# Merge and Termination in Process Algebra

J.C.M. Baeten,
*Dept. of Computer Science, University of Amsterdam,*
*P.O. Box 41882, 1009 DB Amsterdam, The Netherlands*

R.J. van Glabbeek,
*Dept. of Software Technology, Centre for Mathematics and Computer Science,*
*P.O. Box 4079, 1009 AB Amsterdam, The Netherlands*

**Abstract:** In VRANCKEN [14], the empty process ε was added to the Algebra of Communicating Processes of BERGSTRA & KLOP [3, 4]. Reconsidering the definition of the parallel composition operator merge, we found that it is preferable to explicitly state the termination option. This gives an extra summand in the defining equation of merge, using the auxiliary operator √ (tick). We find that tick can be defined in terms of the encapsulation operator $\partial_H$. We give an operational and a denotational semantics for the resulting system ACP√, and prove that they are equal. We consider the Limit Rule, and prove it holds in our models.
**Note:** Partial support received from the European Communities under ESPRIT contract no. 432, An Integrated Formal Approach to Industrial Software Development (Meteor).

## 1. INTRODUCTION

Having been introduced to the Algebra of Communicating Processes of BERGSTRA & KLOP [3, 4], many people ask the question why there is no neutral element for the sequential composition ·. The neutral element for alternative composition + is the constant δ, that is used to denote deadlock, unsuccessful termination. A constant ε satisfying the laws $\varepsilon\cdot x = x\cdot\varepsilon = x$ must stand for an **empty process**, a process that terminates immediately and successfully. The investigation of what happens when we want to add such a constant to ACP was started by KOYMANS & VRANCKEN [9]. It turned out that the constant ε is very useful, but that the technicalities involved were substantial. For instance, the just quoted paper contained a non-associative merge operator. This problem was remedied in VRANCKEN [14], where the theory ACP was modified and extended to $ACP^{\varepsilon}$. In practice, the constant ε already showed its usefulness in BERGSTRA, KLOP & OLDEROG [5], where ε was needed to define the constant Δ denoting divergence.

This paper was motivated by a reconsideration of the interaction of merge and empty process in the papers [9], [14]. Merge is the parallel composition operator ||. Not considering communication for the moment, the merge of processes x and y will interleave the actions of x and y. In x||y, there are three possibilities: a step from x can be executed, or a step from y, or the process can terminate (only if both x and y have that option). These options are present in the defining axiom of merge:

$$x\|y = x\mathbin{\lfloor\!\lfloor} y + y\mathbin{\lfloor\!\lfloor} x + \surd(x)\cdot\surd(y).$$

Here, we use the auxiliary operators $\lfloor\!\lfloor$ (left-merge) and √ (used to indicate termination). Now, in [9] and [14], the left-merge is used also to indicate the termination possibility. We think that a separation of the two notions makes a more refined treatment possible, and can lead to a better understanding of the issues involved.

In section 2, we first discuss termination in a setting without communication, using the free merge. In section 3, we add communication, and prove some theorems, such as the elimination theorem and the expansion theorem. We also briefly discuss infinite processes. In section 4, we discuss different semantics for our theory, namely a term model and two graph models. The term model is based on action relations, is operational (cf. MILNER [11], PLOTKIN [13]), while the second graph model is denotational. We prove that these models are isomorphic.

In these models, guarded recursive specifications have unique solutions. We prove that these models are complete for our theory w.r.t. closed terms, i.e. we have a complete axiomatisation for them. A short proof of this fact was not published before, even for the theory ACP without empty process.

Finally, in section 5, we consider the Limit Rule of BAETEN & BERGSTRA [1], and prove that a restricted version holds in our models. The Limit Rule says that if we have an equation that holds for all finite processes, then it holds for all processes.

ACKNOWLEDGEMENT. The authors express their gratitude to Jan Bergstra and Henk Goeman, for help in developing the concepts defined in this paper.

## 2. PROCESS ALGEBRA WITH FREE MERGE

In this section, we consider the case of merge without communication, the so-called *free merge*. Our starting point is the theory PA as defined in BERGSTRA & KLOP [2], without empty process $\varepsilon$. For other algebraical theories of concurrency, see e.g. MILNER [10] or HOARE [8].

2.1 Process algebra starts from a collection of given objects, called atomic actions, atoms or steps. These actions are taken to be indivisible, usually have no duration and form the basic building blocks of our systems. The first two compositional operators we consider are $\cdot$, denoting sequential composition, and + for alternative composition. If $x$ and $y$ are two processes, then $x\cdot y$ is the process that starts the execution of $y$ after the completion of $x$, and $x+y$ is the process that chooses either $x$ or $y$ and executes the chosen process (not the other one). Each time a choice is made, we choose from a *set* of alternatives (see axioms A1-3 below). We do not specify whether a choice is made by the process itself, or by the environment. We leave out $\cdot$ and brackets as in regular algebra, so $xy + z$ means $(x\cdot y) + z$. $\cdot$ will always bind stronger than other operators, and + will always bind weaker.
On intuitive grounds $x(y + z)$ and $xy + xz$ present different mechanisms (the moment of choice is different), and therefore, an axiom $x(y + z) = xy + xz$ is not included.
Next, we have the parallel composition operator $\|$, called merge. The merge of processes $x$ and $y$ will interleave the actions of $x$ and $y$. In $x\|y$, either a step from $x$ can be executed, or a step from $y$. These options are present in axiom M1. Here, we use the auxiliary operator $\lfloor\!\lfloor$ (left-merge). Thus, $x\lfloor\!\lfloor y$ is $x\|y$, but with the restriction that the first step comes from $x$, and likewise for $y\lfloor\!\lfloor x$. Axioms M2-4 give the laws for $\lfloor\!\lfloor$.

2.2 DEADLOCK. We enlarge the signature of PA, by adding the special constant $\delta$, denoting deadlock, the acknowledgement of a process that it cannot do anything anymore, the absence of any alternative (see BERGSTRA & KLOP [3, 4]). $\delta$ has axioms A6-7. A process that ends in $\delta$ terminates unsuccessfully. The theory PA plus $\delta$ is called $PA_\delta$.

2.3 SIGNATURE AND AXIOMS. $A$ is a given (finite) set of atomic actions. All elements of $A$ are constants of $PA_\delta$. Further, $PA_\delta$ has binary operators $+,\cdot,\|,\lfloor\!\lfloor$, and a constant $\delta$ ($\delta \notin A$).
The axioms of $PA_\delta$ are presented in table 1. There $a \in A\cup\{\delta\}$, and $x,y,z$ are arbitrary processes.

| | | | |
|---|---|---|---|
| $x + y = y + x$ | A1 | $x\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x$ | M1 |
| $(x + y) + z = x + (y + z)$ | A2 | $a\lfloor\!\lfloor x = ax$ | M2 |
| $x + x = x$ | A3 | $ax\lfloor\!\lfloor y = a(x\|y)$ | M3 |
| $(x + y)z = xz + yz$ | A4 | $(x + y)\lfloor\!\lfloor z = x\lfloor\!\lfloor z + y\lfloor\!\lfloor z$ | M4 |
| $(xy)z = x(yz)$ | A5 | | |
| $x + \delta = x$ | A6 | | |
| $\delta x = \delta$ | A7 | | |

Table 1. $PA_\delta$.

2.4 EMPTY PROCESS. Now we add the empty process $\varepsilon$, giving us the theory $PA\surd$. $\varepsilon$ is the neutral element of sequential composition, so has axioms $\varepsilon x = x\varepsilon = x$ (A8,9). In a sum, as in $x + \varepsilon$, it tells us

that the process can terminate immediately and successfully. We introduce the operator $\surd$ to indicate whether or not a process can terminate immediately: $\surd(x) = \varepsilon$ if $x$ has the termination option, and $\surd(x) = \delta$ otherwise. Axioms Te1-4 give an axiomatisation of $\surd$: we just rename all atomic actions into $\delta$, and distribute $\surd$ over $+$ and $\cdot$.
Now, in $x\|y$, there are three possibilities: we can start with a step from $x$, or with a step from $y$, or the process can terminate, if both $x$ and $y$ have this option. A simple case distinction learns us that the termination summand of $x\|y$ can be represented by $\surd(x)\cdot\surd(y)$ $(= \surd(y)\cdot\surd(x))$. See axiom MT1. Finally, axioms EM2-4 are the laws for left-merge.

2.5 SIGNATURE AND AXIOMS. We have in the signature of theory PA$\surd$ as constants all elements of $A\cup\{\delta,\varepsilon\}$, the binary operators $+,\cdot,\|,\lfloor\!\lfloor$, and the unary operator $\surd$. The axioms are presented in table 2 below. There $a \in A\cup\{\delta\}$, and $x,y,z$ are arbitrary processes.

| | | | |
|---|---|---|---|
| $x + y = y + x$ | A1 | $\delta + \varepsilon = \varepsilon$ | A6 |
| $(x + y) + z = x + (y + z)$ | A2 | $\delta x = \delta$ | A7 |
| $\varepsilon + \varepsilon = \varepsilon$ | A3 | $\varepsilon x = x$ | A8 |
| $(x + y)z = xz + yz$ | A4 | $x\varepsilon = x$ | A9 |
| $(xy)z = x(yz)$ | A5 | | |
| $x\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x + \surd(x)\cdot\surd(y)$ | MT1 | $\surd(\varepsilon) = \varepsilon$ | Te1 |
| $\varepsilon\lfloor\!\lfloor x = \delta$ | EM2 | $\surd(a) = \delta$ | Te2 |
| $ax\lfloor\!\lfloor y = a(x\|y)$ | EM3 | $\surd(x + y) = \surd(x) + \surd(y)$ | Te3 |
| $(x + y)\lfloor\!\lfloor z = x\lfloor\!\lfloor z + y\lfloor\!\lfloor z$ | EM4 | $\surd(xy) = \surd(x)\cdot\surd(y)$ | Te4 |

Table 2. PA$\surd$.

2.6 REMARKS. In PA$\surd$, we have different versions of the axioms A3 and A6 of $PA_\delta$. Using axioms A4,7,8, it can be seen that the new versions are equivalent to the old ones. As we will see in section 3, the axiom M2 of $PA_\delta$ is derivable from PA$\surd$ if we add the extra axiom $\varepsilon\|x = x$. It is debatable whether or not this axiom $\varepsilon\|x = x$ should be included in PA$\surd$. We have chosen not to, since it is derivable for all closed terms (see section 3). PA$\surd$ differs from $PA^\varepsilon$ of VRANCKEN [14], by the use of the termination operator $\surd$. In [14], the termination option is represented by $\varepsilon\lfloor\!\lfloor x$.
Originally, we considered a binary operator $\downarrow$ instead of the unary operator $\surd$. $\downarrow$ is the so-called "termination merge", and $x\downarrow y = \varepsilon$ iff both $x$ and $y$ have the termination option, and $\delta$ otherwise. In this case, the axiom MT1 would read $x\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x + x\downarrow y$. The operator $\downarrow$ can be axiomatised by the following laws:

$$x\downarrow y = y\downarrow x,\quad \varepsilon\downarrow\varepsilon = \varepsilon,\quad ax\downarrow y = \delta,\quad (x + y)\downarrow z = x\downarrow z + y\downarrow z.$$

The idea to use a unary operator came from Jan Bergstra.
Before we discuss some consequences of PA$\surd$, we first introduce communication in section 3. Results for the system PA$\surd$, so without communication, can be obtained from the results in section 3 by forgetting the communication function.

