MATHEMATICAL PROBLEMS IN COMPUTATION THEORY
BANACH CENTER PUBLICATIONS, VOLUME 21
PWN – POLISH SCIENTIFIC PUBLISHERS
WARSAW 1988

AN OPERATIONAL SEMANTICS FOR PROCESS ALGEBRA

J. C. M. BAETEN

Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

J. A. BERGSTRA

University of Amsterdam, State University of Utrecht

J. W. KLOP

Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

We consider operational rewrite rules, expressing the dynamic behaviour of configurations of objects. This gives an algebraic way to generalize Petri net theory. We give a detailed description of operational rewrite rules for various process algebras, and show that in each case, the operational semantics thus obtained, is equivalent to the regular denotational semantics of processes. This theory can be used to describe features like true concurrency and real-time behaviour for concurrent, communicating processes.

Introduction

Previous papers on process algebra (see e.g. Bergstra and Klop [6], [7]) give semantics for such algebras that are mainly denotational in nature, such as the model of projective sequences, the initial algebra and (to a lesser extent) the graph model. In this paper, we present a more operational semantics for

---

1980 Mathematics Subject Classification: 68B10, 68C01, 68D25, 68F20.

1982 CR Categories: F.1.1, F.1.2, F.3.2, F.4.3.

Key Words. Phrases. *Process algebra, concurrency, operational semantics, true concurrency, real-time behaviour, rewrite rules, object-oriented, Petri net theory.*

Note. Partial support received from the European Communities under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

process algebra. We do this with the use of *operational rewrite rules* (called transformation rules in Bergstra, Heering and Klop [5], where they were introduced), that adjoined to an algebra (mostly the initial algebra of a set of equations over a signature) give an *operational rewrite system* or object-oriented algebraic specification. We remark that operational rewrite rules can be used to provide an algebraisation of Petri nets, so that any concurrency issue that can be discussed with the help of Petri nets, can also be discussed in an algebraical setting with the help of operational rewrite rules. Then we proceed to define in detail operational rewrite rules for various process algebras, namely BPA (Basic Process Algebra), PA (Process Algebra, with a merge operator, without communication, with or without a constant for deadlock) and ACP (Algebra of Communicating Processes), and discuss in addition a constant ε (different from τ!) denoting the empty process. In each case, we prove in a precise way that the operational behaviour of a process is equal to its denotational behaviour. For this proof, it is necessary, to first rename the internal operational transformation steps into ε, so to abstract from these steps.

For several other '$\mathcal{O} = \mathcal{M}$ proofs' (operational semantics = denotational semantics) see De Bakker [4].

Our operational rewrite rules for algebras with merge allow us to put processes in parallel, so that at each point we are dealing with a multiset of objects (or terms), called a configuration, and because operational rewrite rules can work simultaneously on disjoint subconfigurations, we can execute processes in parallel, and discuss issues like true concurrency and real-time behaviour. In the last section, we give as an example an explicit real-time semantics for concurrent, communicating processes. To illustrate this semantics, we give an efficient parallel implementation of a four-bit buffer, that works twice as fast as a sequential implementation.

Table of contents

1. Operational rewrite rules

1.1. Most of the notations discussed in this paragraph were introduced in Bergstra, Heering and Klop [5], where also more detailed information can be found.

Let Σ be a many-sorted algebraic signature, and let $A \in \mathrm{Alg}(\Sigma)$ be an

algebra of signature Σ (often called an *abstract data type*). Elements of A are called *points* or *objects*. A multi-set of objects we call a *configuration*. Configurations exhibit dynamic behaviour, may perform *transformation steps*

$$C \underset{R}{\rightarrow} C'$$

Transformation steps are instantiations of *transformation rules*, or *operational rewrite rules*. We will always have that $A = T_I(\Sigma, E)$, i.e., A is the initial algebra of a set of equations E over Σ. If R is a set of operational rewrite rules, we call $((\Sigma, E), R)$ an *operational rewrite system* (*ORS*) or an *object-oriented algebraic specification*.

1.2. An *operational rewrite rule* is a notation of the form

$$\textit{rule name}\left(\frac{\textit{configuration before transformation}}{\textit{configuration after transformation}}\right)$$

or $r(C/C')$. Here C, C' are finite multisets from $T_\Sigma(X)$ (the set of terms over Σ with variables from some set of variables X). Actually, we should consider terms modulo the equality relation generated by E (which correspond to elements of the initial algebra). Now if C_1, C'_1, C_2 are finite multisets from T_Σ (the set of closed Σ-terms), so that there is a valuation (a mapping from variables X to closed Σ-terms T_Σ) such that applying the valuation to the elements of C will result in the elements of C_1 (up to term equality), and applying the valuation to C' gives C'_1, then

$$C_1 \cup C_2 \underset{r}{\rightarrow} C'_1 \cup C_2$$

is a *transformation step* (or *step* for short).

1.3. It can be noted that operational rewrite rules give a generalization of Petri net theory (a general reference to Petri net theory is Reisig [11]). Another algebraisation of Petri nets is given in Boudol, Roucairol and De Simone [8], where this issue is discussed in more detail.

To illustrate this remark, if a net has a set of places P, and a set of markers M (we allow different kinds of marker), and $O(m, p)$ is the statement that marker m is at place p, then a firing is given by an operational rewrite rule of the form

$$\textit{rulename}\left(\frac{O(m_1, p_1), \ldots, O(m_n, p_n)}{O(m'_1, p'_1), \ldots, O(m'_k, p'_k)}\right)$$

(with m_i, $m'_j \in M$ and p_i, $p'_j \in P$ for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant k$).

EXAMPLE. Let a Petri net be given by Fig. 1.

The places are named as indicated, and we have only one kind of marker, so we can omit the mention of markers in our rules.

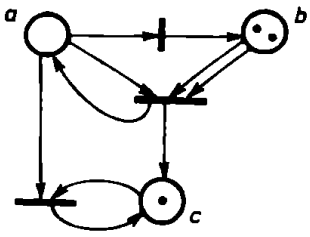


Fig. 1

The net is given by:

$$r_1\left(\frac{O(a)}{O(b)}\right),\ r_2\left(\frac{O(a),\ O(b),\ O(b)}{O(a),\ O(c)}\right),\ r_3\left(\frac{O(a),\ O(c)}{O(c)}\right),$$

and the configuration displayed is $\{O(b),\ O(b),\ O(c)\}$.

2. Basic process algebra

2.1. Now we start the investigation of operational rewrite rules in process algebra, an algebraic theory of concurrent, communicating process, described in Bergstra and Klop [7]. In order to modularise our investigation, and as a simple first illustration of the use of operational rewrite rules, we consider *Basic Process Algebra* or BPA (see Bergstra and Klop [6]). BPA starts with a finite collection of atomic processes and has two binary operators, namely $+$, denoting alternative composition, (nondeterministic) choice or sum, and $\cdot$, denoting sequential composition. Furthermore, we have a constant ε, denoting the empty process and successful termination, that will simplify the algebraic description. The constant ε was introduced in Koymans and Vrancken [10].

The sort of atomic actions, A, is a subsort of the sort of processes, P. For more information about specifications and initial algebras using subsorts, see Goguen [9].

2.2. Definition. $BPA_\varepsilon = (\Sigma_{BPA\varepsilon}, E_{BPA\varepsilon})$, where the signature $\Sigma_{BPA\varepsilon}$ is given by:

$$\Sigma_{BPA\varepsilon}: \begin{cases} S \text{ (Sorts):} & A,\ P, \text{ with } A \subseteq P, \\ F \text{ (Functions):} & + \ : P \times P \to P, \\ & \cdot \ : P \times P \to P, \\ C \text{ (Constants):} & \varepsilon \in P - A, \end{cases}$$

and the set of equations E_{BPA_ε} is given in Table 1.

$x+y = y+x$
$(x+y)+z = x+(y+z)$
$x+x = x$
$(x+y)z = xz+yz$
$(xy)z = x(yz)$
$\varepsilon x = x$
$x\varepsilon = x$

Table 1

Here x, y, z are variables ranging over P, called process variables. We call a BPA_ε-term closed if it contains no process variables. Such a term may contain atomic steps though (variables ranging over A). Note that the axiom $x+x = x$ is equivalent to $\varepsilon+\varepsilon = \varepsilon$, for $x+x = \varepsilon x+\varepsilon x = (\varepsilon+\varepsilon)x = \varepsilon x = x$, and the other direction is immediate. We do not explicitly require that A be finite, although that will be the case in all applications. Using these axioms, we can show that each closed term can be written in the form ε, at or t_1+t_2, with $a \in A$ and t, t_1, t_2 closed terms. Induction arguments (and recursive definitions) will usually consider these three cases.

2.3. Now we consider the dynamic properties of processes given by BPA_ε-terms. We will have three operational rewrite rules, corresponding to three basic actions of a process:

- performing (executing) an atomic action (an $a \in A$);
- making a nondeterministic choice;
- terminating (modelled by ε).

