emptyset = \sum_{j=1}^{N} r(d_j) \cdot a_1^j \cdot b_1 \cdot X_j + r(stop) \cdot stop_1 \cdot stop_2$$

with for $k = 1, ..., N$ and $N > 1$:

$$\begin{aligned} X_k &= \textstyle\sum_{j=1}^{N} r(d_j) \cdot \mathrm{Push}_j + s(d_k) \cdot \mathrm{Pop}_k \\ \mathrm{Push}_k &= \mathrm{Shift\,1\,to\,2} \cdot a_1^k \cdot N\,\mathrm{Shift\,2\,to\,1} \cdot X_k \\ \mathrm{Pop}_k &= a_1^k \cdot \tfrac{1}{N}\,\mathrm{Shift\,1\,to\,2} \cdot \mathrm{Test}_\emptyset \end{aligned}$$

$$\begin{aligned} \mathrm{Shift\,1\,to\,2} &= (a_1 \cdot a_2)^* c_1 \cdot b_2 \\ N\,\mathrm{Shift\,2\,to\,1} &= (a_2 \cdot a_1^N)^* c_2 \cdot b_1 \\ \tfrac{1}{N}\,\mathrm{Shift\,1\,to\,2} &= (a_1^N \cdot a_2)^* c_1 \cdot b_2 \end{aligned}$$

$$\begin{aligned} \mathrm{Test}_\emptyset &= a_2 \cdot a_1 \cdot \mathrm{Test}_1 + c_2 \cdot X_\emptyset \\ \mathrm{Test}_1 &= a_2 \cdot a_1 \cdot \mathrm{Test}_2 + c_2 \cdot b_1 \cdot X_1 \\ \mathrm{Test}_2 &= a_2 \cdot a_1 \cdot \mathrm{Test}_3 + c_2 \cdot b_1 \cdot X_2 \\ &\vdots \\ \mathrm{Test}_N &= a_2 \cdot a_1 \cdot \mathrm{Test}_1 + c_2 \cdot b_1 \cdot X_N \end{aligned}$$

or in case $N = 1$, as above, except for

$$\mathrm{Test}_1 = a_2 \cdot a_1 \cdot \mathrm{Test}_1 + c_2 \cdot b_1 \cdot X_1.$$

Let γ for $j = 1, 2$ be defined on $(B \setminus A)^2$ such that

$$\gamma(a_j, \overline{a}_j) = \gamma(b_j, \overline{b}_j) = \gamma(c_j, \overline{c}_j) =$$
$$\gamma(stop_j, \overline{stop}_j) =$$
$$i \in B \setminus A.$$

Taking

$$H = \{a_j, \overline{a}_j, b_j, \overline{b}_j, c_j, \overline{c}_j, stop_j, \overline{stop}_j \mid j = 1, 2\},$$
$$I = \{i\},$$

we show that $\tau_I \circ \partial_H(X_\emptyset \parallel P_1(0) \parallel P_2(0))$ behaves as the stack S_ϵ.

$$\begin{aligned} &\tau_I \circ \partial_H(X_\emptyset \parallel P_1(0) \parallel P_2(0)) \\ &= \textstyle\sum_{j=1}^{N} r(d_j) \cdot \tau_I \circ \partial_H(a_1^j \cdot b_1 \cdot X_j \parallel P_1(0) \parallel P_2(0)) + \\ &\quad r(stop) \cdot \tau_I \circ \partial_H(stop_1 \cdot stop_2 \parallel P_1(0) \parallel P_2(0)) \\ &= \textstyle\sum_{j=1}^{N} r(d_j) \cdot \tau^j \cdot \tau_I \circ \partial_H(b_1 \cdot X_j \parallel P_1(j) \parallel P_2(0)) + \\ &\quad r(stop) \cdot \tau \cdot \tau_I \circ (stop_2 \parallel P_2(0)) \\ &= \textstyle\sum_{j=1}^{N} r(j) \cdot \tau^{j+1} \cdot \tau_I \circ \partial_H(X_j \parallel Q_1(j) \parallel P_2(0)) + \\ &\quad r(stop) \cdot \tau \cdot \tau \\ &\stackrel{(1)}{=} \textstyle\sum_{j=1}^{N} r(d_j) \cdot \tau_I \circ \partial_H(X_j \parallel Q_1(\ulcorner d_j\urcorner) \parallel P_2(0)) + \\ &\quad r(stop). \end{aligned}$$