## 3. ALGEBRA OF COMMUNICATING PROCESSES

We introduce the communication of the system ACP (Algebra of Communicating Processes) of BERGSTRA & KLOP [3, 4]. If two processes simultaneously execute two atomic actions that can communicate, the result is a communication action. In $x\|y$, we will add a fourth summand: the execution of a communication action, with components from $x$ and $y$. Below we define the system ACP$\surd$.

3.1 SIGNATURE AND AXIOMS. A is a given (finite) set of atomic actions. On A, we assume that a **communication function** $\gamma$ is given: $\gamma$ is a partial binary function, that is commutative and associative, i.e. for all $a,b,c \in A$:

$$\gamma(a,b) = \gamma(b,a)$$
$$\gamma(\gamma(a,b),c) = \gamma(a,\gamma(b,c)),$$

(and each side of these equations is defined just when the other side is). If $\gamma(a,b)$ is defined (we write $\gamma(a,b)\downarrow$), and $\gamma(a,b) = c$, it means that actions $a$ and $b$ can communicate, and their communication is $c$; if $\gamma(a,b)$ is not defined, we say that $a$ and $b$ do not communicate.

All elements of A are constants of ACP$\surd$. Further, ACP$\surd$ has binary operators $+,\cdot,\|,\lfloor\!\lfloor,|$, unary operators $\partial_H$, $\varepsilon_K$ (for $H,K \subseteq A$) and constants $\delta,\varepsilon$.

The axioms of ACP$\surd$ are listed in table 3 below.

There, $a \in A\cup\{\delta\}$, $H,K \subseteq A$ and $x,y,z$ are arbitrary processes.

| | | | |
|---|---|---|---|
| $x + y = y + x$ | A1 | $\delta + \varepsilon = \varepsilon$ | A6 |
| $(x + y) + z = x + (y + z)$ | A2 | $\delta x = \delta$ | A7 |
| $\varepsilon + \varepsilon = \varepsilon$ | A3 | $\varepsilon x = x$ | A8 |
| $(x + y)z = xz + yz$ | A4 | $x\varepsilon = x$ | A9 |
| $(xy)z = x(yz)$ | A5 | | |
| | | $a \mid b = \gamma(a,b)$ if $\gamma(a,b)\downarrow$ | CF1 |
| | | $a \mid b = \delta$ otherwise | CF2 |
| $x\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x + x \mid y + \surd(x)\cdot\surd(y)$ | EM1 | $x \mid y = y \mid x$ | EM5 |
| $\varepsilon\lfloor\!\lfloor x = \delta$ | EM2 | $x \mid \varepsilon = \delta$ | EM6 |
| $ax\lfloor\!\lfloor y = a(x\|y)$ | EM3 | $x \mid ay = (x \mid a)\lfloor\!\lfloor y$ | EM7 |
| $(x + y)\lfloor\!\lfloor z = x\lfloor\!\lfloor z + y\lfloor\!\lfloor z$ | EM4 | $x \mid (y + z) = x \mid y + x \mid z$ | EM8 |
| | | | |
| $\partial_H(\varepsilon) = \varepsilon$ | D0 | $\varepsilon_K(\varepsilon) = \varepsilon$ | E0 |
| $\partial_H(a) = a$ if $a \notin H$ | D1 | $\varepsilon_K(a) = a$ if $a \notin K$ | E1 |
| $\partial_H(a) = \delta$ if $a \in H$ | D2 | $\varepsilon_K(a) = \varepsilon$ if $a \in K$ | E2 |
| $\partial_H(x + y) = \partial_H(x) + \partial_H(y)$ | D3 | $\varepsilon_K(x + y) = \varepsilon_K(x) + \varepsilon_K(y)$ | E3 |
| $\partial_H(xy) = \partial_H(x)\cdot\partial_H(y)$ | D4 | $\varepsilon_K(xy) = \varepsilon_K(x)\cdot\varepsilon_K(y)$ | E4 |

Table 3. ACP$\surd$.

$|$ is the **communication merge**: $x \mid y$ is just like $x\|y$, but with the restriction that the first step must be a communication action, with components from $x$ and $y$. Now if $\gamma(a,b) = c$, we can calculate that $a\|b = ab + ba + c$; if we do not want the $a,b$ to occur separately, but only in the communication, we have to encapsulate them, using the **encapsulation operator** $\partial_H$: if $H = \{a,b\}$, we get $\partial_H(a\|b) = c$. $\partial_H$ blocks all atomic actions from $H \subseteq A$, by renaming them into $\delta$. Lastly, $\varepsilon_K$ is also a renaming operator, that erases all actions from $K \subseteq A$, by renaming them into $\varepsilon$.

We did not list $\surd$ as an operator of ACP$\surd$, because it has become definable: $\surd$ is just $\partial_A$, the operator that renames *all* atomic actions into $\delta$. This fact was pointed out to us by Henk Goeman. We will still use the notation $\surd$, though.

3.2 REMARKS. Axioms Te1-4 of PA$\surd$ are just the axioms D0,2,3,4 for the operator $\partial_A$. ACP$\surd$ differs from ACP$^\varepsilon$ of VRANCKEN [14], by the use of the termination operator. Other differences are that in [14], $\gamma$ is a total function from $A\times A$ to $A\cup\{\delta\}$, while in this paper, $\gamma$ is a partial function from $A\times A$ to $A$. Also, in the axioms in [14], $a$ varies over $A$, not over $A\cup\{\delta\}$, which necessitates more axioms. Lastly, we left out the axiom $(x \mid y) \mid z = x \mid (y \mid z)$, as we saw no reason for its inclusion (it will be an axiom of Standard Concurrency, see below).

The system $ACP^{\varepsilon}$ itself differs in several aspects from the system ACP of BERGSTRA & KLOP [3]. Most of these differences were a consequence of the addition of the constant $\varepsilon$. Another difference is the inclusion of axiom EM5, the commutativity of the communication merge, which decreased the number of axioms needed.

3.3 LEMMA. The following equations are derivable from the system $ACP\surd$ ($a,b \in A\cup\{\delta\}$):
1. $ax\|by = by\|ax$ 2. $x\,|\,\delta = \delta$
PROOF: Straightforward. For 2, use EM6,8 and A6.

3.4 LEMMA. In the system $ACP\surd$ plus extra axiom $\varepsilon\|x = x$ the following equations are derivable ($a,b\in A\cup\{\delta\}$):
1. $x = x\lfloor\varepsilon + \surd(x)$ 2. $x\|\varepsilon = x$ 3. $a\lfloor x = ax$
4. $a\,|\,bx = ax\,|\,b = (a\,|\,b)x$ 5. $ax\,|\,by = (a\,|\,b)(x\|y)$
PROOF: Straightforward.

3.5 Note that equation 3.4.1 states that we can write each process as the sum of its termination option ($\surd(x)$) and the summands that start with an atomic action ($x\lfloor\varepsilon$). Equations 3.4.3-5 are axioms of ACP. In the next lemma, we focus on another equation that is of special interest, namely the assertion that $\surd(x)$ must be either $\varepsilon$ or $\delta$ (note that $x = x + \varepsilon$ amounts to saying that $x$ has an $\varepsilon$-summand):

$\surd(x) = \varepsilon$ iff $x = x + \varepsilon$, and $\surd(x) = \delta$ otherwise (Te5).

3.6 LEMMA. In the system $ACP\surd$ plus extra axiom Te5 the following equations are derivable:
1. $x\|y = y\|x$ 2. $\surd(x)\lfloor y = \delta$ 3. $\surd(x)\,|\,y = \delta$
PROOF: Straightforward.

3.7 DEFINITION. A **basic term** is a closed term of the form

$$t = a_0t_0 + \ldots + a_{n-1}t_{n-1} + b_0 + \ldots + b_{m-1}\ (+\varepsilon)$$

for certain $n,m \in \mathbb{N}$, certain $a_i,b_j \in A\cup\{\delta\}$, basic terms $t_i$ and the summand $\varepsilon$ may or may not occur. If the summand $\varepsilon$ does not occur, we must have $n+m > 0$.
We usually abbreviate such expressions, in this case to $t = \Sigma_{i<n}\, a_it_i + \Sigma_{j<m}\, b_j\ (+\varepsilon)$. Note that we can always write $t = \Sigma_{i<n}\, a_it_i + \Sigma_{j<m}\, b_j + \surd(t)$, for it is easy to see that $\surd(t) = \varepsilon$ iff $t$ has a summand $\varepsilon$, and $\surd(t) = \delta$ otherwise.
The set of basic terms **BT** can be inductively built up as follows (working modulo laws A1-3 and A9):
1. $\varepsilon \in$ BT
2. if $a \in A\cup\{\delta\}$ and $x \in$ BT, then $ax \in$ BT
3. if $x,y \in$ BT, then $x+y \in$ BT.
Alternatively, if $\Sigma_{i<0}\, x_i$ denotes $\delta$, we can build up BT as follows:
– If $n \in \mathbb{N}$, $a_i \in A\cup\{\delta\}$ and $t_i \in$ BT (for $i<n$), then $\Sigma_{i<n}\, a_it_i\ (+\varepsilon) \in$ BT.
Both these inductive schemes can be used in proofs.

3.8 DEFINITION. Let $p$ be a process. We say $p$ has a **head normal form** if there is an $n \in \mathbb{N}$, processes $p_i$ ($i<n$), and constants $a_i \in A\cup\{\delta\}$ such that $p = \Sigma_{i<n}\, a_ip_i\ (+\varepsilon)$.
Note that by definition, all basic terms have a head normal form. It is easy to prove that all processes that have a head normal form satisfy Te5 of 3.5, and the following equations:
1. $\surd(x\lfloor y) = \surd(x\,|\,y) = \delta$ 2. $\surd(\surd(x)) = \surd(x)$.

3.9 THEOREM. For every closed ACP$\sqrt{}$-term t there is a basic term s such that ACP$\sqrt{} \vdash t = s$.
This is the so-called **elimination theorem.**
PROOF: Let RACP$\sqrt{}$ be the full system ACP$\sqrt{}$, excluding axioms A1-2 and EM5, but including equation $a \between x = ax$, used as a rewrite system (from left to right), modulo axioms A1-2 and EM5. Working *modulo* axioms A1-2 and EM5 means that we consider terms that are equal using these axioms, to be identical. Note that $a \between x = ax$ follows from A9, EM3 and $\varepsilon \| x = x$. Below we will prove that, using RACP$\sqrt{}$, any closed term t can be rewritten to a basic term s, and moreover ACP$\sqrt{} \vdash \varepsilon \| t = t$ for closed terms t. From this the elimination theorem follows.
Let RACP$\sqrt{}$–E denote the system RACP$\sqrt{}$ without the rewrite rules E0-4.
We start by proving that RACP$\sqrt{}$–E is a terminating rewrite system on $\varepsilon_K$-free terms, and all its normal forms are basic terms. We first need some definitions. We define the **length** and **width** of a closed ACP$\sqrt{}$-term t without occurrences of the $\varepsilon_K$-operator inductively in table 4 below. As an auxiliary operator, we also define Te(t), the number of termination possibilities of t. The awkward expression for w(u||v) is the result of working out the four summands of EM1.
Roughly, the length of a term indicates the maximal number of steps that can occur when the term is executed, and the width gives the number of alternatives at the start of the execution, multiplied by 2 for every renaming operator $\partial_H$ around the term. Finally, we define the **size** of t, **s(t)**, to be the pair <l(t), w(t)>, with pairs ordered alphabetically.

| t = | Te(t) | l(t) | w(t) |
|---|---|---|---|
| $\varepsilon$ | 1 | 1 | 1 |
| $a \in A \cup \{\delta\}$ | 0 | 1 | 1 |
| $u + v$ | Te(u) + Te(v) | max(l(u),l(v)) | w(u) + w(v) |
| $u \cdot v$ | Te(u)·Te(v) | l(u) + l(v) | w(u) + Te(u)·w(v) |
| $u \| v$ | Te(u)·Te(v) | l(u) + l(v) | 3·w(u) + (1 + 2·Te(u))·w(v) + w(u)·w(v) |
| $u \between v$ | 0 | l(u) + l(v) | w(u) |
| $u \mid v$ | 0 | l(u) + l(v) | w(u)·w(v) |
| $\partial_H(u)$ | Te(u) | l(u) | 2·w(u) |

Table 4. Termination count, length and width of an $\varepsilon_K$-free ACP$\sqrt{}$-term.