Specifically, we have the rules

$$a\left(\frac{ax}{x}\right) \quad (\text{for } a \in A), \qquad i+\left(\frac{x+y}{x}\right), \qquad \varepsilon\left(\frac{\varepsilon}{-}\right).$$

Note that when we give operational rewrite rules, we display a configuration $C = \{c_1, \ldots, c_n\}$ by writing $c_1, \ldots, c_n$, and we display the empty configuration $\emptyset$ by leaving an empty space. The rules of Meije (see Austry and Boudol [1]) are similar, and can also be considered as operational rewrite rules.

2.4. Examples. Let $x \in T_\Sigma$, a, b, $c \in A$. Then

1. $\{x\} \xrightarrow[i+]{} \{x\}$ for $x = x+x$;

2. $\{a\} \xrightarrow[a]{} \{\varepsilon\} \xrightarrow[\varepsilon]{} \emptyset$ for $a = a\varepsilon$;
3. $\{a+b\} \xrightarrow[i+]{} \{b\}$ for $a+b = b+a$;
4. $\{(a+b)c\} \xrightarrow[i+]{} \{ac\}$ for $(a+b)c = ac+bc$.

2.5. As we saw in 2.4, many transformation sequences are possible from a given closed Σ-term. This leads us to consider *transformation graphs*. Before giving a precise definition for a general operational rewrite system, we give some examples for BPA_ε with the rules of 2.3.

EXAMPLES. Let a, b, $c \in A$, and distinct. Term $a(b+c)$ has the transformation graph in Fig. 2a, term $ab+ac$ has the graph in 2b.

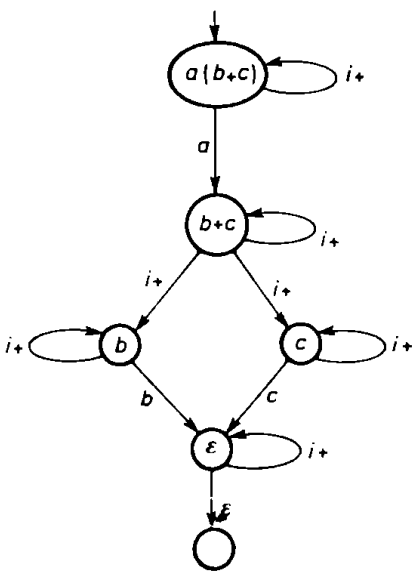


Fig. 2a

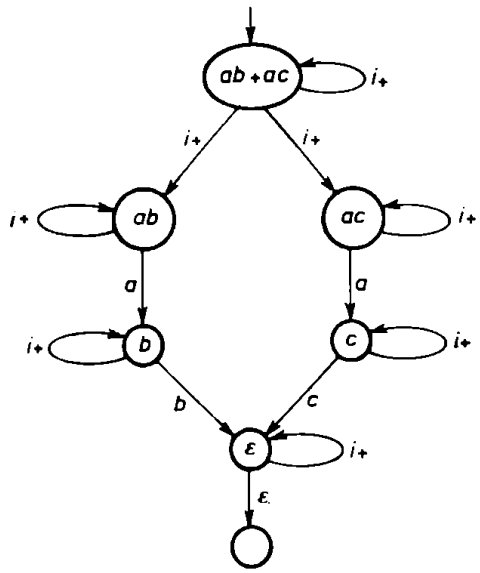


Fig. 2b

2.6. DEFINITION. Let $((\Sigma, E), R)$ be an operational rewrite system. Let G be the set of all rooted, directed, finitely branching, labelled multigraphs. By the previous sentence we mean that each $g \in G$ has a set of nodes and a set of edges (with each edge going from one node to another) and a root (an element of the set of nodes). Furthermore, each node in g has finitely many edges going from it, there can be more than one edge between two nodes, and each edge has a label. We identify graphs that only differ in the names of nodes and edges. For more information, see Baeten, Bergstra and Klop [3]. In this case, we take as the set of labels the set R of names of operational rewrite rules. Now define a map

$$\varphi\colon T_\Sigma \to G$$

as follows: if $t \in T_\Sigma$, then the set of nodes of $\varphi(t)$ is the set of all configurations, reachable by transformation steps from configuration $\{t\}$, the root is given by $\{t\}$, and if, for two nodes C, C^1 in $\varphi(t)$, there is a transformation step $C \xrightarrow[r]{} C^1$ $(r \in R)$, then there is an edge in $\varphi(t)$ from C to C^1

with label r. We call $\varphi(t)$ the *transformation graph* of t. Two examples for the *ORS* $(BPA_\varepsilon,\ A \cup \{\varepsilon, i+\})$ are given in 2.5.

Next, if $I \subseteq R$, we define a map

$$\varepsilon_I\colon\ G \to G$$

as follows: if $g \in G$, we obtain $\varepsilon_I(g)$ from g by changing all labels from I to ε.

Finally, we define

$$T_I = \{\varepsilon_I \circ \varphi(t)\colon\ t \in T_\Sigma\},$$

the set of *I-abstracted transformation graphs*. We will prove that for

$$(BPA_\varepsilon,\ A \cup \{\varepsilon, i+\}),$$

$T_{\{i+\}}$ is isomorphic to the initial algebra of BPA_ε, by considering the standard model $G/\leftrightarrows_\varepsilon$.

2.7. The standard model. $G/\leftrightarrows_\varepsilon$ of BPA_ε, obtained from G (without a label $i+$) by dividing out the congruence relation $\leftrightarrows_\varepsilon$ (ε-bisimulation) was introduced in Koymans and Vrancken [10]. In this article, much information about ε-bisimulation can be found. Here, we just give the basic definitions.

2.8. Definition. Let $g,\ h \in G$. We say that g and h are *ε-bisimilar*, notation $g \leftrightarrows_\varepsilon h$ iff there is a relation R between nodes of g and nodes of h such that:

1. the roots are related;

2. for each node n_1 in g and each node n_2 in h, if $n_1 R n_2$, and, starting from n_1, we can do a number of ε-steps (possibly 0) followed by a step $a \in A$ to a node n_1', then there is a node n_2' in h such that $n_1' R n_2'$ and, starting from n_1, we can do a number of ε-steps (possibly 0) followed by a step a to n_2';

3. for each node n_1 in g and each node n_2 in h, if $n_1 R n_2$, and, starting from n_1, we can do a number of ε-steps to an endnode, then, starting from n_2, we can do a number of ε-steps to an endnode;

4, 5 same as 2, 3, but with the roles of g and h interchanged.

2.9. Examples. $(a,\ b \in A)$

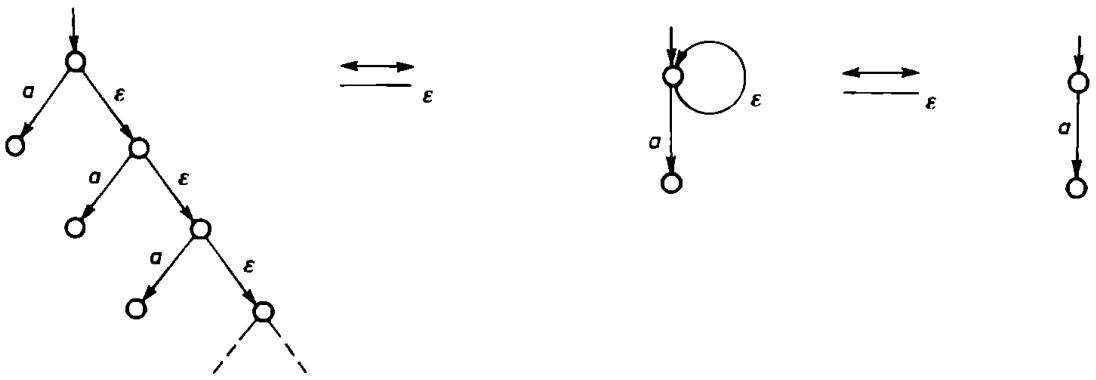


Fig. 3

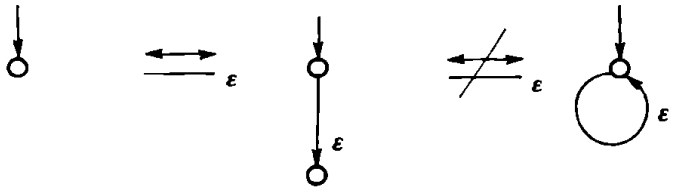


Fig. 4

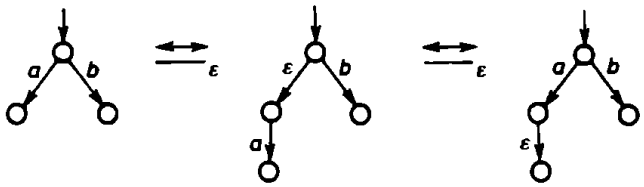


Fig. 5

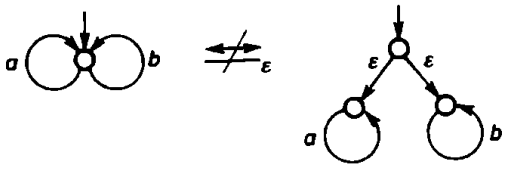


Fig. 6

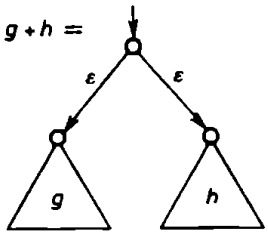
2.10. THEOREM. $G/\leftrightarrows_\varepsilon \models BPA_\varepsilon$.