We are done if $\tau_I \circ \partial_H(X_j \parallel Q_1(\ulcorner d_j w\urcorner) \parallel P_2(0))$ behaves as $S_{d_j w}$, the stack with contents $d_j w$ for some $w \in D^*$. We prove this by first analyzing the behaviour of $\partial_H(X_j \parallel Q_1(\ulcorner d_j w\urcorner) \parallel P_2(0))$. This analysis is arranged in a graphical style in Figure 2, where $P \xrightarrow{a} Q$ represents the statement $P = a \cdot Q$ for some $a \in A$, $P \xrightarrow{w} Q$ represents $P = w \cdot Q$ and branching from expressions represents application of $+$. So the uppermost expression in the picture underneath with its arrows and resulting expressions represents the equation

$$\begin{aligned} &\partial_H(X_j \parallel Q_1(\ulcorner d_j w\urcorner) \parallel P_2(0)) = \\ &\quad s(d_j) \cdot \partial_H(\mathrm{Pop}_j \parallel Q_1(j + N\,\ulcorner w\urcorner) \parallel P_2(0)) + \\ &\quad \textstyle\sum_{k=1}^{N} r(d_k) \cdot \partial_H(\mathrm{Push}_k \parallel Q_1(\ulcorner d_j w\urcorner) \parallel P_2(0)) \end{aligned}$$

which is obviously derivable. With the axiom $x = x \cdot \tau$ (T1) and identity (1) above, it can be seen from Figure 2 that

$$\tau_I \circ \partial_H(X_\emptyset \parallel P_1(0) \parallel P_2(0))$$
$$\tau_I \circ \partial_H(X_j \parallel Q_1(\ulcorner d_j w\urcorner) \parallel P_2(0))$$

satisfy the equations for S_ϵ and $S_{d_j w}$, respectively ($j = 1, ..., N$ and $w \in D^*$). By RSP it follows that

$$\begin{aligned} &S_\epsilon = \\ &\tau_I \circ \partial_H(X_\emptyset \parallel ((\overline{a}_1 \,|\!\!\!\cdot\, \overline{b}_1) \cdot \overline{c}_1)^* \overline{stop}_1 \parallel ((\overline{a}_2 \,|\!\!\!\cdot\, \overline{b}_2) \cdot \overline{c}_2)^* \overline{stop}_2) \end{aligned}$$

where the abbreviations $P_j(0)$ are written out.

Finally, we claim that $X_\emptyset$ and all other process identifiers in its definition represent regular processes, and can hence be expressed in $\mathrm{ACP}_\tau^{*\sharp}(B, \gamma)$ with handshaking, provided $B \supseteq A$ is sufficiently large (depending on N, the size of D). This follows from Theorem 3.4. We refrain from expressing $X_\emptyset$ as a *-expression (parameterized with N).

Hence, the empty stack S_ϵ can be expressed in $\mathrm{ACP}_\tau^{*\sharp}(B, \gamma)$ with handshaking. ■