The proof now proceeds via a number of claims.
CLAIM 1: Let t be a closed ACP$\sqrt{}$-term with no $\varepsilon_K$-operator. Then:
i. application of A1-2, EM5 or a rewrite rule does not increase the size of t;
ii. any proper subterm of t has a smaller size than t.
PROOF: Easy.
CLAIM 2: The rewrite system RACP$\sqrt{}$–E is (strongly) terminating for closed ACP$\sqrt{}$-terms without $\varepsilon_K$-operator.
PROOF: Suppose it is not terminating. Let t be a closed ACP$\sqrt{}$-term of minimal size, such that there is an infinite reduction sequence $t \to t_1 \to t_2 \to$ ..... A reduction on $t_i$ is called **external** (outermost) if it works on the main operator of $t_i$, and **internal** if it works on a proper subterm of $t_i$. From claim 1 it follows that it is not possible that from some $i \in \mathbb{N}$ on, the sequence consists of internal reductions only. Therefore, there must be infinitely many external reductions in this sequence. Now note the following facts:
- Among these external reductions there are no reductions A3, A6-9, CF1-2, EM2,6 or D0-2, since they decrease the size of the term, contradicting the minimality of t.
- Therefore, there are no external reductions in the sequence, working on a term u + v, and hence there are no external reductions resulting in a term u + v.
- Thus, all external reductions in the sequence are from the list A5, EM3,7, D4 and $a \between x = ax$, so result in a term $u \between v$ or $u \cdot v$.

• The allowed external reductions working on a term $u \between v$ (EM3 and $a \between x = ax$) result in a term $u' \cdot v'$.
• The only allowed external reduction working on a term $u \cdot v$ is A5. It results in another term $u' \cdot v'$, but with $u'$ having a smaller size than $u$.
Thus, apart from the first two, all external reductions must be A5-reductions. Therefore, in $t \to t_1 \to t_2 \to \ldots$ we have, from some $i$ on, $t_i = u_i \cdot v_i$, with $s(u_i)$ decreasing with each external reduction. This is impossible, and so claim 2 is proved.
CLAIM 3: All closed terms without $\varepsilon_K$-operator, which are normal forms w.r.t. the rewrite system RACP$\surd$–E, are basic terms.
PROOF: By induction on the structure of closed normal forms $t$. $t$ must be a constant $a \in A$, $\varepsilon, \delta$, or a term $u+v$, $u \cdot v$, $u \| v$, $u \between v$, $u \mid v$ or $\partial_H(u)$. Since also $u$ and $v$ are normal forms, we may assume that they are basic terms. If $t = u \cdot v$ with $u \notin A$, if $t = u \| v$, $u \between v$, $u \mid v$ or $\partial_H(u)$, or if $t$ has more than one $\varepsilon$-summand, then $t$ cannot be a normal form. In the other cases $t$ is a basic term.
CLAIM 4: Using RACP$\surd$, any closed term $t$ can be rewritten to a basic term $s$.
PROOF: For terms without $\varepsilon_K$-operator this follows from claims 2 and 3. For the general case it suffices to prove that for all basic terms $t$ there is a basic term $s$ such that $\varepsilon_K(t)$ reduces to $s$ in RACP$\surd$. This only requires a straightforward induction on the structure of basic terms.
CLAIM 5: ACP$\surd \vdash \varepsilon \| t = t$ for closed terms $t$.
$+ \sum_{j<m} b_j$ $(+ \varepsilon)$. For any application of $a \between x = ax$ in this process, claim 1 learns that $x$ has a smaller size than $t$. Thus ACP$\surd \vdash a \between x = a\varepsilon \between x = a(\varepsilon \| x) = ax$ (by induction) and therefore ACP$\surd \vdash t = s$. Thus ACP$\surd \vdash \varepsilon \| t = \varepsilon \| s = \varepsilon \between s + s \between \varepsilon + \varepsilon \mid s + \surd(\varepsilon)\surd(s) = \delta + \sum_{i<n} a_i(s_i \| \varepsilon) + \sum_{j<m} b_j(\varepsilon \| \varepsilon) + \delta + \surd(s) = \sum_{i<n} a_i s_i + \sum_{j<m} b_j \varepsilon + \surd(s)$ (by induction) $= s = t$.
The elimination theorem now follows from claims 4 and 5.
Note that as a consequence of the elimination theorem, all closed terms have a head normal form.

3.10 PROPOSITION. For all closed ACP$\surd$-terms $x,y,z$ we have the following laws of **standard concurrency**:

$$\varepsilon \| x = x$$
$$\surd(x) = \varepsilon \text{ iff } x = x + \varepsilon, \text{ and } \surd(x) = \delta \text{ otherwise}$$
$$\surd(x \| y) = \surd(x) \cdot \surd(y)$$
$$x \mid (y \mid z) = (x \mid y) \mid z \qquad (x \between y) \between z = x \between (y \| z)$$
$$(x \mid y) \between z = x \mid (y \between z) \qquad x \| (y \| z) = (x \| y) \| z.$$

PROOF: The first one is proved in 3.9. The second and third are easy to prove for all head normal forms, and therefore hold for all closed terms by 3.9.
For the others, note that because of the elimination theorem we can assume that $x,y,z$ are basic terms. We use the second induction scheme in 3.7. Write

$$x = \sum_{i \le n} a_i x_i \ (+ \varepsilon),\ y = \sum_{j \le m} b_j y_j \ (+ \varepsilon) \text{ and } z = \sum_{k \le p} c_k z_k \ (+ \varepsilon)$$

$(a_i, b_j, c_k \in A \cup \{\delta\})$. By induction hypothesis, we can assume that the proposition holds for all triples $(x_i,y,z)$, $(x,y_j,z)$, $(x,y,z_k)$. Then:
1. $x \mid (y \mid z) = \sum_{i,j,k} (a_i \mid (b_j \mid c_k))(x_i \| (y_j \| z_k))$ (EM6, 3.4.5) $= \sum_{i,j,k} ((a_i \mid b_j) \mid c_k)((x_i \| y_j) \| z_k)$ (definition of $\gamma$, induction hypothesis) $= (x \mid y) \mid z$.
2. $(x \between y) \between z = (\{\sum_i a_i x_i \ (+ \varepsilon)\} \between y) \between z = \sum_i (a_i(x_i \| y)) \between z = \sum_i a_i((x_i \| y) \| z) = \sum_i a_i(x_i \| (y \| z))$ (induction hypothesis) $= \sum_i a_i x_i \between (y \| z) = x \between (y \| z)$.
3. $(x \mid y) \between z = (\sum_{i,j} (a_i \mid b_j)(x_i \| y_j)) \between z = \sum_{i,j} (a_i \mid b_j)((x_i \| y_j) \| z) =$
$= \sum_{i,j} (a_i \mid b_j)(x_i \| (y_j \| z))$ (induction hypothesis) $= \sum_{i,j} a_i x_i \mid b_j(y_j \| z) = x \mid (y \between z)$.
4. $x \| (y \| z) = x \between (y \| z) + (y \| z) \between x + x \mid (y \| z) + \surd(x) \cdot \surd(y \| z) =$ (using 3.6.2-3)
$= x \between (y \| z) + (y \between z) \between x + (z \between y) \between x + (y \mid z) \between x + x \mid (y \between z) + x \mid (z \between y) + x \mid (y \mid z) + \surd(x) \cdot \surd(y \| z) =$
$= (x \between y) \between z + y \between (z \| x) + z \between (y \| x) + (y \mid z) \between x + (x \mid y) \between z + (x \mid z) \between y + (x \mid y) \mid z + \surd(x)\surd(y)\surd(z) =$
(using 3.6.1 and EM5)
$= (x \between y) \between z + y \between (x \| z) + z \between (x \| y) + (z \mid y) \between x + (x \mid y) \between z + (z \mid x) \between y + z \mid (x \mid y) + \surd(x)\surd(y)\surd(z) =$

$= (x \mathbin{\mathbb{L}} y) \mathbin{\mathbb{L}} z + (y \mathbin{\mathbb{L}} x) \mathbin{\mathbb{L}} z + z \mathbin{\mathbb{L}} (x \| y) + z \mid (y \mathbin{\mathbb{L}} x) + (x \mid y) \mathbin{\mathbb{L}} z + z \mid (x \mathbin{\mathbb{L}} y) + z \mid (x \mid y) + \surd(x \| y) \cdot \surd(z) =$
$= (x \| y) \mathbin{\mathbb{L}} z + z \mathbin{\mathbb{L}} (x \| y) + z \mid (x \| y) + \surd(x \| y) \cdot \surd(z) = (x \| y) \| z.$

3.11 NOTE. We usually assume that the laws of Standard Concurrency hold for all processes. Therefore, they are often called the *axioms* of Standard Concurrency.
Often, we also assume the following **Handshaking Axiom**:

$$x \mid y \mid z = \delta \qquad \text{(HA)}.$$

It says, that all communication is *binary*, i.e. only involves two communication partners.