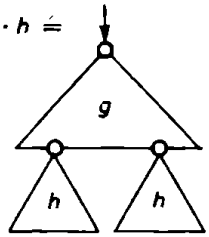
Proof. We define an interpretation from BPA_ε into $G/\leftrightarrows_\varepsilon$ as follows:

1. for each $a \in A$, $[a] =$ 

2. $[\varepsilon] =$ 

3. we define operations $+$, $\cdot$ on G as follows: if g, $h \in G$, then

$g+h =$ 

and $g \cdot h =$ 

(append h to each endnode of g);

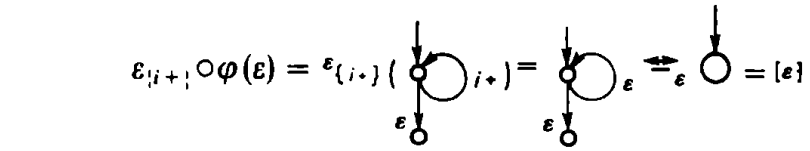
these operations are the interpretations of the functions $+$, $\cdot$ of BPA_ε. The rest of the proof can be found in Koymans and Vrancken [10].

2.11. THEOREM. *For each closed BPA_ε-term t we have*

$$\varepsilon_{\{i+\}} \circ \varphi(t) \leftrightarrows_\varepsilon [t].$$

Proof. We use induction on the structure of t.

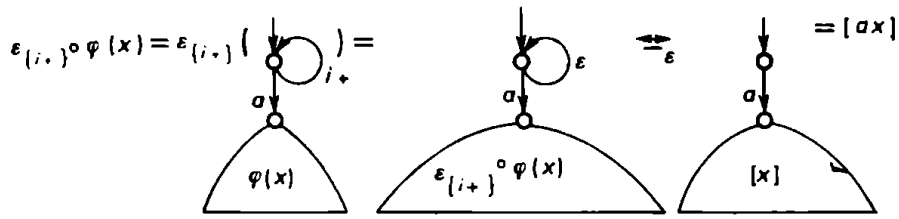
1. $t \equiv \varepsilon$. Then



2. $t \equiv ax$, with $a \in A$, and we know by induction hypothesis that

$$\varepsilon_{\{i+\}} \circ \varphi(x) \leftrightarrows_\varepsilon [x].$$

Then



3. t is a sum of terms, and for each summand x we know by induction hypothesis that $\varepsilon_{\{i+\}} \circ \varphi(x) \leftrightarrows_\varepsilon [x]$. We write $t = \sum_{k=1}^{n} x_k$, where each x_k is not a sum itself (so of the form ax' or ε) and all x_k are different. For simplicity, we shall take $n = 3$ (the general proof is similar). Then, $\varphi(t)$ is displayed in Fig. 7a, and it is not hard to see that $\varepsilon_{\{i+\}} \circ \varphi(t)$ ε-bisimulates with $[t]$, displayed in Fig. 7b.

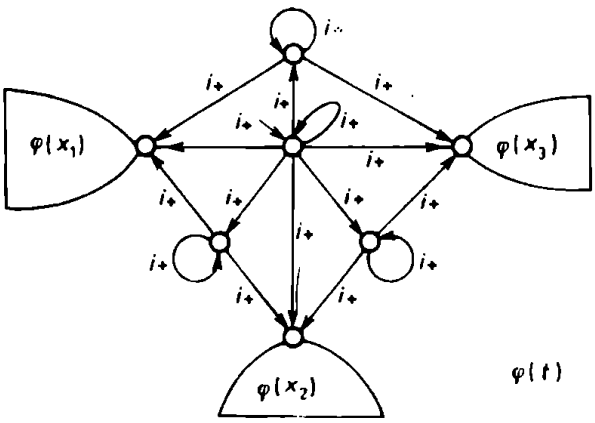


Fig. 7a

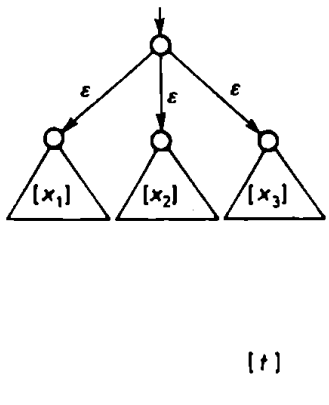


Fig. 7b

(*Note.* Graphs $\varphi(x_1)$, $\varphi(x_2)$, $\varphi(x_3)$ will share nodes.)

2.12. COROLLARY. *$T_{\{i+\}} \models BPA_\varepsilon$, if on $T_{\{i+\}}$ we use the operations $+$, $\cdot$, induced by the $+$, $\cdot$ on BPA_ε-terms.*

Furthermore, we can observe that $T_{\{i+\}}$ is in fact isomorphic to the initial algebra of BPA_ε.

2.13. *Remark.* The proof of 2.11 is semantical in nature, takes place in the model $G/\leftrightarrows_\varepsilon$. Next we will formulate a translation, back from G into the algebra, and give a syntactic proof of 2.11, and extend it at the same time. First we need some additional algebraical machinery.

2.14. DEFINITION. Let (Σ, E) be an equational theory, and X a set of variables. A *recursive specification* is a set of equations $F = \{x = t_x\colon x \in X\}$, with $t_x \in T_\Sigma(X)$. If $x_0 \in X$ is some specified variable, $\mathfrak{A}$ a Σ-algebra and a_0 in $\mathfrak{A}$, we say that a_0 is a *solution of* F (*in* $\mathfrak{A}$ *for* x_0) iff equations F hold in $\mathfrak{A}$, substituting a_0 for x_0 (and other elements from $\mathfrak{A}$ for the other elements of X). (For more information about recursive specifications, see Baeten, Bergstra and Klop [3].)

2.15. DEFINITION. Let $F = \{x = t_x\colon x \in X\}$ be a recursive specification over BPA_ε. An occurrence of a variable y in a right-hand side t_x is called *guarded* iff t_x has a subterm of the form aM, with $a \in A$ and y occurring in M, *unguarded* otherwise. Note that the term εx is unguarded. We call specification F *guarded* iff there is not an infinite sequence $x_0, x_1, x_2, \ldots$, in X such that for each $n \in N$, x_{n+1} occurs unguarded in t_{x_n}.

2.16. THEOREM. *Each guarded recursive specification over* BPA_ε *has a unique solution in* $G/\leftrightarrows_\varepsilon$.

Proof. Adapt the proof in Baeten, Bergstra and Klop [3] to the setting without τ, but with ε.

2.17. DEFINITION. Theorem 2.16 leads us to formulate two algebraic principles, *RDP* and *RSP*. *RDP* is the *Recursive Definition Principle*, which says that for each guarded recursive specification there is a process satisfying its equations. *RSP* is the *Recursive Specification Principle*, which says that for each guarded recursive specification there is at most one process satisfying its equations.

2.18. EXAMPLE. 1. Let F be $x = ax$ (so $X = \{x\}$). There is, by *RDP* $+$ *RSP*, exactly one process satisfying this equation.

We call this process a^ω, and could consider a^ω as a new constant, added to the signature ($a^\omega \in P$).

2. Let F be $\{x = ay,\ y = ax\}$. By *RSP* $x = y$, so by 1. $x = a^\omega$ and $y = a^\omega$.

2.19. DEFINITION. Now, with each $g \in G$ we will associate a recursive specification F, which has g as its unique solution. First, as the set of variables we take the set of nodes of g which are not endpoints; say this set is X and the root has name x_0 (in case $g \equiv \circ$, take $g' \equiv \to\circ \xrightarrow{\varepsilon} \circ$ instead, a

bisimulating graph). Then, if $x \in X$ has edges starting from it with labels $a_1, \ldots, a_n$ to nodes $x_1, \ldots, x_n$ and edges $b_1, \ldots, b_m$ to endpoints $(n+m > 0)$, the equation for x is

$$x = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n + b_1 + b_2 + \ldots + b_m.$$

If the recursive specification F consisting of these equations is guarded, then it can be shown that g is the unique solution of F in G for x_0 (up to ε-bisimulation).