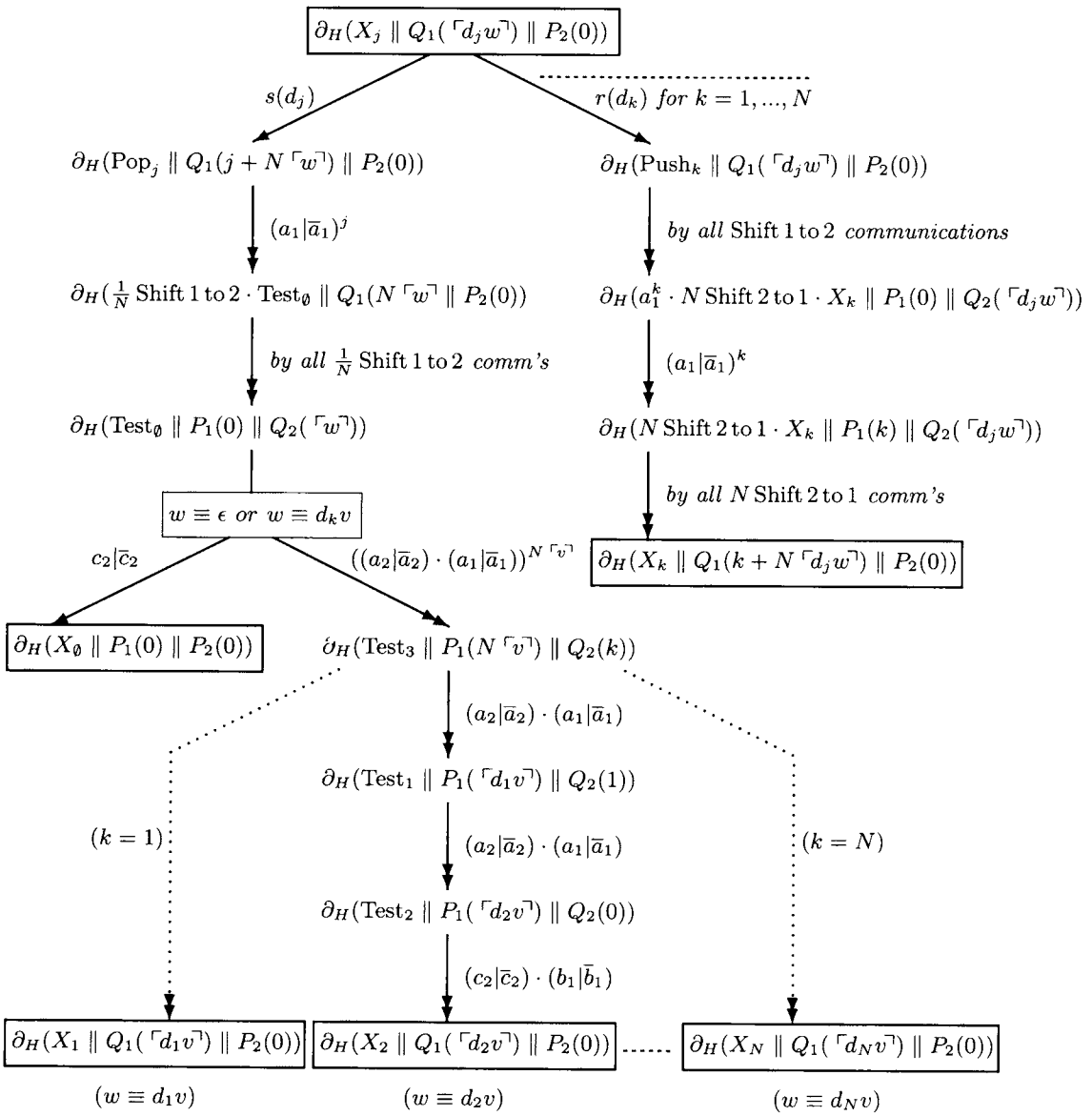


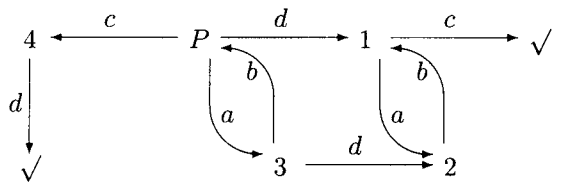
FIGURE 2. Calculations with $\partial_H(X_j \parallel Q_1(\ulcorner d_j w \urcorner) \parallel P_2(0))$

4.3. Hierarchy and expressivity results

All results on the *-hierarchy mentioned in Section 3.2 carry over to the associated $^{*\sharp}$-systems. A similar hierarchy result can be obtained for $^\sharp$-systems.

THEOREM 4.3 $\mathrm{BPA}^{*\sharp}(A) \prec \mathrm{PA}^{*\sharp}(A)$, *and* $\mathrm{BPA}_\delta^{*\sharp}(A) \prec \mathrm{PA}_\delta^{*\sharp}(A)$.

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



with 1 abbreviating $(ab)^*c$
2 abbreviating $b((ab)^*c)$
3 abbreviating $b((ab)^*c) \parallel d$
4 abbreviating d.

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

$Q \in A_\delta$ or $Q \equiv Q_1 + Q_2$. Then Q is not 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 $Q_2 \leftrightarrows 1 \leftrightarrows 4$. But obviously $1 \not\leftrightarrows 4$.

$Q \equiv Q_1{}^*Q_2$. 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 Q. (Here a proof for the related *-systems would end, see Theorem 3.3.)

$Q \equiv Q_1{}^\sharp Q_2$. As the case above. ■

THEOREM 4.4 $\mathrm{PA}_\delta^{*\sharp}(A) \prec \mathrm{ACP}^{*\sharp}(A,\gamma)$.

Proof In a similar way as was done in the proof above, one can show that $(ab)^*c \parallel d$ with $\gamma(c,d)$ the only communication defined, cannot be expressed in $\mathrm{PA}_\delta^{*\sharp}(A)$. Moreover, also $\mathrm{PA}_\delta^*(A) \prec \mathrm{ACP}^*(A,\gamma)$ can be proved using the process mentioned. ■

THEOREM 4.5 *If A contains at least 6 actions, one can find γ with handshaking such that* $\mathrm{ACP}^{*\sharp}(A,\gamma) \prec \mathrm{ACP}_\tau^{*\sharp}(A,\gamma)$.