3.12 PROPOSITION. In ACP$\surd$ with standard concurrency and handshaking axiom we have the following **expansion theorem** ($n \geq 1$):

$$\|_{i \leq n} x_i = \sum_{i \leq n} x_i \mathbin{\mathbb{L}} (\|_{k \leq n, k \neq i} x_k) + \sum_{i<j \leq n} (x_i \mid x_j) \mathbin{\mathbb{L}} (\|_{k \leq n, k \neq i,j} x_k) + \prod_{i \leq n} \surd(x_i)$$

(Where $\|_{i \leq n} x_i$ means $x_0 \| \ldots \| x_n$, and $\prod_{i \leq n} x_i$ means $x_0 \cdot \ldots \cdot x_n$.)
PROOF: This follows from the axioms of standard concurrency and the handshaking axiom similar to the case of ACP (BERGSTRA & TUCKER [6]) or ACP$^\varepsilon$ (VRANCKEN [14]). The only difference is, that we have to keep track of the termination option.
We use induction on $n$. The case $n=1$ is exactly the axiom EM1. The induction step is as follows:
$\|_{i \leq n+1} x_i = (\|_{i \leq n} x_i) \| x_{n+1} =$
$= (\|_{i \leq n} x_i) \mathbin{\mathbb{L}} x_{n+1} + x_{n+1} \mathbin{\mathbb{L}} (\|_{i \leq n} x_i) + (\|_{i \leq n} x_i) \mid x_{n+1} + \surd(\|_{i \leq n} x_i) \surd(x_{n+1}).$
We consider these four terms in turn, and use the induction hypothesis. The first:
$(\|_{i \leq n} x_i) \mathbin{\mathbb{L}} x_{n+1} =$
$= \Sigma_{i \leq n} (x_i \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i} x_k)) \mathbin{\mathbb{L}} x_{n+1} + \Sigma_{i<j \leq n} ((x_i \mid x_j) \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i,j} x_k)) \mathbin{\mathbb{L}} x_{n+1} + \delta$ (by 3.6 and EM4) =
$= \Sigma_{i \leq n} x_i \mathbin{\mathbb{L}} ((\|_{k \leq n,\, k \neq i} x_k) \| x_{n+1}) + \Sigma_{i<j \leq n} (x_i \mid x_j) \mathbin{\mathbb{L}} ((\|_{k \leq n,\, k \neq i,j} x_k) \| x_{n+1})$ (use 3.10) =
$= \Sigma_{i \leq n} x_i \mathbin{\mathbb{L}} (\|_{k \leq n+1,\, k \neq i} x_k) + \Sigma_{i<j \leq n} (x_i \mid x_j) \mathbin{\mathbb{L}} ((\|_{k \leq n+1,\, k \neq i,j} x_k).$
The second term is equal to $x_{n+1} \mathbin{\mathbb{L}} (\|_{k \leq n+1,\, k \neq n+1} x_k)$, and the third:
$(\|_{i \leq n} x_i) \mid x_{n+1} =$
$= \Sigma_{i \leq n} (x_i \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i} x_k)) \mid x_{n+1} + \Sigma_{i<j \leq n} ((x_i \mid x_j) \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i,j} x_k)) \mid x_{n+1} + \delta$ (by 3.6 and EM8) =
$= \Sigma_{i \leq n} (x_i \mid x_{n+1}) \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i} x_k) + \Sigma_{i<j \leq n} (x_i \mid x_j \mid x_{n+1}) \mathbin{\mathbb{L}} (\|_{k \leq n,\, k \neq i,j} x_k)$ (use 3.10) =
$= \Sigma_{i<n+1} (x_i \mid x_{n+1}) \mathbin{\mathbb{L}} (\|_{k \leq n+1,\, k \neq i,n+1} x_k)$ (by handshaking axiom).
Finally, the fourth term is equal to $\prod_{i \leq n+1} \surd(x_i)$ by the third axiom of standard concurrency.
Adding the obtained expressions gives the desired result.

3.13 Until now, we have mainly looked at closed terms. However, most processes encountered in practice cannot be represented by a closed term, by an element of the initial algebra of ACP$\surd$, but will be specified recursively. Therefore, we are interested in models that also contain infinite processes, processes that can perform infinitely many actions consecutively. The algebraic way to represent such processes is by means of recursive specifications. In this section, we introduce some terminology.

3.14 DEFINITION. A **recursive specification** over ACP$\surd$ is a set of equations $\{X = t_X : X \in V\}$, with $V$ a set of variables, and $t_X$ a term over ACP$\surd$ and variables $V$. No other variables may occur in $t_X$. There is exactly one equation $X = t_X$ for each variable $X$.
A **solution** of the recursive specification $E$ (in a certain domain) is an interpretation of the variables of $V$ as processes such that the equations of $E$ are satisfied.
The **Recursive Definition Principle (RDP)** says that every recursive specification has a solution. In section 4, we will discuss models of ACP$\surd$ that satisfy RDP.
Recursive specifications are used to define (or specify) processes. If $E$ has a unique solution, and $X \in V$, let $\langle X \mid E \rangle$ denote the $X$-component of this solution. If $E$ has more than one solution, $\langle X \mid E \rangle$ denotes 'one of the solutions of $E$', and can be regarded as a kind of variable, ranging over these

solutions. If E has no solutions (possible in a model, not satisfying RDP), then no meaning can be attached to $\langle X \mid E \rangle$. In a recursive language, the syntactical constructs $\langle X \mid E \rangle$ may appear in the construction of terms. This limits the class of models of the language to the ones satisfying RDP.
If $E = \{X = t_X : X \in V\}$ is a recursive specification, and t a term, then $\langle t \mid E \rangle$ denotes the term t in which each occurrence of a variable $X \in V$ is replaced by $\langle X \mid E \rangle$. Thus, the assumption that the terms $\langle X \mid E \rangle$ are solutions of E may be stated as follows ($X \in V$):

$$\langle X \mid E \rangle = \langle t_X \mid E \rangle.$$

Note that we cannot have that *every* recursive specification has a unique solution, for $E = \{X = X\}$ has every process as a solution. Therefore, we formulate the condition of guardedness below, and will claim that in the models of section 4, every guarded recursive specification does have a unique solution.

3.15 DEFINITION. i. Let t be a term over ACP$\sqrt{}$ without $\varepsilon_K$-operator, and X a variable in t. We call the occurrence of X in t **guarded** if X is *preceded* by an atomic action, i.e. t has a subterm of the form $a \cdot s$, with $a \in A$, and this X occurs in s. Otherwise, we call the occurrence of X **unguarded**.
ii. A recursive specification $\{X = t_X : X \in V\}$ is **guarded** if no $\varepsilon_K$-operator appears and each occurrence of a variable in each $t_X$ is guarded.
iii. The **Recursive Specification Principle (RSP)** is the assumption that every guarded recursive specification has at most one solution. Thus, in a model satisfying RDP and RSP, each guarded recursive specification has a unique solution. Also note that each solution of a guarded recursive specification has a head normal form, so results 3.6 hold for such processes.

3.16 NOTE. In section 5, we will formulate the **Limit Rule (LR)**, and we will prove that a restricted version holds in the models of section 4. The Limit Rule says that any equation that holds for all closed terms, holds for all processes.
As a corollary, we find that the axioms of Standard Concurrency of 3.10 hold in the models.

## 4. SEMANTICS

We consider different semantics for ACP$\sqrt{}$. First, we define a term model (using the syntactical constructs $\langle X \mid E \rangle$ of 3.14) by means of action relations. Action relations appear in MILNER [11], PLOTKIN [13] and in the setting of process algebra, in VAN GLABBEEK [7].

4.1 DEFINITION. Let $\mathbb{P}$ be the set of **process expressions**, closed terms over the signature of ACP$\sqrt{}$ and recursion constructs $\langle X \mid E \rangle$ of 3.14. On $\mathbb{P}$, we define binary predicates $\rightarrow^a$ for each $a \in A$, and a unary predicate $\downarrow$, generated by the rules in table 5 below.

| | |
|---|---|
| $a \rightarrow^a \varepsilon$ | $\varepsilon\downarrow$ |
| $x \rightarrow^a x' \Rightarrow x+y \rightarrow^a x' \,\&\, y+x \rightarrow^a x'$ | $x\downarrow \Rightarrow (x+y)\downarrow \,\&\, (y+x)\downarrow$ |
| $x \rightarrow^a x' \Rightarrow xy \rightarrow^a x'y$ | |
| $x\downarrow \,\&\, y \rightarrow^a y' \Rightarrow xy \rightarrow^a y'$ | $x\downarrow \,\&\, y\downarrow \Rightarrow (xy)\downarrow$ |
| $x \rightarrow^a x' \Rightarrow x\|\|y \rightarrow^a x'\|\|y,\ y\|\|x \rightarrow^a y\|\|x' \,\&\, x\lfloor\!\lfloor y \rightarrow^a x'\|\|y$ | $x\downarrow \,\&\, y\downarrow \Rightarrow (x\|\|y)\downarrow$ |
| $x \rightarrow^a x' \,\&\, y \rightarrow^b y' \,\&\, \gamma(a,b) = c \Rightarrow x\|\|y \rightarrow^c x'\|\|y' \,\&\, x \mid y \rightarrow^c x'\|\|y'$ | |
| $x \rightarrow^a x' \,\&\, a \notin H \Rightarrow \partial_H(x) \rightarrow^a \partial_H(x')$ | $x\downarrow \Rightarrow \partial_H(x)\downarrow$ |
| $x \rightarrow^a x' \,\&\, a \notin K \Rightarrow \varepsilon_K(x) \rightarrow^a \varepsilon_K(x')$ | $x\downarrow \Rightarrow \varepsilon_K(x)\downarrow$ |
| $x \rightarrow^a x',\ a \in K \,\&\, \varepsilon_K(x') \rightarrow^b y \Rightarrow \varepsilon_K(x) \rightarrow^b y$ | $x \rightarrow^a x',\ a \in K \,\&\, \varepsilon_K(x')\downarrow \Rightarrow \varepsilon_K(x)\downarrow$ |
| $\langle t_X \mid E \rangle \rightarrow^a y \Rightarrow \langle X \mid E \rangle \rightarrow^a y$ | $\langle t_X \mid E \rangle\downarrow \Rightarrow \langle X \mid E \rangle\downarrow$ |

Table 5. Action relations.

The intuitive meaning is as follows:
• $x \to^a y$ means that x can evolve into y by executing the atomic action a,
• $x\downarrow$ means that x has a termination option.
Note that we defined the action relations in such a way that $x \to^a x'$ iff $\varepsilon x \to^a x'$, and $x\downarrow$ iff $\varepsilon x\downarrow$. This is why we can consider the process expression p to be identical to the expression $\varepsilon p$, i.e. we consider process expressions modulo axiom A8. This identification makes the following proofs easier.

4.2 DEFINITION. A **bisimulation** is a binary relation R on $\mathbb{P}$, satisfying (for $a \in A$):
1. if $R(p,q)$ and $p \to^a p'$, then there is a $q'$ such that $q \to^a q'$ and $R(p',q')$;
2. if $R(p,q)$ and $q \to^a q'$, then there is a $p'$ such that $p \to^a p'$ and $R(p',q')$;
3. if $R(p,q)$, then $p\downarrow$ if and only if $q\downarrow$.
If there exists a bisimulation R with $R(p,q)$, we say p and q **are bisimilar**, and write $p \leftrightarrows q$.
The notion of bisimulation was introduced by PARK [12]. Also see MILNER [11], BERGSTRA & KLOP [4] and VRANCKEN [14].

4.3 THEOREM. $\leftrightarrows$ is a congruence on ACP$\sqrt{}$-terms.
PROOF: We have to check the following:
1. $p \leftrightarrows p$ 2. $p \leftrightarrows q \Rightarrow q \leftrightarrows p$ 3. $p \leftrightarrows q \,\&\, q \leftrightarrows r \Rightarrow p \leftrightarrows r$
4. $p \leftrightarrows p' \,\&\, q \leftrightarrows q' \Rightarrow p \square q \leftrightarrows p' \square q'$ for $\square = +, \cdot, \|, \between, \between, |$; $\partial_H(p) \leftrightarrows \partial_H(p')$, likewise for $\varepsilon_K$.
Now, let $p,q,r,p',q' \in \mathbb{P}$ and let R,S be bisimulations on $\mathbb{P}$.
We define the following relations on $\mathbb{P}$.
I: $I(p,p)$ for $p \in \mathbb{P}$.
$R^{-1}$: $R^{-1}(p,q)$ iff $R(q,p)$.
$R \circ S$: $R \circ S(p,r)$ iff $\exists q \in \mathbb{P}$: $R(p,q)$ and $S(q,r)$.
$R \cdot q$: $R \cdot q(p \cdot q, p' \cdot q)$ iff $R(p,p')$.
$R \| S$: $R \| S(p \| q, p' \| q')$ iff $R(p,p')$ and $S(q,q')$.
$\partial_H(R)$, $\varepsilon_K(R)$: $\partial_H(R)(\partial_H(p), \partial_H(q))$ iff $R(p,q)$, and similarly for $\varepsilon_K$.
Now 1,2 and 3 follow since I, $R^{-1}$ and $R \circ S$ are bisimulations (as can be checked easily).
For 4, suppose $R(p,p')$ and $S(q,q')$.
+: $R \cup S \cup \{(p+q, p'+q')\}$ is a bisimulation relating $p + q$ and $p' + q'$;
$\cdot$ $R \cdot q \cup S$ is a bisimulation relating $p \cdot q$ and $p' \cdot q'$;
$\|$: $R \| S$ is a bisimulation relating $p \| q$ and $p' \| q'$;
$\between$: $R \| S \cup \{(p \between q, p' \between q')\}$ is a bisimulation relating $p \between q$ and $p' \between q'$;
$|$: $R \| S \cup \{(p \mid q, p' \mid q')\}$ is a bisimulation relating $p \mid q$ and $p' \mid q'$;
$\partial_H$, $\varepsilon_K$: $\partial_H(R)$ is a bisimulation relating $\partial_H(p)$ and $\partial_H(p')$, and $\varepsilon_K(R)$ one relating $\varepsilon_K(p)$ and $\varepsilon_K(p')$.