2.20. Definition. Now we extend BPA_ε by adding a renaming operator ε_I. More specifically,

$$BPA_\varepsilon + = (\Sigma_{BPA_\varepsilon +}\ E_{BPA_\varepsilon +}),$$

where

$$\Sigma_{BPA_\varepsilon +} = \Sigma_{BPA_\varepsilon} \cup \begin{cases} S: \\ F:\ \varepsilon_I:\ P \to P \quad (\text{where } I \subseteq A) \\ C: \end{cases}$$

and

$$E_{BPA_\varepsilon +} = E_{BPA_\varepsilon} \cup \boxed{\begin{array}{l} \varepsilon_I(\varepsilon) = \varepsilon \\ \varepsilon_I(a) = a \text{ if } a \notin I \\ \varepsilon_I(a) = \varepsilon \text{ if } a \in I \\ \varepsilon_I(x+y) = \varepsilon_I(x) + \varepsilon_I(y) \\ \varepsilon_I(xy) = \varepsilon_I(x)\varepsilon_I(y) \end{array}}$$

Table 1

$$\boxed{\dfrac{\text{for each } n\ x_n = i_n x_{n+1} + y_n\ (i_n \in I)}{\varepsilon_I(x_0) = \sum_{n=0}^{\infty} \varepsilon_I(y_n)} \quad EAR}$$

Table 2

here a, $i_n \in A$ (for $n \in N$); x, x_n, y, $y_n \in P$ (for $n \in N$). The first five equations define ε_I on the initial algebra of closed terms: elements of $I \subseteq A$ are changed to ε, other elements are unchanged. These equations enable us to eliminate the operator ε_I from any closed term.

The last equation is a conditional equation, the Epsilon Abstraction Rule (EAR), and is needed to deal with (solutions of) recursive specifications (infinite processes, as opposed to the closed terms, which are finite processes, and need to terminate in finitely many steps). Note that all these axioms are true in $G/\leftrightarrows_\varepsilon$. Heuristically, we see $I \subseteq A$ as the set of *internal steps*, steps we do not want to see when viewing the process from the outside. We abstract

from steps in I by using ε_I, renaming them into ε. A drawback of this ε-abstraction is that the choice-structure of a process is not preserved. By this we mean the following: process $a(ib+c)$ (a, i, b, $c \in A$) has a point where b is possible but c is not, i.e., the tree in Fig. 8a has a node with an outgoing edge labelled b but no outgoing edge labelled c; while process $\varepsilon_{\{i\}}(a(ib+c)) = a(b+c)$ has no such node. This drawback is not present in τ-abstraction, which is not dealt with in this paper (EAR is the 'ε-analogon' of the τ-rule Koomen's Fair Abstraction Rule, see Baeten, Bergstra and Klop [3]).

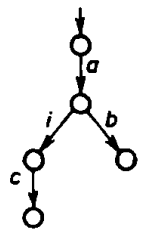


Fig. 8a

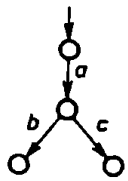


Fig. 8b

2.21. THEOREM. *In $BPA_\varepsilon+$, we have for each closed term t: if F is the specification belonging to the graph $\varphi(t)$, and x_0 corresponds (in F) to the root of $\varphi(t)$, then $\varepsilon_{\{i+\}}(x_0) = t$.*

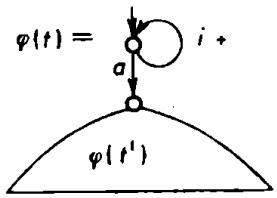
Proof. Note that specification F contains an extra atom $i+$, and that F in each case will be a guarded recursive specification. We prove the theorem by induction on t.

1. $t \equiv \varepsilon$. Then

$\varphi(t) =$ 

so F is $x_0 = (i+)x_0 + \varepsilon$, and by EAR $\varepsilon_{\{i+\}}(x_0) = \varepsilon$.

2. $t \equiv at'$, and suppose the theorem holds for t' ($a \in A$). Then



so F is

$$\begin{cases} x_0 = (i+)x_0 + ax_1, \\ \text{equations } F' \text{ belonging to } \varphi(t') \text{ with variable } x_1 \text{ for the root.} \end{cases}$$

By induction hypothesis we have $\varepsilon_{\{i+\}}(x_1) = t'$, so, by applying EAR

$$\varepsilon_{\{i+\}}(x_0) = \varepsilon_{\{i+\}}(ax_1) = a\varepsilon_{\{i+\}}(x_1) = at' = t.$$

3. t is a sum of terms, say $t = \sum_{k=1}^{n} t_k$, where each t_k is not a sum, and suppose the theorem holds for each t_k. The specification F will have a variable x_σ for each subset σ of $\{1, \ldots, n\}$, with rootvariable $x_{\{1,\ldots,n\}}$. Look at Fig. 7a for the case $n = 3$. F consists of the following equations:

$$\begin{cases} x_\varrho = \sum_{\substack{\sigma \subseteq \varrho \\ \sigma \neq \emptyset}} (i+)\, x_\sigma \text{ for each } \varrho \subseteq \{1, \ldots, n\} \text{ with } |\varrho| \geqslant 2, \\ \text{equations } F_k \text{ for } t_k, \text{ with variable } x_{\{k\}} \text{ for the root (for each } 1 \leqslant k \leqslant n). \end{cases}$$

By induction hypothesis $\varepsilon_{\{i+\}}(x_{\{k\}}) = t_k$, if $1 \leqslant k \leqslant n$, and so, applying EAR to equation

$$x_{\{k,l\}} = (i+)\, x_{\{k,l\}} + (i+)\, x_{\{k\}} + (i+)\, x_{\{l\}} \qquad (1 \leqslant k < l \leqslant n),$$

we obtain

$$\varepsilon_{\{i+\}}(x_{\{k,l\}}) = \varepsilon_{\{i+\}}\big((i+)\, x_{\{k\}} + (i+)\, x_{\{l\}}\big) = \varepsilon_{\{i+\}}(x_{\{k\}}) + \varepsilon_{\{i+\}}(x_{\{l\}}) = t_k + t_l.$$

Similarly, since

$$x_{\{k,l,m\}} = (i+)\, x_{\{k,l,m\}} + (i+)\, x_{\{k,l\}} + (i+)\, x_{\{k,m\}} + (i+)\, x_{\{l,m\}} + (i+)\, x_{\{k\}} + \\ + (i+)\, x_{\{l\}} + (i+)\, x_{\{m\}} \qquad (1 \leqslant k < l < m \leqslant n),$$

we get

$$\varepsilon_{\{i+\}}(x_{\{k,l,m\}}) = (t_k + t_l) + (t_k + t_m) + (t_l + t_m) + t_k + t_l + t_m = t_k + t_l + t_m,$$

and the general result follows.

2.22. *Remark.* Note that $\dfrac{x = \varepsilon x + a}{x = a}$ $(a \in A)$ is *not* a valid conditional equation: the equation $x = \varepsilon x + a$, an *unguarded* recursive specification, has infinitely many solutions, even in the initial algebra (for instance $\{a + b\colon b \in A\}$ are all solutions).

2.23. *Remark.* Theorem 2.21 can be extended to infinite processes. To give an example, if a^ω is the solution of the guarded recursive specification $\{x = ax\}$ (see 2.18), then $\varphi(a^\omega)$ is the graph in Fig. 9.

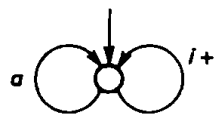


Fig. 9

This graph has specification $\{x_0 = (i+)x_0 + ax_0\}$, so by $\mathrm{EAR}\varepsilon_{\{i+\}}(x_0) = a^\omega$.

2.24. *Remark.* There are several ways in which we could formulate the operational rewrite rules of 2.3, for which we still have Theorems 2.11 and 2.21. An alternative is to have rules

$$a\left(\frac{ax+y}{x}\right) \quad (\text{for } a \in A) \text{ and } \quad \varepsilon\left(\frac{\varepsilon+x}{}\right),$$

thus avoiding the use of an 'internal' step $(i+)$, and the need for abstraction $(\varepsilon_{\{i+\}})$ in Theorems 2.11 and 2.21.

3. Process algebra

3.1. Note that in BPA_ε, if the initial configuration is a multiset of 1 element, then each resulting configuration will have 0 or 1 element, so we actually do not use the multiset aspect of 1.2; namely that operational rewrite rules can work on subsets of a configuration. This is changed when we extend BPA by a parallel composition, an operator merge, obtaining the system PA, *Process algebra* (see Bergstra and Klop [7]). It is unfortunate, that we have to drop the extra constant ε at the same time. This is because problems have arisen in connection with the combination of $\|$ and ε. Defining $\varepsilon \lfloor\!\lfloor x = x$, as is done in Koymans and Vrancken [10], leads to a merge operator which is not associative, an unpleasant situation (to see this, calculate terms $((a+\varepsilon)\|b)\|c$ and $(a+\varepsilon)\|(b\|c)$, where $a, b, c \in A$). If we want an associative merge operator, we should not allow the execution of the ε-step in $(a+\varepsilon)\|b$, and then, term $(a+\varepsilon)$ behaves different in a merge than outside a merge, making the formulation of operational rewrite rules very difficult for recent developments, see [12]. Therefore, we consider the system PA without ε, but we will still use the model $G/\leftrightarrows_\varepsilon$ for the analogon of Theorem 2.11, and the system BPA_ε for the analogon of Theorem 2.21.