Proof Consider the following property of *root-cyclic* states over $\mathrm{ACP}^{*\sharp}(A,\gamma)$:

EXIT PROPERTY Let R be a root-cyclic state, and C a cycle containing R. If two states in C have exits, then they have the *same* exits.

The exit property holds for $\mathrm{ACP}^{*\sharp}(A,\gamma)$. This can be proved by induction on the syntactic structure of root-cyclic states.

As a consequence it follows that the regular process P defined by $P = a\cdot Q + a$ and $Q = a\cdot P + b$ cannot be specified in $\mathrm{ACP}^{*\sharp}(A,\gamma)$. However, adapting Lemma 3.2 to this process, it follows that P can be specified in $\mathrm{ACP}_\tau^*(A,\gamma)$ provided $A \supseteq \{a,b,i,k_1,k_2,k_3\}$, and hence also in $\mathrm{ACP}_\tau^{*\sharp}(A,\gamma)$. ■

So we have the following hierarchy (for A containing at least 6 actions):

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

Finally, we have the following expressivity result for $\mathrm{ACP}_\tau^{*\sharp}(A,\gamma)$:

THEOREM 4.6 *For each finitely branching, computable graph with labels from A there is a finite extension B of A such that this graph can be expressed in* $\mathrm{ACP}_\tau^{*\sharp}(B,\gamma)$ *with handshaking only, and the actions in A not subject to communication.*

In this paper we do not want to introduce the graph model for $\mathrm{ACP}_\tau(A,\gamma)$. For a detailed introduction, see [1]. Theorem 4.6 follows immediately from one of the main results in [3]. The idea is that a computable (finitely branching) graph can be characterized by a rooted τ-bisimilar one that apart from its root has out-degree of at most two. Such a graph can be defined with help of two computable functions. Each computable function can in turn be represented in $\mathrm{ACP}_\tau^{*\sharp}(A,\gamma)$ with handshaking. This representation is based on the modelling of a Turing machine with help of two terminating stacks and a regular control process (using FIR_1, see Section 3.3). We have seen that such processes can be defined in $\mathrm{ACP}_\tau^{*\sharp}(A,\gamma)$ with handshaking.

5. CONCLUSIONS

The results described in this paper form a further step in the equational founding of process algebra. We can focus on a small number of recursion operations such as * and $^\sharp$. Using such operations, processes can simply be represented by closed terms.

As pointed out by a referee, the results stated in this paper go through in the setting with *branching bisimulation*, introduced in [14, 15].

ACKNOWLEDGEMENTS

We thank Wan Fokkink, Jos van Wamel and Chris Verhoef for some clarifying discussions. We also thank a referee for useful suggestions. J. A. Bergstra was supported in part by ESPRIT basic research action 6454, CONFER.

REFERENCES

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

[2] J. C. M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, ed., *Proceedings CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pp. 477 – 492. Springer-Verlag, Berlin, 1993.

[3] J. C. M. Baeten, J. A. Bergstra and J. W. Klop. On the consistency of Koomen's fair abstraction rule. *Theoretical Computer Science*, 51(1/2):129–176, 1987.

[4] J. C. M. Baeten, J. A. Bergstra and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM*, 40(3):653–682, 1993.

[5] 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, eds, *Math. Methods of Spec. and Synthesis of Software Systems '85, Math. Research 31*, pp. 9–23, Berlin, 1986. Akademie-Verlag. First appeared as: Report CS-R8404, CWI, Amsterdam, 1984.

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

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

[8] J. A. Bergstra and J. W. Klop. Process algebra: specification and verification in bisimulation semantics. In M. Hazewinkel, J. K. Lenstra and L. G. L. T. Meertens, eds, *Mathematics and Computer Science II*, CWI Monograph 4, pp. 61–94. North-Holland, Amsterdam, 1986.

[9] 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.

[10] J. A. Bergstra, I. Bethke and A. Ponse. Process algebra with iteration. Report P9314, Programming Research Group, University of Amsterdam, 1993.

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

[12] W. J. Fokkink and H. Zantema. Basic process algebra with iteration: completeness of its equational axioms. This issue.

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

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

[15] R. J. van Glabbeek and W. P. Weijland. Refinement in branching time semantics. Report CS-R8922, CWI, Amsterdam, 1989. Also appeared in: *Proceedings AMAST Conference*, May 1989, Iowa, USA, pp. 197–201.

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

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

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

[19] A. Ponse. Process algebra and dynamic logic. In J. van Eijck and A. Visser, eds, *Logic and Information Flow*, MIT Press, Cambridge, MA, 1994.

[20] 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.