4.4 THEOREM. $\mathbb{P}/\leftrightarrows$ is a model of ACP$\sqrt{}$.
PROOF: Straightforward.

4.5 THEOREM. ACP$\sqrt{}$ is a complete axiomatisation of $\mathbb{P}/\leftrightarrows$ for closed terms (without recursion constructs <X|E>).
PROOF: We have to show that if $t \leftrightarrows s$ holds for closed terms t,s, then ACP$\sqrt{} \vdash t = s$. Since $\mathbb{P}/\leftrightarrows$ is a model for ACP$\sqrt{}$, the elimination theorem tells us that we only have to prove this for basic terms t,s. For basic terms, this follows by means of a structural induction argument (using the second inductive scheme from 3.7) from the following two observations, that are not hard to prove:
i. $t \to^a t'$ iff t has a summand $at'$;

ii. $t\downarrow$ iff t has a summand $\varepsilon$.

4.6 THEOREM. RDP holds in $\mathbb{P}/\leftrightarrow$.
PROOF: This is immediate: the $\leftrightarrow$-congruence class of $\langle X \mid E\rangle$ is the X-component of a solution of E in $\mathbb{P}/\leftrightarrow$.

It takes some more work to prove that the principles RSP and LR hold in $\mathbb{P}/\leftrightarrow$. Therefore, we will skip this here, and treat this in section 5.
In the sequel, we describe a graph model for ACP$\sqrt{}$, that can be considered as a visualisation of the model $\mathbb{P}/\leftrightarrow$ above.

4.7 DEFINITIONS. In this paper, a **graph** is a *rooted, countably branching, directed multigraph*. An edge goes from a node to another (or the same) node. We consider only countably branching graphs, so each node has only a countable number of outgoing edges. Graphs need not be finite (have finitely many nodes and edges), but we must be able to reach every node from the root in finitely many steps. An **endnode** of a graph is a node with no outgoing edges. A **path** $\pi$ in a graph g is a finite alternating sequence of connected nodes and edges of g. A **tree** is a graph in which the root has no incoming edge, and all other nodes have exactly one incoming edge. Note that a tree has no **cycles**, no path from a node back to the same node. The **zero graph** $\mathbb{0}$ consists of a single node and no edges.
A **process graph** is a graph in which each edge is labeled with an element of A, the set of atomic actions, and nodes may have a label $\downarrow$. Such nodes are called **termination nodes**. $\mathbb{G}_\downarrow$ is the set of all process graphs. An **a-step** in a process graph from s to s' is an edge going from s to s' with label $a \in A$, notation $s \rightarrow^a s'$.

4.8 DEFINITION. We define a map graph from the set of process expressions $\mathbb{P}$ to the set of process graphs $\mathbb{G}_\downarrow$ as follows. Let $p \in \mathbb{P}$. graph(p) has a node for each $q \in \mathbb{P}$ that is reachable from p (i.e. there is a series of atomic actions $a_1,\ldots,a_n$ such that $p \rightarrow^{a_1} \ldots \rightarrow^{a_n} q$). The node corresponding to p itself will be the root of graph(p). There is an edge labeled a between two nodes exactly when the a-labeled action relation holds between the corresponding process expressions. A node receives an $\downarrow$-label exactly when the corresponding process expression can terminate.
Conversely, we define a map term from $\mathbb{G}_\downarrow$ to $\mathbb{P}$ as follows. Let $g \in \mathbb{G}_\downarrow$. We define a guarded recursive specification E as follows: take a variable $X \in V$ for every node in g. Then, if the node X has outgoing edges labeled $a_1,\ldots,a_n$, to nodes $X_1,\ldots,X_n$ respectively, we take as equation for X in E:

$$X = a_1X_1 + \ldots + a_nX_n\ (+\ \varepsilon),$$

with the summand $\varepsilon$ appearing iff X has a $\downarrow$-label. If X has no outgoing edges, and no $\downarrow$-label, we put $X = \delta$. Then, if $X_0$ is the variable of the root of g, we define $\mathrm{term}(g) = \langle X_0 \mid E\rangle$.
Next, the notion of bisimulation translates easily to the present case, as we see below.

4.9 DEFINITION. Let $g,h \in \mathbb{G}_\downarrow$, and let R be a relation between the nodes of g and the nodes of h. R is a **bisimulation** between g and h, notation $R: g \leftrightarrow h$, iff
1. the roots of g and h are related;
2. if R(s,t) and $s \rightarrow^a s'$ is an edge in g (with $a \in A$, s,s' nodes of g and t a node of h), then there is a node t' in h such that $t \rightarrow^a t'$ and R(s',t');
3. if R(s,t) and $t \rightarrow^a t'$ is an edge in h, then there is a node s' in g such that $s \rightarrow^a s'$ and R(s',t');
4. if R(s,t), then $s\downarrow$ (node s has a $\downarrow$-label) if and only if $t\downarrow$.
If there exists a bisimulation between g and h, we say g and h **are bisimilar**, and write $g \leftrightarrow h$.

4.10 PROPOSITION. i. If $p,q \in \mathbb{P}$ then $p \leftrightarrow q$ iff $\mathrm{graph}(p) \leftrightarrow \mathrm{graph}(q)$;
ii. If $g,h \in \mathbb{G}_\downarrow$ then $g \leftrightarrow h$ iff $\mathrm{term}(g) \leftrightarrow \mathrm{term}(h)$;

iii. For $g \in \mathbb{G}_{\downarrow}$ $graph(term(g)) \equiv g$, for $p \in \mathbb{P}$ $term(graph(p)) \leftrightarrows p$.
PROOF: In iii, $g \equiv h$ means that g and h are isomorphic. This is the case if there exists a bijective bisimulation between them. Let N be the set of nodes of graph $g \in \mathbb{G}_{\downarrow}$. Then $\{X_i : i \in N\}$ is the set of variables, used in the construction of term(g) and $\{\langle X_i \mid E\rangle : i \in N\}$ is the set of $q \in \mathbb{P}$, reachable from $term(g) = \langle X_0 \mid E\rangle$. This gives a bijective mapping between the node sets of g and graph(term(g)), which is clearly a bisimulation. So we have proved the first part of iii. Now i. is trivial and ii. as well as the second part of iii. follow easily:
– $g \leftrightarrows h \Leftrightarrow graph(term(g)) \leftrightarrows graph(term(g)) \Leftrightarrow term(g) \leftrightarrows term(h)$,
– $graph(term(g)) \leftrightarrows g$, so $graph(term(graph(p))) \leftrightarrows graph(p)$, so $term(graph(p)) \leftrightarrows p$.

4.11 From 4.10 we can conclude that we can define all operators on $\mathbb{G}_{\downarrow}$ as the image of the same operators on $\mathbb{P}$ (i.e. $g + h = graph(term(g) + term(h))$, etc.). It is also possible to define the operators explicitly on $\mathbb{G}_{\downarrow}$, but we will not do so here. Now, the models $\mathbb{G}_{\downarrow}/\leftrightarrows$ and $\mathbb{P}/\leftrightarrows$ become isomorphic models, and thus, $\mathbb{G}_{\downarrow}/\leftrightarrows$ is a sound and complete model of ACP$\sqrt{}$, in which RDP and RSP hold.

4.12 EXAMPLES. In fig. 1, we use an incoming arrow without a label to indicate the root of a graph, and an outgoing arrow without a label to indicate a termination node. In i, we see $graph(\delta) = graph(\langle X \mid X=X\rangle)$; in ii, $graph(\varepsilon)$; in iii. $graph(\langle X \mid X = aX\rangle)$; in iv, $graph(a + \varepsilon)$, in v, $graph(ab + ac)$, and in vi, $graph(a(b + c))$ (for $a,b,c \in A$). Note that the graphs in v, vi do not bisimulate.

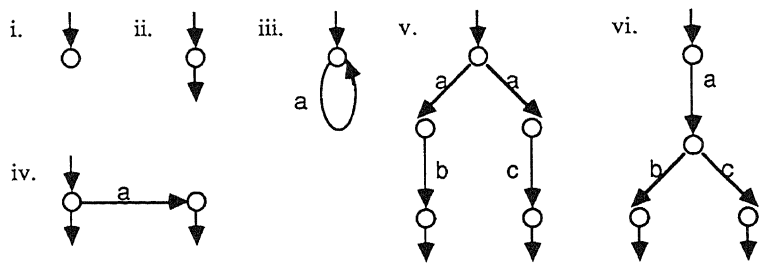

Fig. 1.

Although the graph model $\mathbb{G}_{\downarrow}/\leftrightarrows$ is very useful, we still want to present another graph model $\mathbb{G}_{\varepsilon\delta}/\leftrightarrows_{\varepsilon}$, which is more denotational. In this second graph model, we also have edges with label ε and δ. This increases the expressive power, and simplifies the definition of operators + and $\varepsilon_K$, but makes the definition of bisimulation and the operators $\|, \lfloor\!\lfloor, |$ harder. The model $\mathbb{G}_{\varepsilon\delta}/\leftrightarrows_{\varepsilon}$ is essentially the same model as the graph model of VRANCKEN [14].

4.13 DEFINITION. A process graph in the set $\mathbb{G}_{\varepsilon\delta}$ differs from a graph in $\mathbb{G}_{\downarrow}$ in three aspects: first, graphs must be finitely branching, second, edges are labeled with elements of $A \cup \{\delta,\varepsilon\}$, and third, we have no node labels. We will see that the restriction to finitely branching graphs is not a real restriction.

4.14 DEFINITION. We define the notion of an ε-bisimulation on $\mathbb{G}_{\varepsilon\delta}$ as in VRANCKEN [14]. In this definition, we need the following notation: $\twoheadrightarrow^{\varepsilon}$ stands for a path of ε-edges, a connected series of 0 or more ε-steps (so $\twoheadrightarrow^{\varepsilon}$ is the transitive and reflexive closure of $\rightarrow^{\varepsilon}$).
Let g,h be process graphs, and let R be a relation between nodes of g and nodes of h. R is an ε-**bisimulation** between g and h, notation $R: g \leftrightarrows_{\varepsilon} h$, iff

1. The roots of g and h are related.
2. If R(s,t) and from s, we can do a generalized ε-step followed by an a-step to a node s' ($s \to^{\varepsilon} \to^{a} s'$) with $a \in A$ (so $a \neq \varepsilon$, $a \neq \delta$), then from t in h, we can do a generalized ε-step, followed by an a-step to a node t' with R(s',t'), so $t \to^{\varepsilon} \to^{a} t'$.
3. Vice versa: if R(s,t) and $t \to^{\varepsilon} \to^{a} t'$ is a path in h with $a \in A$, then, in g, there is a node s' such that $s \to^{\varepsilon} \to^{a} s'$ and R(s',t').
4. If R(s,t), s' is an endnode in g, and $s \to^{\varepsilon} s'$, then, in h, there is a node t' such that t' is an endpoint and $t \to^{\varepsilon} t'$.
5. Vice versa: if R(s,t), t' is an endnode in h, and $t \to^{\varepsilon} t'$, then, in g, there is a node s' such that s' is an endpoint and $s \to^{\varepsilon} s'$.