3.2. Definition. $PA = (\Sigma_{PA}, E_{PA})$, where

$$\Sigma_{PA} = \begin{cases} S\colon A, P, \text{ with } A \subseteq P, \\ F\colon +\colon P \times P \to P, \\ \cdot\colon P \times P \to P, \\ \|\colon P \times P \to P \text{ (merge)}, \\ \lfloor\!\lfloor\colon P \times P \to P \text{ (left-merge)}, \\ C, \end{cases}$$

and E_{PA} is

$$\boxed{\begin{array}{c} x+y = y+x \\ (x+y)+z = x+(y+z) \\ x+x = x \\ (x+y)z = xz+yz \\ (xy)z = x(yz) \\ x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x \\ a \lfloor\!\lfloor x = ax \\ (ax) \lfloor\!\lfloor y = a(x \| y) \\ (x+y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z \end{array}}$$

Table 3

(here x, y, $z \in P$ and $a \in A$).

3.3. Comments.

1. The left-merge $\lfloor\!\lfloor$ is an auxiliary operator, which enables us to give a finite axiomatisation of merge $\|$. Intuitively, $x \lfloor\!\lfloor y$ is $x \| y$, but with the restriction that the first step must come from x.

2. The first equation says that we consider $x \| y$ to be the *arbitrary interleaving* or *shuffle* of processes x and y.

3. The last three equations recursively define the left-merge on closed terms. Note that a definition by recursion now has cases a, ax and $x+y$, in the absence of ε.

3.4. LEMMA. *Let x, y be closed PA-terms. Then*:

1. $x \| y = y \| x$,
2. $(x \| y) \| z = x \| (y \| z)$.

Proof. 1. Immediate; 2. See Bergstra and Klop [6].

3.5. Operational rewrite rules. We could use operational rewrite rules for PA, analogous to those given in 2.3, since the operators $\|$ and $\lfloor\!\lfloor$ can be eliminated from any closed term (see Bergstra and Klop [6]). However, we do not want to do that, but want to split a term $x \| y$ into a configuration $\{x, y\}$, in order to be able to discuss issues like true concurrency later on. Then, we are forced to abandon a rule like $(i+)$, as the following example (3.6) shows. For the same reason, we cannot have a rule of the form

$$\left(\frac{(x \| y)+z}{x,\ y}\right),$$

for such a rule would also mean that a choice is made without executing a

real step (an $a \in A$). A consequence is, that we cannot 'split the merge' in a term of the form $(x \| y)+z$ directly, but have to evaluate $x \| y$ first, until an atom guards the term. We now present the operational rewrite rules for PA. We treat $+$ as indicated in 2.24

$$a\left(\frac{a+x}{}\right), \quad a\left(\frac{ax+y}{x}\right) \quad \text{(for each } a \in A\text{)},$$

$$i\|\left(\frac{x \| y}{x,\ y}\right).$$

Note that the first rule is a combination of rule a and ε, when we have the constant ε.

3.6. Example. Suppose we have a rule $(i+)$ as in 2.3, in addition to the rule $(i\|)$ of 3.5. Let a, b, $c \in A$, and consider the term $(a+b)\|c$. Forgetting trivial rule-applications, its transformation graph will be the graph in Fig. 10, and after abstracting from $i+$ and $i\|$, this graph ε-bisimulates with the graph of $(a+b)c+c(a+b)+ca+cb$. which is not correct. The two offending edges are indicated.

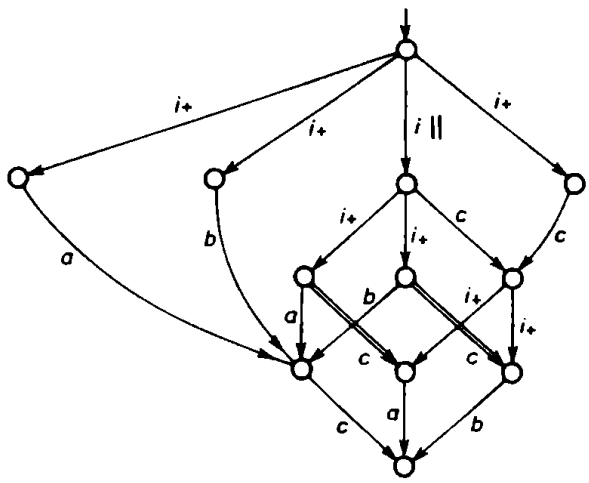


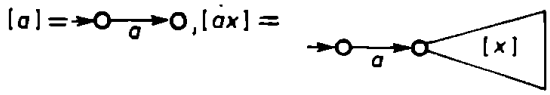
Fig. 10

3.7. Examples. Let a, $b \in A$, and let x be any closed PA-term.

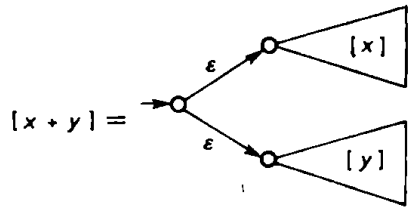
1. $\{a\} \xrightarrow{a} \emptyset$, since $a = a+a$. Likewise $\{ax\} \xrightarrow{a} \{x\}$;
2. $\{a \| b\} \xrightarrow[i\|]{} \{a, b\} \xrightarrow{a} \{b\} \xrightarrow{b} \emptyset$;
3. $\{a \| b\} \xrightarrow[i\|]{} \{a, b\} \xrightarrow{b} \{a\} \xrightarrow{a} \emptyset$.
4. $\{a \| b\} \xrightarrow{a} \{b\}$, since $a \| b = ab+ba$.

3.8. *Remark.* Since operators $\|$, $\lfloor\!\lfloor$ can be eliminated from every closed PA-term, we have that each closed PA-term is equal to a BPA_ε-term, and so

$G/\leftrightarrows_\varepsilon$ becomes a model for PA. Let $[t]$ be the standard interpretation of a closed PA-term in G. E.g., put



and



Now we want to prove the analogon of Theorem 2.11 for PA-terms. In this proof, we will need to use induction on multisets of terms, so we need a well-founded ordering on them. We define such an ordering in 3.10.

3.10. Definitions. 1. Let t be closed PA-term. We define the *length* of t, $|t|$ (intuitively the maximal number of atomic steps that t can do), inductively: $|a| = 1$ for any $a \in A$; $|xy| = |x| + |y|$ for closed terms x, y; $|x+y| = \max\{|x|, |y|\}$ for closed terms x, y. Note that this definition ensures that $x = y \Rightarrow |x| = |y|$.

2. Let $\lesssim$ be the normal ordering of multisets of natural numbers (so for example $\{4\} \lesssim \{4, 4\} \lesssim \{5\}$). This is a linear well-ordering. Then, if C_1, C_2 are two multisets of closed PA-terms, we define $C_1 < C_2$ iff $\{|t|: t \in C_1\} \lesssim \{|t|: t \in C_2\}$, and we obtain a partial well-ordering.

3.11. Theorem. *In G, we have for each closed PA-term t:*

$$\varepsilon_{\{t\|\}} \circ \varphi(t) \leftrightarrows_\varepsilon [t].$$

Proof. If C is a multiset of terms, then $\varphi(C)$ is the transformation graph obtained by taking C as the initial configuration (so $\varphi(\{t\}) = \varphi(t)$). We use induction on finite multisets C, ordered as defined in 3.10, to prove that

$$\varepsilon_{\{t\|\}} \circ \varphi(C) \leftrightarrows_\varepsilon [\mathop{\|}_{t \in C} t]$$

(here $\mathop{\|}_{t \in C} t$ means $t_1 \| \ldots \| t_n$, if $C = \{t_1, \ldots, t_n\}$). Since there are no PA-terms of length 0, the basis for the induction will consist of those multisets C with a single element of length 1.

Case 1. $C = \{t\}$, with $|t| = 1$. Then it is easy to see that we must have $t = \sum_{i=1}^{n} a_i$ for some $a_i \in A$. Also, because of the crucial observation $|x\|y| = |x|$

$+|y|$, rule $i\|$ cannot be applied to t, and we can only apply rule a_i $(1 \leqslant i \leqslant n)$ to configuration $\emptyset$,

$$\varepsilon_{\{i\|\}} \circ \varphi(t) = \varphi(t) \leftrightarrows_\varepsilon [t]$$

follows, because the two graphs in Fig. 11 ε-bisimulate (we took $n = 3$).

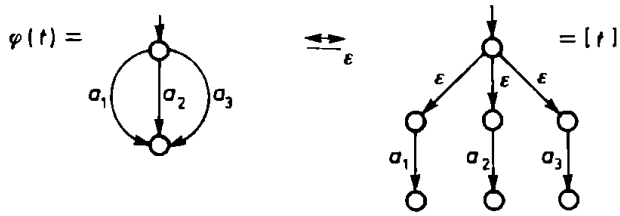


Fig. 11

Case 2. Otherwise. By induction, we can suppose we have proved the theorem for each multiset $D < C$. Three kinds of steps could be possible from C:

1. If a $t \in C$ is of the form $a+x$, with $a \in A$, then we can do a step a to $D = C - \{t\}$;

2. If a $t \in C$ is of the form $ax+y$, with $a \in A$, then we can do a step a to $E = (C \cup \{x\}) - \{t\}$;

3. If a $t \in C$ is of the form $x \| y$, then we can do a step $i\|$ to $F = (C \cup \{x, y\}) - \{t\}$.

Note that D, E, $F < C$, so induction applies. Thus we have $\varphi(F) \leftrightarrows_\varepsilon [\|_{t \in F} t] = [\|_{t \in C} t]$, so that applying $i\|$ gives the right result. It is not hard to finish the proof.

Now we prove the analogon of Theorem 2.21 for the ORS (PA, $A \cup \{i\|\}$). Note that a specification of a graph from $G/\leftrightarrows_\varepsilon$ will be a specification in BPA_ε, so that the following proof can take place in $BPA_{\varepsilon+}$.

3.12. THEOREM. *In $BPA_\varepsilon+$, we can prove for each closed PA-term t: if F specifies $\varphi(t)$, with variable x_0 for the root, then $\varepsilon_{\{i\|\}}(x_0) = t$.*

Proof. Besides atoms from A, F will contain an extra atom $i\|$. In each case, F will be guarded. As in 3.11, we use induction on nonempty finite multisets C, to prove: if F specifies $\varphi(C)$, with variable x_0 for the root, then $\varepsilon_{\{i\|\}}(x_0) = \|_{t \in C} t$.

Case 1. $C = \{t\}$, with $|t| = 1$. We then know that $t = \sum_{i=1}^{n} a_i$ for some $a_i \in A$. We see that $F = \{x_0 = \sum_{i=1}^{n} a_i\}$, and the result is immediate.

Case 2. Otherwise. Three kinds of steps are possible from C:

1. Suppose k atomic steps are possible, say $a_1, \ldots, a_k$, to nonempty configurations $B_1, \ldots, B_k$ $(k \geqslant 0)$;

2. Suppose l atomic steps are possible, say $b_1, \ldots, b_l$, to the empty configuration, $\emptyset$ $(l \geqslant 0)$;

3. Suppose there are m distinct ways to write a $t \in C$ as $t' \| t''$. Then m steps $i\|$ are possible, to configurations $D_1, \ldots, D_m$. Note that configurations B_i, D_j are $< C$. Specification F will have a variable y_i for configuration B_i, and z_j for configuration D_j. F consists of the specifications for $B_1, \ldots, B_k$, $D_1, \ldots, D_m$ plus equation

$$x_0 = \sum_{i=1}^{k} a_i y_i + \sum_{p=1}^{l} b_p + \sum_{j=1}^{m} (i\|) z_j.$$

By induction

$$\varepsilon_{\{i\|\}}(z_j) = \mathop{\|}_{t \in D_j} t, \quad \text{so} \quad \varepsilon_{\{i\|\}}(z_j) = \mathop{\|}_{t \in C} t,$$

and

$$\varepsilon_{\{i\|\}}(x_0) = \sum_{i=1}^{k} a_i \varepsilon_{\{i\|\}}(y_i) + \sum_{p=1}^{l} b_p + \mathop{\|}_{t \in C} t,$$

and it is not hard to finish the proof.

3.13. Deadlock. We can extend our theory by adding a constant δ for *deadlock* (or lock), the acknowledgement of a process that it cannot do anything anymore, has no alternative. δ will therefore be used for unsuccessful termination. To be precise:

$$\delta = (\Sigma_\delta, E_\delta), \quad \text{with} \quad \Sigma_\delta = \begin{cases} S\colon A, P, & \text{with} \quad A \subseteq P, \\ F\colon \\ C\colon \delta \in A \end{cases}$$

and

$$E_\delta = \boxed{\begin{array}{c} x + \delta = x \\ \delta x = \delta \end{array}}$$

Table 4

Here $x \in P$. We can add δ to BPA_ε, obtaining $BPA_{\varepsilon\delta}$; to $BPA_\varepsilon+$, obtaining $BPA_{\varepsilon\delta}+$, and to PA, obtaining PA_δ. In $BPA_{\varepsilon\delta}+$, we must require that for each ε_I, $\delta \notin I$, so that $\varepsilon_I(\delta) = \delta$.

3.14. Notes. 1. In $BPA_{\varepsilon\delta}$, the axiom $x + \delta = x$ is equivalent to $\varepsilon + \delta = \varepsilon$ for if $\varepsilon + \delta = \varepsilon$ holds, we have $x + \delta = \varepsilon x + \delta x = (\varepsilon + \delta) x = \varepsilon x = x$.

2. In PA_δ, we can prove by a straightforward induction that for all closed terms x we have $x \| \delta = x\delta$.

3.15. THEOREM. *$G/\leftrightarrows_\varepsilon$ is a model for $BPA_{\varepsilon\delta}+$ and for PA_δ.*

Proof. Define $[\delta] =$  $_{\varepsilon}$. Note that then

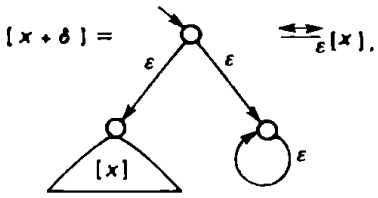


Fig. 12

and $[\delta x] = [\delta]$ since $[\delta]$ has no endpoint. Also EAR still holds, in fact we can give an algebraic definition of δ by putting $\delta = \varepsilon_{\{a\}}(a^{\omega})$, where a^{ω} is the solution of $\{x = ax\}$.

3.16. Operational rewrite rules. For $BPA_{\varepsilon\delta}$, we use the same operational rewrite rules as for BPA_{ε}; for PA_{δ}, the same rules as for PA, so there is no rule for δ. Thus, we get the ORS $(BPA_{\varepsilon\delta}, (A-\{\delta\}) \cup \{\varepsilon, i+\})$ and the ORS $(PA, (A-\{\delta\}) \cup \{i\|\})$. We formulate theorems as before, and indicate the difference in the proofs.

3.17. THEOREM. *In G, we have for each closed $BPA_{\varepsilon\delta}$-term t,*

$$\varepsilon_{\{i+\}} \circ \varphi(t) \leftrightarrows_{\varepsilon} [t].$$

Proof. If $t = \delta$, we have

$\varepsilon_{\{i+\}} \circ \varphi(\delta) = \varepsilon_{\{i+\}}($  $i+) =$  $\varepsilon = [\delta]$.

Since $\{t\} = \{t+\delta\} \xrightarrow[i+]{} \{\delta\} \xrightarrow[i+]{} \{\delta\}$, each node in each $\varphi(t)$ will have an extra summand

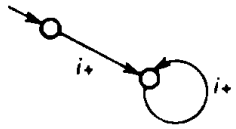
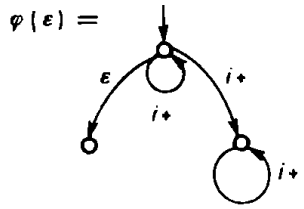


Fig. 13

so for instance



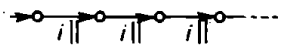
but these summands can disappear after abstraction.

3.18. THEOREM. *In $BPA_{\varepsilon\delta}+$, we have for each closed $BPA_{\varepsilon\delta}$-term t, that if F is the specification belonging to graph $\varphi(t)$, and x_0 corresponds to the root, then $\varepsilon_{\{i+\}}(x_0) = t$.*

Proof. If $t = \delta$, we have $F = \{x_0 = (i+)x_0\} = \{x_0 = (i+)x_0 + \delta\}$, so by EAR $\varepsilon_{\{i+\}}(x_0) = \delta$. Other cases are similar.

3.19. THEOREM. *In G, we have for each closed PA_δ-term t,*

$$\varepsilon_{\{i\|\}} \circ \varphi(t) \leftrightarrows_\varepsilon [t].$$

Proof. We define $|\delta| = 0$. Since $\delta = \delta \| \delta$, we get $\varphi(\delta) :=$ 

and, after abstraction, this graph bisimulates with $[\delta]$. We remark, using 3.14.2, that only terms ending in δ will be different than in 3.11. Those terms will get in addition infinite sequences of $i\|$ steps, bisimulating with $[\delta]$ in each case.

3.20. THEOREM. *In $BPA_{\varepsilon\delta}+$, we have for each closed PA_δ-term t, that if F specifies $\varphi(t)$, with variable x_0 for the root, then $\varepsilon_{\{i\|\}}(x_0) = t$.*

Proof. If $t = \delta$, we have $F = \{x_n = (i\|)x_{n+1} : n \in N\}$, and by EAR $\varepsilon_{\{i\|\}}(x_0) = \delta$. As an example of a different term, consider $t = a\delta$ (some $a \in A - \{\delta\}$). F consists of equations

$$x_n = ay_n + (i\|)x_{n+1} \quad \text{and} \quad y_n = (i\|)y_{n+1}, \quad (n \in N).$$

Since all equations for y_n are satisfied by y, the unique solution of $\{y = (i\|)y\}$, $y_n = y$ for each n by RSP. Then $\varepsilon_{\{i\|\}}(x_0) = a \cdot \varepsilon_{\{i\|\}}(y) = a\delta$, by applying EAR twice.