Graphs g and h are **ε-bisimilar**, $g \leftrightarrows_{\varepsilon} h$, if there is an ε-bisimulation between g and h.

4.15 EXAMPLES. See fig. 2 below. We have $a,b \in A$, so $\neq \delta,\varepsilon$.

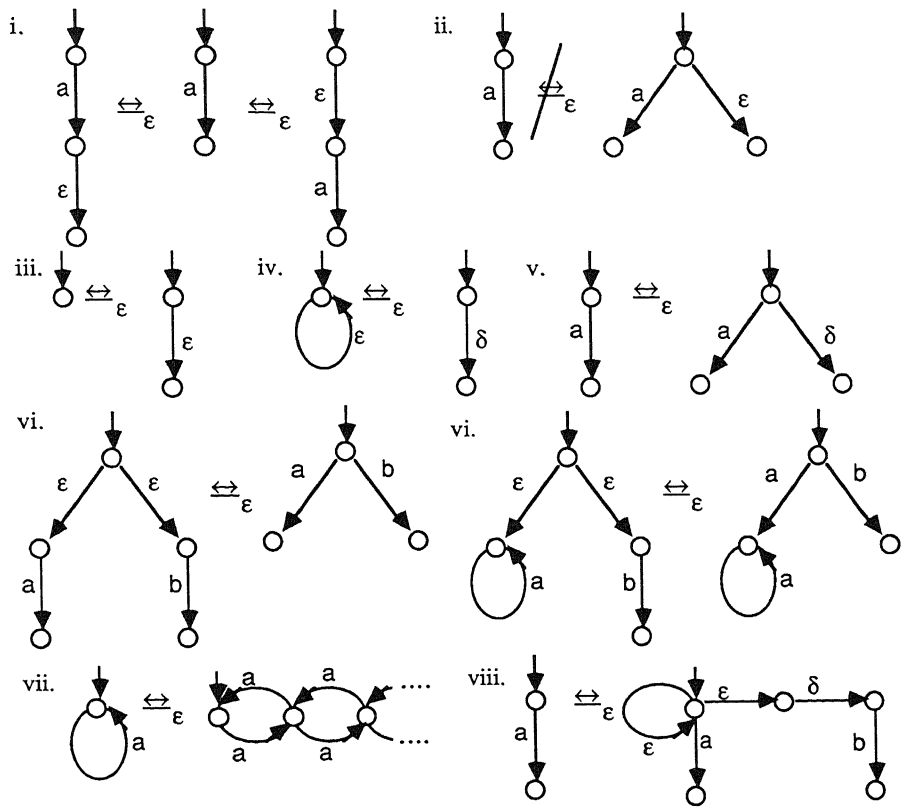


Fig. 2.

4.16 LEMMA. $\leftrightarrows_{\varepsilon}$ is an equivalence relation on $\mathbb{G}_{\varepsilon\delta}$.
PROOF: Straightforward.

4.17 LEMMA. Each $\leftrightarrows_{\varepsilon}$-equivalence class contains a nonzero process tree.
PROOF: Let g be a process graph. We find a tree h that is ε-bisimilar to g by *unrolling* g, i.e. we have a node in h for each path from the root in g. Edges and labels in h are defined in the obvious way; the root

of h corresponds to the empty path in g. We leave the details of this construction, and the verification of the ε-bisimilarity, to the reader. We use the notation tree(g) for the tree obtained by unrolling g. If h turns out to be the zero graph 0, we use the second tree in 4.15.iii instead.

4.18 $\mathbb{G}_{\varepsilon\delta}/\underline{\leftrightarrow}_\varepsilon$ will be the domain of the graph model for ACP$\sqrt{}$. The interpretation of a constant $u \in A\cup\{\delta,\varepsilon\}$ is the equivalence class of the graph with two nodes and a single edge between them labeled u. What remains is the definition of the operators of ACP$\sqrt{}$ on $\mathbb{G}_{\varepsilon\delta}/\underline{\leftrightarrow}_\varepsilon$. We will define these operators on $\mathbb{G}_{\varepsilon\delta}$ (the parallel operators only on process *trees*) and will then show that $\underline{\leftrightarrow}_\varepsilon$ is a congruence relation w.r.t. them.

4.19 DEFINITIONS. 1. +. If $g,h \in \mathbb{G}_{\varepsilon\delta}$, graph g+h is obtained by taking the graphs of g and h and adding one new node r, that will be the root of g+h. Then, we add two edges labeled ε: from r to the root of g, and from r to the root of h.
EXAMPLE:

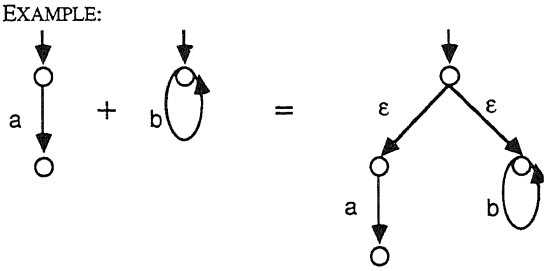


Fig. 3.

Note that it doesn't work to just identify the roots of g and h: if we do that in the example, it is possible to do an a-step after having done some b-steps. This fact is also illustrated in example 4.15.vi, and complicates the explicit definition of + in $\mathbb{G}_\downarrow$.
2. ·. If $g,h \in \mathbb{G}_{\varepsilon\delta}$, graph g·h is obtained by identifying all endpoints of g with the root node of h. If g has no endpoints, the result is just g. The root of g·h is the root of g.
EXAMPLE:

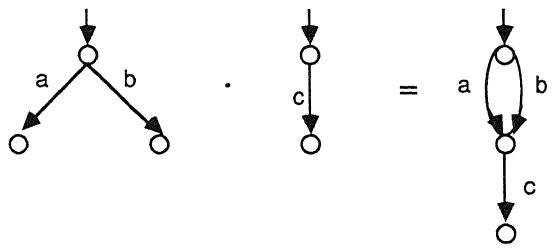


Fig. 4.

3. ||. The definition of the merge on $\mathbb{G}_{\varepsilon\delta}$ is rather complicated. Therefore, we will only define the merge on nonzero process *trees*. Using lemma 4.17, this definition can be extended to $\mathbb{G}_{\varepsilon\delta}$.
If g,h are nonzero process trees, graph g||h is the cartesian product graph of graphs g and h, with 'diagonal' edges added for communication steps, and with non-ε-edges 'orthogonal' to an incoming ε-step turned into δ-steps. By this, we mean the following: if (s,t) is a node in g||h, then it has the following outgoing edges ($u,v \in A\cup\{\delta,\varepsilon\}$, $a,b \in A$):
i. an edge $(s,t) \to^u (s',t)$ if $s \to^u s'$ is an edge in g, and $u = \varepsilon$ or h has no edge $t'' \to^\varepsilon t$;
ii. an edge $(s,t) \to^\delta (s',t)$ if $s \to^u s'$ is an edge in g, $u \neq \varepsilon$ and h has an edge $t'' \to^\varepsilon t$;

iii. an edge $(s,t) \to^{v} (s,t')$ if $t \to^{v} t'$ is an edge in h, and $u = \varepsilon$ or g has no edge $s'' \to^{\varepsilon} s$;
iv. an edge $(s,t) \to^{\delta} (s,t')$ if $t \to^{v} t'$ is an edge in h, $u \neq \varepsilon$ and g has an edge $s'' \to^{\varepsilon} s$;
v. an edge $(s,t) \to^{\gamma(a,b)} (s',t')$ if $s \to^{a} s'$ is an edge in g, $t \to^{b} t'$ is an edge in h and $\gamma(a,b)$ is defined (these are the *diagonal* edges).
The root of g||h is the pair of roots of g and h.
Edges $(s,t) \to^{u} (s',t)$ are called *vertical* edges, and edges $(s,t) \to^{u} (s,t')$ are *horizontal* edges.
EXAMPLE: Suppose $\gamma(a,b) = c$, and $\gamma(d,b)$ is not defined. See fig. 5.

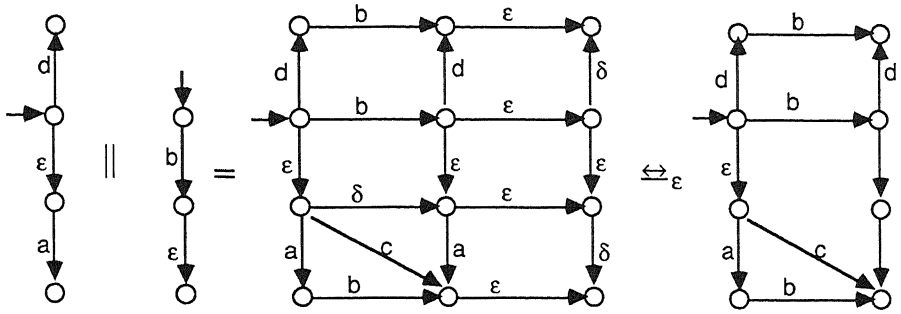


Fig. 5.

In this example, we see why some edges must be blocked, must be turned into δ: we have to make the one b-step δ, for if we start with a b-step, a d-step must still be possible.
4. ||. If g,h are nonzero process trees, graph g||h is obtained from graph g||h by turning all horizontal and diagonal edges, that are reachable from the root by a generalized ε-step, into δ-edges.
EXAMPLE: the last example turns into:

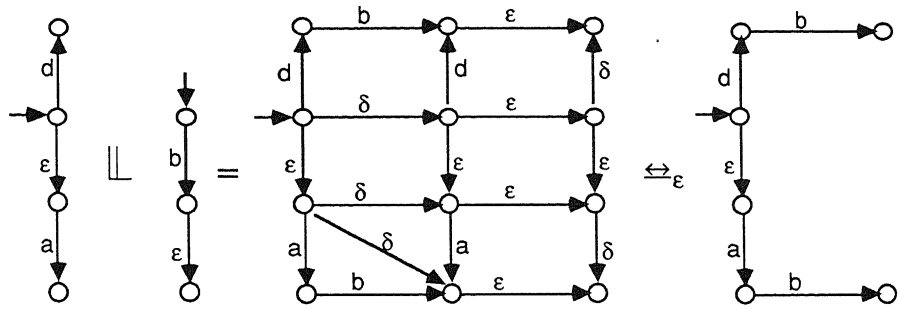


Fig. 6.

5. |. Similar to 4: If g,h are nonzero process trees, graph g|h is obtained from graph g||h by turning all horizontal and vertical edges, that are reachable from the root by a generalized ε-step, and do not have label ε, or do have label ε but lead to an endpoint, into δ-edges.
EXAMPLE: we use the same example. See following page.