3.21. True concurrency. We consider two parts of the transformation tree of $a\|b$ in Figs. 14 and 15.

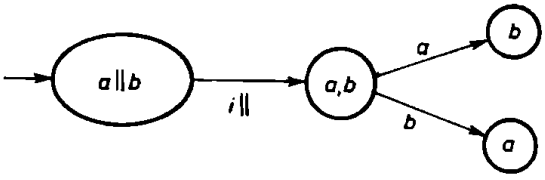


Fig. 14

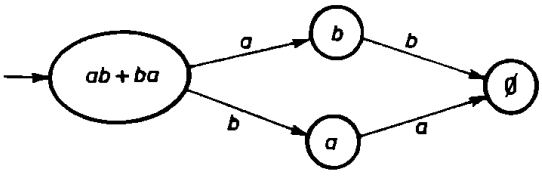


Fig. 15

In Fig. 14, we split the merge $a\|b$, and get configuration $\{a, b\}$. Then, we can either do a step a, working on subconfiguration $\{a\}$, or a step b, working on subconfiguration $\{b\}$. Thus, two steps are possible, working on

disjoint subconfigurations. In Fig. 15, we consider $a \| b$ not as a merge, but as $ab+ba$. Again, we can do a step a or a step b, but now note that these two steps work on the *same* configuration. With the term *true concurrency*, we mean the idea that in a computer having more than one processor, processes can run on different processors *in parallel*, if not simultaneous then at least faster than they would run sequentially on one processor. This is also the idea that can be modeled by Petri nets. As we saw in 1.3, operational rewrite rules constitute a more powerful, and more algebraical, approach than Petri nets. We saw in Sections 2 and 3, that operational rewrite rules can capture the operational behaviour of processes, without losing their denotational identity. Splitting a merge $x \| y$ into a configuration $\{x, y\}$ captures the idea of putting parallel processes on different processors, so that, in Fig. 14 steps a and b can be executed simultaneously, or if not simultaneously, at least faster than a sequential execution as in Fig. 15. In Section 4, we will enlarge this approach to encompass parallel processes with communication, so we will define an operational rewrite system for ACP, the Algebra of Communicating Processes. In Section 5, we will give a real-time semantics of processes, that encorporates the ideas about parallelism above.

4. Algebra of communicating processes

4.1. Now we want to extend the system PA with a communication function, so that in a merge $x \| y$, we could also do a communication step $a|b$ with a a step from x and b a step from y. $a|b$ will again be an atomic step or δ, so $|$ is a function from $A \times A$ to A. If $a|b = \delta$, we say a and b do not communicate. In applications, we will always specify the communication function on atoms. In order to give an equational definition of the (communicating) merge, we will extend $|$ to a function from $P \times P$ to P. For more information, see Bergstra and Klop [7].

4.2. Definition. $ACP = (\Sigma_{ACP}, E_{ACP})$, with

$$\Sigma_{ACP} = \begin{cases} S\colon A, P, \quad \text{with } A \subseteq P, \\ F\colon +\colon P \times P \to P, \\ \quad \cdot\colon P \times P \to P, \\ \quad \|\colon P \times P \to P, \\ \quad \lfloor\!\lfloor\colon P \times P \to P, \\ \quad |\colon A \times A \to A \\ \quad |\colon P \times P \to P \\ \quad \partial_H\colon P \to P \quad (\text{for } H \subseteq A), \\ C\colon \delta \in A, \end{cases}$$

and E_{ACP} is displayed in Table 5. Here $x, y, z \in P$; $a, b \in A$, $H \subseteq A$.

ACP.

$x+y=y+x$	A1
$x+(y+z)=(x+y)+z$	A2
$x+x=x$	A3
$(x+y)z=xz+yz$	A4
$(xy)z=x(yz)$	A5
$x+\delta=x$	A6
$\delta x=\delta$	A7
$a\mid b=b\mid a$	C1
$(a\mid b)\mid c=a\mid(b\mid c)$	C2
$\delta\mid a=\delta$	C3
$x\parallel y=x\mathbin{\llcorner\!\!\llcorner} y+y\mathbin{\llcorner\!\!\llcorner} x+x\mid y$	CM1
$a\mathbin{\llcorner\!\!\llcorner} x=ax$	CM2
$(ax)\mathbin{\llcorner\!\!\llcorner} y=a(x\parallel y)$	CM3
$(x+y)\mathbin{\llcorner\!\!\llcorner} z=x\mathbin{\llcorner\!\!\llcorner} z+y\mathbin{\llcorner\!\!\llcorner} z$	CM4
$(ax)\mid b=(a\mid b)x$	CM5
$a\mid(bx)=(a\mid b)x$	CM6
$(ax)\mid(by)=(a\mid b)(x\parallel y)$	CM7
$(x+y)\mid z=x\mid z+y\mid z$	CM8
$x\mid(y+z)=x\mid y+x\mid z$	CM9
$\partial_H(a)=a$ if $a\notin H$	D1
$\partial_H(a)=\delta$ if $a\in H$	D2
$\partial_H(x+y)=\partial_H(x)+\partial_H(y)$	D3
$\partial_H(xy)=\partial_H(x)\,\partial_H(y)$	D4

Table 5

4.3. Notes. 1. $|\colon A\times A\to A$ is the restriction of $|\colon P\times P\to P$ to $A\times A$.

2. In addition to the axioms presented in Table 5, we assume the *Handshaking Axiom* (HA),

$x\mid y\mid z=\delta$

Table 6

which says that all communication must be binary. We call all atoms in $A\mid A=\{c\in A-\{\delta\}\colon \exists a,\ b\in A\ \ c=a\mid b\}$ the *communication actions*; thus, handshaking means that communication actions do not communicate themselves.

4.4. LEMMA. *The following hold for all closed ACP-terms x, y, z and atoms a, b:*

1. $x\mid y=y\mid x$;
2. $x\parallel y=y\parallel x$;
3. $(x\mid y)\mid z=x\mid(y\mid z)$;

4. $(x \| y) \| z = x \| (y \| z)$;

5. $x \| \delta = x\delta$.

Proof. 1, 2, 3, 4: see Bergstra and Klop [6].

5. By induction:

1. $a \| \delta = a\delta + \delta a + a | \delta = a\delta + \delta + \delta = a\delta$;

2. $ax \| \delta = a(x \| \delta) + \delta ax + ax | \delta = ax\delta + \delta + \delta = ax\delta$;

3. $(x+y) \| \delta = (x+y) \mathbin{\|\!\lfloor} \delta + \delta \mathbin{\|\!\lfloor} (x+y) + (x+y) | \delta = x \mathbin{\|\!\lfloor} \delta + y \mathbin{\|\!\lfloor} \delta + \delta + x | \delta + y | \delta = (x \mathbin{\|\!\lfloor} \delta + \delta \mathbin{\|\!\lfloor} x + x | \delta) + (y \mathbin{\|\!\lfloor} \delta + \delta \mathbin{\|\!\lfloor} y + y | \delta) = x \| \delta + y \| \delta = x\delta + y\delta = (x+y)\delta$.

4.5. *Remark.* By Bergstra and Klop [6], we can eliminate operators $\|$, $\mathbin{\|\!\lfloor}$, $|$, ∂_H from all closed ACP-terms. Thus, $G/\leftrightarrows_e$ becomes a model for ACP, when we define the interpretation of δ as in 3.15.

4.6. Operational rewrite rules. Again note, that since we can eliminate all operators $\|$, $\mathbin{\|\!\lfloor}$, $|$, ∂_H from closed terms (see Bergstra and Klop [6]) the rules of 2.3 would work in this case too. However, we want to consider the merge as we did in 3.5. Then it is not too hard to extend the rules of 3.5 to the case with communication. Specifically, we would have the rules:

$$a\left(\frac{a+x}{}\right), \quad a\left(\frac{ax+y}{x}\right) \quad \text{for} \quad a \in A - \{\delta\};$$

$$i\|\left(\frac{x \| y}{x, y}\right), \quad \text{and} \quad a|b\left(\frac{a+x,\ b+y}{}\right), \quad a|b\left(\frac{ax+y,\ b+z}{x}\right),$$

$$a|b\left(\frac{ax+y,\ bz+w}{x, z}\right) \quad \text{if} \quad a, b \in A \quad \text{and} \quad a|b \neq \delta.$$

However, we also want to have a rule for the encapsulation operator ∂_H. For, in general, a network of communicating processes will be given by a term of the form

$$\partial_H(x_1 \| x_2 \| \ldots \| x_n),$$

where H will contain all communication 'halves', i.e., $H = \{a \in A\colon \exists b \in A\ a|b \neq \delta$ (and a and b occur in $x_1, \ldots, x_n)\}$, so that communications among $x_1, \ldots, x_n$ are encapsulated, shielded off from the environment. Without a rule for ∂_H we cannot 'split the merge' in the term $x_1 \| \ldots \| x_n$. But then, we have to remember in some way which actions cannot be performed, and also the relative positions of the ∂_H and the merges. To give an example, if $a, b \in A$, $a \neq b$ and $a|a = b$, then

$$\partial_{\{a\}}(a) \| \partial_{\{a\}}(a) = \delta \quad \text{but} \quad \partial_{\{a\}}(a \| a) = b.$$

Therefore, we will define operational rewrite rules not over the initial algebra of ACP, but over an algebra of tuples $\langle t_1, \ldots, t_n, t_{n+1} \rangle$, where $n \geqslant 0$, t_{n+1} is

a closed ACP-term, and if $1 \leqslant j \leqslant n$, then $t_j \subseteq A$ or $t_j \in \{l, r\}$ (we use the symbols l, r (left, right) to indicate where a split of a merge takes place). If $\sigma = \langle s_1, \ldots, s_{n+1} \rangle$ and $\varrho = \langle r_1, \ldots, r_{n+1} \rangle$ are two such tuples of equal length, then we define $\sigma = \varrho$ iff $s_1 \equiv r_1, \ldots, s_n \equiv r_n$ and $ACP \vdash s_{n+1} = r_{n+1}$. A piece of notation: if σ and ϱ are two tuples, then $\sigma * \varrho$ is their concatenation. Configurations will be multisets of tuples, and the initial configuration for a closed ACP-term t is $\{\langle t \rangle\}$. In 4.8, we will present a set of operational rewrite rules for ACP, that takes these features into account. First we will define the *length* of an ACP-term (compare 3.10).