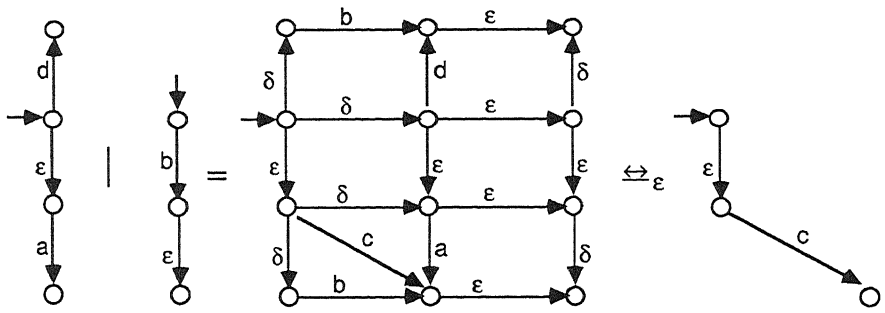


Fig. 7.

6. $\partial_H$, $\varepsilon_K$. If $g \in \mathbb{G}_{\varepsilon\delta}$, obtain $\partial_H(g)$ by replacing all labels in $g$ from $H$ by $\delta$, and obtain $\varepsilon_K(g)$ by replacing all labels from $K$ by $\varepsilon$.
This finishes the definition of the operators of ACP$\sqrt{}$ on $\mathbb{G}_{\varepsilon\delta}$. Then we also have the operators on $\mathbb{G}_{\varepsilon\delta}/\leftrightarrow_\varepsilon$, if we use the following theorem 4.21.

4.20 NOTE. In VRANCKEN [14], the parallel operators $\|, \lfloor\!\lfloor, |$ are defined on a wider class of graphs, a class which is closed under these operators. This makes proofs of statements about them much easier.

4.21 THEOREM. $\leftrightarrow_\varepsilon$ is a congruence relation on $\mathbb{G}_{\varepsilon\delta}$.
PROOF: As in VRANCKEN [14].

4.22 THEOREM. $\mathbb{G}_{\varepsilon\delta}/\leftrightarrow_\varepsilon$ is a model of ACP$\sqrt{}$.
PROOF: As in VRANCKEN [14].

4.23 REMARK. We also obtain models of ACP$\sqrt{}$, if instead of limiting ourselves to finitely branching graphs, we allow all countably branching graphs, the set $\mathbb{G}_{\varepsilon\delta}{}^{\infty}$. Also, the set $\mathbb{R}$ of all finite (or *regular*) process graphs modulo $\leftrightarrow_\varepsilon$ and the set $\mathbb{F}$ of all finite and acyclic process graphs modulo $\leftrightarrow_\varepsilon$ form models of ACP$\sqrt{}$.
As we already stated in 4.13, it does not matter that we limit ourselves to finitely branching graphs instead of countably branching graphs. This is the content of the following lemma.

4.24 LEMMA. Let $g \in \mathbb{G}_{\varepsilon\delta}{}^{\infty}$. Then there is a graph $h \in \mathbb{G}_{\varepsilon\delta}$ such that $g \leftrightarrow_\varepsilon h$.
PROOF: The proof is visualised in fig. 8 below: we can replace an infinite branching by a "spine" of $\varepsilon$-steps with the summands branching off consecutively.

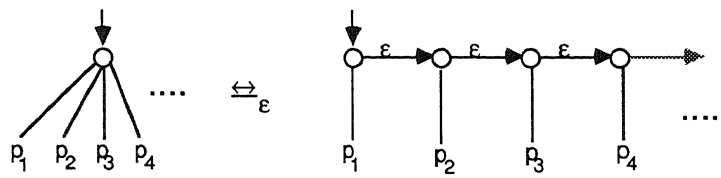


Fig. 8.

4.25 DEFINITION. Now we show that the models $\mathbb{G}_{\varepsilon\delta}/\leftrightarrow_\varepsilon$ and $\mathbb{P}/\leftrightarrow$ are isomorphic. A map from $\mathbb{G}_{\varepsilon\delta}$ to $\mathbb{G}_\downarrow$ is defined in three steps, as follows. Let $g \in \mathbb{G}_{\varepsilon\delta}$.

1. Unroll g to a tree, as defined in 4.17. This gives tree(g).
2. Leave out all intermediate ε-steps, i.e. if $s \to^{\varepsilon} t$ is an edge in tree(g), and t is not an endpoint, then leave out the edge and identify s and t; this gives int(tree(g)). The operation int may cause infinite branchings to appear, so we can only say $int(tree(g)) \in \mathbb{G}_{\varepsilon\delta}^{\infty}$.
3. Leave out all remaining ε-edges, leaving a node-label ↓ for each removed edge, i.e. if $s \to^{\varepsilon} t$ is an edge (so t is an endpoint), remove it and attach a label ↓ to s. Furthermore, attach a label ↓ to each endnode. Then, leave out all δ-edges, and remove all nodes and edges that cannot be reached any more from the root. This gives end(int(tree(g))). Note that $end(int(tree(g))) \in \mathbb{G}_{\downarrow}$.

4.26 LEMMA. 1. Let $g \in \mathbb{G}_{\varepsilon\delta}$. Then: $g \leftrightarrows_{\varepsilon} tree(g) \leftrightarrows_{\varepsilon} int(tree(g))$.
2. Let $g,h \in \mathbb{G}_{\varepsilon\delta}^{\infty}$ with no intermediate ε-edges. Then: $g \leftrightarrows_{\varepsilon} h \Leftrightarrow end(g) \leftrightarrows end(h)$.
PROOF: 1. The first bisimulation is motivated in 4.17. Note that the second only holds in case we are dealing with trees: we must have that an ε-edge does not have any 'neighbours', there must be no other edges between the same two nodes. In case we do have trees, the bisimulation is easy, for the endpoints of ε-steps need not be related at all.
2. This is easy.

4.27 We can conclude from lemma 4.26 that the models $\mathbb{G}_{\downarrow}/\leftrightarrows$ and $\mathbb{G}_{\varepsilon\delta}/\leftrightarrows_{\varepsilon}$ are isomorphic models (since it is not hard to see that the resulting mapping from $\mathbb{G}_{\varepsilon\delta}/\leftrightarrows_{\varepsilon}$ to $\mathbb{G}_{\downarrow}/\leftrightarrows$ is a surjective homomorphism w.r.t. the operators), and thus, $\mathbb{G}_{\varepsilon\delta}/\leftrightarrows_{\varepsilon}$ is a sound and complete model of ACP$\surd$, in which RDP and RSP hold.

4.28 THEOREM. The theory ACP$\surd$ is a conservative extension of the theory ACP of BERGSTRA & KLOP [3], i.e. for all closed ACP-terms s,t we have:

$$ACP\surd \vdash s = t \Leftrightarrow ACP \vdash s = t.$$

PROOF: The theory ACP consists of axioms A1-7 of $PA_{\delta}$ (see 2.3), axioms CF1,2, EM3,4,8 and D1-4 of ACP$\surd$ (see 3.1), the equations in 3.4.3,4,5, axiom EM1 of ACP$\surd$ without the last summand, and the axiom $(x + y) \mid z = x \mid z + y \mid z$. As in 4.5 (using ii. $t \to^{a} \varepsilon$ iff t has a summand a), we can show that ACP is a complete axiomatisation of $\mathbb{P}/\leftrightarrows$ for closed terms, i.e. for closed ACP-terms t,s we have $t \leftrightarrows s$ iff $ACP \vdash t = s$. Together with 4.5, this gives the conservativity.

## 5. LIMIT RULE

In this section, we discuss the Limit Rule, introduced in BAETEN & BERGSTRA [1]. Furthermore, we present the Fresh Atom Principle (FAP), first mentioned in VAANDRAGER [15]. We show that FAP, a restricted version of the Limit Rule, and also the Recursive Specification Principle of 3.15 hold in our models.

5.1 DEFINITION. Let $s(x_1,...,x_n)$, $t(x_1,...,x_n)$ be ACP$\surd$-terms with variables among $x_1,...,x_n$. Let $s(p_1,...,p_n)$, $t(p_1,...,p_n)$ be the terms obtained after substituting $p_1,...,p_n$ for $x_1,...,x_n$, respectively.
Then the Limit Rule reads:
LR: $s(p_1,...,p_n) = t(p_1,...,p_n)$ for all $p_1,...,p_n \in BT \Rightarrow s(x_1,...,x_n) = t(x_1,...,x_n)$.
We leave as an open question whether LR holds in the models of section 4. Next, we formulate a restricted version of LR, LR$^{-}$, that will be shown valid in these models.

5.2 DEFINITION. In order to formulate LR$^{-}$, we should realize that the theory ACP$\surd$ has the set of atomic actions A, and the communication function γ on A, as parameters. Thus, whenever we state that ACP$\surd$ $\vdash p = q$, we mean that for every choice of parameters A and γ (A containing at least the atoms occurring in p,q), we can derive p = q. Also, the models are parametrised by A and γ, so when we state that p = q

holds in a model, we mean that it holds for every choice of parameters. This practice can lead to misunderstandings, however, when we have an implication, as in the Limit Rule. The Limit Rule as stated, means:

for every choice of parameters $A,\gamma$:
if $s(p_1,...,p_n) = t(p_1,...,p_n)$ for all $p_1,...,p_n \in BT$, then $s(x_1,...,x_n) = t(x_1,...,x_n)$.

The restricted version LR$^-$ will have two restrictions: first we will limit ourselves to terms not involving $\varepsilon_K$-operators, and second, we will put the quantification over all parameters in a different place:

Let $s(x_1,...,x_n)$, $t(x_1,...,x_n)$ be ACP$\sqrt{}$-terms without $\varepsilon_K$-operator.
LR$^-$: If for every choice of parameters $A,\gamma$ and for all $p_1,...,p_n \in BT$ we have
$s(p_1,...,p_n) = t(p_1,...,p_n)$,
then (for every choice of parameters $A,\gamma$) $s(x_1,...,x_n) = t(x_1,...,x_n)$.

5.3 DEFINITION. The **Fresh Atom Principle (FAP)** says that we can use new (or 'fresh') atomic actions in proofs. In fact, using FAP (without justification!) is already standard practice in many writings on process algebra. FAP was introduced informally in VAANDRAGER [15], although the name was used earlier by Jan Willem Klop.
Here again, it is important to mention the parameters explicitly.
Suppose we have an atomic action set $A$ and communication function $\gamma$ given. Then we add an atom $f \notin A$, and extend $\gamma$ to $A\cup\{f\}$, yielding $\gamma^*$. Now FAP says, that an equation $p = q$ over the signature with parameters $(A,\gamma)$ may be proved using the parameters $(A\cup\{f\}, \gamma^*)$ in the proof.
Semantically, we can formulate this as follows.
Let $\mathfrak{A}$ be a model of ACP$\sqrt{}$, i.e. for every choice of parameters $A,\gamma$, we have a model $\mathfrak{A}(A,\gamma)$ for the theory ACP$\sqrt{}$ with parameters $A,\gamma$. Now a (parametrised) model $\mathfrak{A}$ satisfies FAP, if such an embedding $(A, \gamma) \to (A\cup\{f\}, \gamma^*)$ can be extended to an injective homomorphism $\mathfrak{A}(A,\gamma) \to \mathfrak{A}(A\cup\{f\},\gamma^*)$.

5.4 PROPOSITION. $\mathbb{P}/\underline{\leftrightarrow}$ satisfies FAP.
PROOF: We have to prove that if $p \underline{\leftrightarrow} q$ in $\mathbb{P}(A\cup\{f\})$ and $p,q$ are process expressions over $A$, then also $p \underline{\leftrightarrow} q$ in $\mathbb{P}(A)$.
So let $p,q$ be process expressions over $A$. Then all action relations starting from $p$ and $q$ have labels from $A$, and all process expressions reachable from $p$ and $q$ are again expressions over $A$. Thus, any bisimulation on $\mathbb{P}(A\cup\{f\})$ relating $p$ and $q$ can be restricted to $\mathbb{P}(A)$, and $R \subseteq \mathbb{P}(A) \times \mathbb{P}(A)$ is a bisimulation over action relations $\{\to^a : a \in A\cup\{f\}\}$, iff $R$ is a bisimulation over action relations $\{\to^a : a \in A\}$.

5.5 LEMMA. Let $f$ be an atomic action such that for all $a,b \in A$ $\gamma(a,f)\uparrow$ (is not defined) and $\gamma(a,b) \neq f$.
Then:
$\varepsilon_{\{f\}}(x \square y) = \varepsilon_{\{f\}}(x) \square \varepsilon_{\{f\}}(y)$ for $\square = +,\cdot,\|,\lfloor\!\lfloor,|$, and $\varepsilon_{\{f\}}(\partial_H(x)) = \partial_H(\varepsilon_{\{f\}}(x))$ for $H \subseteq A$.
PROOF: Straightforward.

5.6 DEFINITION. In order to show that LR$^-$ and RSP hold in $\mathbb{P}/\underline{\leftrightarrow}$ we need some auxiliary notions, that may also be interesting in their own right. First we define the **projections** of a process. To that end, we enlarge the signature of ACP$\sqrt{}$ with unary operators $\pi_n$, for $n \in \mathbb{N}$. Then we add the axioms PR (for $a \in A\cup\{\delta\}$) (on the following page).

| |
|---|
| $\pi_n(\varepsilon) = \varepsilon$ |
| $\pi_0(ax) = \delta$ |
| $\pi_{n+1}(ax) = a \cdot \pi_n(x)$ |
| $\pi_n(x + y) = \pi_n(x) + \pi_n(y)$ |

Table 6. Projection.

We see that the operator $\pi_n$ cuts off the process after it has executed n (atomic) steps; the remaining steps are replaced by $\delta$. In order to define the operators $\pi_n$ on the models $\mathbb{P}/\leftrightarrow$ and $\mathbb{G}\downarrow/\leftrightarrow$, we provide the following action rules for $\pi_n$:

$$x \to^a x' \Rightarrow \pi_{n+1}(x) \to^a \pi_n(x') \qquad x\downarrow \Rightarrow \pi_n(x)\downarrow.$$

It is easy to check that $\leftrightarrow$ remains a congruence on $\mathbb{P}$ and $\mathbb{P}/\leftrightarrow$ satisfies the axioms of table 6.

5.7 PROPOSITION. The following equations are derivable for closed ACP$\sqrt{}$-terms. Moreover, they hold in $\mathbb{P}/\leftrightarrow$.

1. $\pi_n(x \mathbin{\square} y) = \pi_n(\pi_n(x) \mathbin{\square} \pi_n(y))$ for $\square = +, \cdot, \|, \lfloor\!\lfloor, \mid$.
2. $\pi_n(\partial_H(x)) = \partial_H(\pi_n(x))$ for $H \subseteq A$.

PROOF: Straightforward, using one of the inductive schemes in 3.7. Note that the analogous statement for the operator $\varepsilon_K$ does *not* hold.

5.8 DEFINITION. The process $g \in \mathbb{G}\downarrow/\leftrightarrow$ is **finitely branching** if there is a finitely branching graph in its equivalence class. Since $\mathbb{P}/\leftrightarrow$ and $\mathbb{G}_{\varepsilon\delta}/\leftrightarrow_\varepsilon$ are isomorphic to $\mathbb{G}\downarrow/\leftrightarrow$, this property carries over to the other models.

5.9 PROPOSITION. The domain of finitely branching processes (inside one of our models) is closed under the operators $+, \cdot, \|, \lfloor\!\lfloor, \mid, \partial_H$, but *not* under $\varepsilon_K$.

PROOF: Straightforward.

5.10 PROPOSITION. Let $p \in \mathbb{P}/\leftrightarrow$ be finitely branching, and let $n \in \mathbb{N}$. Then there is a basic term $q_n$ such that

$\pi_n(p) = q_n$ holds in $\mathbb{P}/\leftrightarrow$.

PROOF: This is easiest to see in the model $\mathbb{G}\downarrow/\leftrightarrow$, and transfers by isomorphism to $\mathbb{P}/\leftrightarrow$.

5.11 PROPOSITION. For any process $p \in \mathbb{P}(A)/\leftrightarrow$, and fresh atom $f \notin A$, there is a finitely branching process $q \in \mathbb{P}(A \cup \{f\})/\leftrightarrow$ such that $\varepsilon_{\{f\}}(q) = p$.

PROOF: This follows by considering fig. 8 in 4.24: replace every infinite branching by a spine of f-steps to obtain q; renaming f into $\varepsilon$ gives a process that bisimulates with p.

5.12 DEFINITION. The **Restricted Approximation Induction Principle (AIP$^-$)** says that a finitely branching process is completely determined by its finite projections, i.e. if p is finitely branching, and q is such that $\pi_n(p) = \pi_n(q)$ for all n, then $p = q$.

The "–" refers to a version of AIP without the restriction to finitely branching processes. For more information on AIP$^-$, see VAN GLABBEEK [7].

5.13 THEOREM. AIP$^-$ holds in $\mathbb{P}/\leftrightarrow$.

PROOF: As in VAN GLABBEEK [7]. There, a version of AIP$^-$ is used, which is less restrictive, with bounded processes instead of finitely branching processes. It is easy to see that a finitely branching processes is bounded in the sense of [7].

5.14 THEOREM. The Recursive Specification Principle RSP holds in $\mathbb{P}/\leftrightarrow$.

PROOF: Let $E$ be a guarded recursive specification over variables $V$, and $X \in V$. Since all variables have a head normal form (3.15), process $p = \langle X \mid E \rangle / \leftrightarrows$ is finitely branching. By 5.10, the finite projections of $p$ are equal to basic terms. But it is easy to see that these basic terms only depend on the equations in $E$, and not on the particular solution. So, any solution must have the same finite projections, and hence is equal to $p$ by AIP$^-$.

5.15 THEOREM. The Restricted Limit Rule LR$^-$ holds in $\mathbb{P}/\leftrightarrows$.
PROOF: Let $s(x_1,...,x_n)$, $t(x_1,...,x_n)$ be ACP$\sqrt{}$-terms without $\varepsilon_K$-operator such that for every choice of parameters $A,\gamma$, and any $p_1,...,p_n \in BT$ we have $s(p_1,...,p_n) = t(p_1,...,p_n)$ holds in $\mathbb{P}(A)/\leftrightarrows$. We have to show that $s(x_1,...,x_n) = t(x_1,...,x_n)$ holds in $\mathbb{P}/\leftrightarrows$. This is the case if for every choice $A,\gamma$ we have $s(p_1,...,p_n) \leftrightarrows t(p_1,...,p_n)$ for all $p_1,...,p_n \in \mathbb{P}(A)$.
So let $A,\gamma$ be given and suppose $p_1,...,p_n \in \mathbb{P}(A)$. Let $f$ be a fresh atom. Choose (using 5.11) finitely branching $q_1,...,q_n \in \mathbb{P}(A \cup \{f\})$ such that $\varepsilon_{\{f\}}(q_i) = p_i$ $(1 \leq i \leq n)$. For each $k \in \mathbb{N}$, choose (using 5.10) basic terms $r_1^k,...,r_n^k$ such that $\pi_k(q_i) = r_i^k$. Now by 5.7 we have for each $k \in \mathbb{N}$

$$\pi_k(s(q_1,...,q_n)) = \pi_k(s(\pi_k(q_1), ..., \pi_k(q_n))) = \pi_k(s(r_1^k,...,r_n^k)) =$$
$$= \pi_k(t(r_1^k,...,r_n^k)) = \pi_k(t(q_1,...,q_n)).$$

Thus, by AIP$^-$ and 5.9, we have $s(q_1,...,q_n) = t(q_1,...,q_n)$, and hence, by 5.5,

$$s(p_1,...,p_n) = s(\varepsilon_{\{f\}}(q_1), ..., \varepsilon_{\{f\}}(q_n)) = \varepsilon_{\{f\}}(s(q_1,...,q_n)) = \varepsilon_{\{f\}}(t(q_1,...,q_n)) = t(p_1,...,p_n).$$

## REFERENCES


[1] J.C.M.BAETEN & J.A.BERGSTRA, *Global renaming operators in concrete process algebra,* report CS-R8521, Centre for Math. & Comp. Sci., Amsterdam 1985. To appear in Inf. & Computation.
[2] J.A.BERGSTRA & J.W.KLOP, *Fixed point semantics in process algebras,* report IW 206, Mathematical Centre, Amsterdam 1982.
[3] J.A.BERGSTRA & J.W.KLOP, *Process algebra for synchronous communication,* Inf. & Control 60 (1/3), pp. 109 - 137, 1984.
[4] J.A.BERGSTRA & J.W.KLOP, *Algebra of communicating processes,* in: Proc. CWI Symp. Math. & Comp. Sci. (J.W.de Bakker, M.Hazewinkel & J.K.Lenstra, eds.), pp. 89 - 138, North- Holland, Amsterdam 1986.
[5] J.A.BERGSTRA, J.W.KLOP & E.-R. OLDEROG, *Failures without chaos: a new process semantics for fair abstraction,* in: Proc. IFIP Conf. on Formal Description of Programming Concepts - III, Ebberup 1986, (M.Wirsing, ed.), North-Holland, Amsterdam, pp. 77 - 103, 1987.
[6] J.A.BERGSTRA & J.V.TUCKER, *Top down design and the algebra of communicating processes,* Sci. of Comp. Progr. 5 (2), pp. 171 - 199, 1985.
[7] R.J.VAN GLABBEEK, *Bounded nondeterminism and the approximation induction principle in process algebra,* in: Proc. STACS 87 (F.J.Brandenburg, G.Vidal-Naquet & M.Wirsing eds.), Springer LNCS 247, pp. 336 - 347, 1987.
[8] C.A.R.HOARE, *Communicating sequential processes,* Prentice Hall 1985.
[9] C.P.J.KOYMANS & J.L.M.VRANCKEN, *Extending process algebra with the empty process ε,* report LGPS 1, Dept. of Philosophy, State University of Utrecht, The Netherlands 1985.
[10] R.MILNER, *A calculus of communicating systems,* Springer LNCS 92, 1980.
[11] R.MILNER, *Lectures on a calculus of communicating systems,* in: Seminar on concurrency (S.D.Brookes, A.W.Roscoe & G.Winskel, eds.), Springer LNCS 197, pp. 197 - 220, 1985.
[12] D.M.R.PARK, *Concurrency and automata on infinite sequences,* in: Proc. 5th GI Conf. (P.Deussen, ed.), Springer LNCS 104, pp. 167 - 183, 1981.
[13] G.PLOTKIN, *An operational semantics for CSP,* in: Proc. Conf. Formal Description of Progr. Concepts II (D.Bjørner, ed.), pp. 199 - 223, North-Holland, Amsterdam 1982.
[14] J.L.M.VRANCKEN, *The algebra of communicating processes with empty process,* report FVI 86-01, Dept. of Comp. Sci., Univ. of Amsterdam 1986.
[15] F.W.VAANDRAGER, *Process algebra semantics of POOL,* report CS-R8629, Centre for Math. & Comp. Sci., Amsterdam 1986.