4.7. Definition. If t is a closed ACP-term, then $|t|$, the *length* of t, is the maximal number of atomic steps that t can do, counting communication steps double. Inductively:

1. $|\delta| = 0$;
2. $|a| = 2$ if a is a communication step (i.e., $a \in A \mid A$);
3. $|a| = 1$ if a is an atom which is not a communication step (i.e., $a \in A - (A \mid A)$);
4. $|xy| = |x| + |y|$ if x, y are closed terms;
5. $|x+y| = \max\{|x|, |y|\}$ if x, y are closed terms.

Again note that with this definition we have $ACP + HA \vdash x = y \Rightarrow |x| = |y|$. Also, note that $|\partial_H(x)| \leqslant |x|$ for all $H \subseteq A$ and all closed terms x. We remark that length could also be defined for infinite processes, that are defined by a finite recursive specification, by stipulating that the length of a process is the maximal number of steps to termination, or until a previously attained state is reached.

4.8. Operational rewrite rules for ACP. Let σ, ϱ_1, ϱ_2 be tuples consisting of subsets of A and the symbols l, r. We have the following rules:

$$a\left(\frac{\sigma * \langle a+x \rangle}{}\right), \quad a\left(\frac{\sigma * \langle ax+y \rangle}{\sigma * \langle x \rangle}\right) \quad \text{if } a \in A - \{\delta\} \text{ and } a \notin H \text{ for each } H \text{ in } \sigma;$$

$$i\|\left(\frac{\sigma * \langle x \| y \rangle}{\sigma * \langle l, x \rangle, \sigma * \langle r, y \rangle}\right) \quad \text{if } x, y \neq \delta;$$

$$i\partial\left(\frac{\sigma * \langle \partial_H(x) \rangle}{\sigma * \langle H, x \rangle}\right) \quad \text{if } H \subseteq A \text{ and } |\partial_H(x)| = |x|;$$

$$a \mid b\left(\frac{\sigma * \varrho_1 * \langle a+x \rangle, \sigma * \varrho_2 * \langle b+y \rangle}{}\right), \quad a \mid b\left(\frac{\sigma * \varrho_1 * \langle ax+y \rangle, \sigma * \varrho_2 * \langle b+z \rangle}{\sigma * \varrho_1 * \langle x \rangle}\right),$$

$$a \mid b\left(\frac{\sigma * \varrho_1 * \langle ax+y \rangle, \sigma * \varrho_2 * \langle bz+w \rangle}{\sigma * \varrho_1 * \langle x \rangle, \sigma * \varrho_2 * \langle z \rangle}\right),$$

the last three if a, $b \in A$, $a \mid b \neq \delta$ and

1. first $(\varrho_1) = l$, first $(\varrho_2) = r$,
2. $a \notin H$ for each H in ϱ_1,
3. $b \notin H$ for each H in ϱ_2,
4. $a \mid b \notin H$ for each H in σ.

Notes. 1. In rule $i\|$, we have the condition $x, y \neq \delta$, or equivalently, $|x|$, $|y| \geqslant 1$. The condition ensures that $\{|x||y|\} > \{|x|, |y|\}$, which is useful in the following inductive proofs.

2. In rule $i\partial$, condition $|\partial_H(x)| = |x|$ is also useful in the following inductive proofs. This is not an unreasonable assumption to make, for if $|\partial_H(x)| < |x|$, we have that $\partial_H(x)$ ends in deadlock, a condition we would not want any program $\partial_H(x_1 \| \ldots \| x_n)$ to satisfy (note: this is why we need to count communication steps as double steps).

4.9. Example. Suppose $a, b, c \in A$ are distinct, and $a \mid b = c$. Then

$$\{\langle \partial_{\{a,b\}}(\partial_{\{b,c\}}(a) \| \partial_{\{a,c\}}(b))\rangle\} \xrightarrow[i\partial]{} \{\langle\{a, b\}, \partial_{\{b,c\}}(a) \| \partial_{\{a,c\}}(b)\rangle\} \xrightarrow[i\|]{}$$

$$\{\langle\{a, b\}, l, \partial_{\{b,c\}}(a)\rangle, \langle\{a, b\}, r, \partial_{\{a,c\}}(b)\rangle\} \xrightarrow[i\partial]{} \ldots$$

$$\ldots \xrightarrow[i\partial]{} \{\langle\{a, b\}, l, \{b, c\}, a\rangle, \langle\{a, b\}, r, \{a, c\}, b\rangle\} \xrightarrow[c]{} \emptyset,$$

but also

$$\{\langle \partial_{\{a,b\}}(\partial_{\{b,c\}}(a) \| \partial_{\{a,c\}}(b))\rangle\} = \{\langle c\rangle\} \xrightarrow[c]{} \emptyset.$$

4.10. Theorem. *For each closed ACP-term t we have in G that $\varepsilon_I \circ \varphi(t) \leftrightarrows_\varepsilon [t]$ (where $I = \{i\|, i\partial\}$).*

Proof. We consider multisets C of tuples as defined above, and define term $(C) = \{\text{last}(\sigma)\colon \sigma \in C\}$, the multiset of terms of C, and we order these multisets according to term length, so $C < D \Leftrightarrow \{|t|\colon t \in \text{term}(C)\} \lesssim \{|t|\colon t \in \text{term}(D)\}$. We observe that in a graph $\varphi(t)$ (so starting from a node $\{\langle t\rangle\}$), only multisets C can appear of the following 2 types:

1. $C = \{\sigma * \langle\delta\rangle\}$ for some σ;
2. for all $t \in \text{term}(C)$ we have $t \neq \delta$.

This is because of the restriction in $i\|$. Thus, we are done if we prove that for all multisets C of this form, we have

(a) $\varepsilon_I \circ \varphi(C) \leftrightarrows_\varepsilon [\mathop{\|}_{t \in \text{term}(C)} t]$.

In order to prove this, we need to prove another statement as well, which does not have a simple form. We will prove (a) and

(b) if $C \xrightarrow[I\partial]{} C'$, then $\varepsilon_I \circ \varphi(C) \leftrightarrows_\varepsilon \varepsilon_I \circ \varphi(C')$ by induction on C.

Case 1. $C = \{\sigma * \langle\delta\rangle\}$ for some σ. The only rule we can apply is $i\partial$, namely $C \xrightarrow[i\partial]{} \{\sigma * \langle H, \delta\rangle\}$ for each $H \subseteq A$. $\varphi(C)$ is shown in Fig. 16, and (a) and (b) are easy.

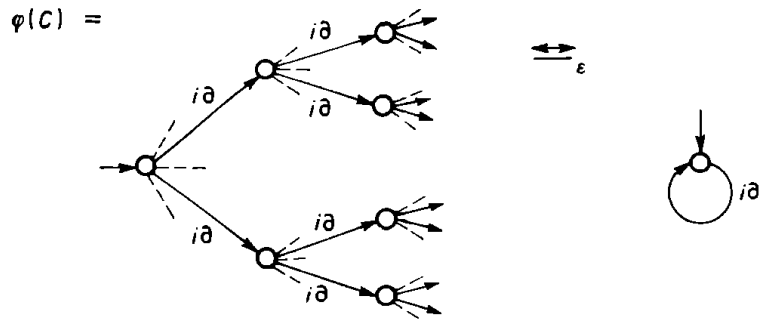


Fig. 16

Case 2. $t \neq \delta$ for each $t \in \mathrm{term}(C)$. We can assume that (a) and (b) hold for all configurations $D < C$ reachable from C. Four kinds of steps may be possible from C:

1. If $\varrho \in C$ is of the form $\sigma * \langle t \rangle$, and t can be written as $\partial_H(x)$, with $H \subseteq A$ and $|x| = |t|$, we can do a step $i\partial$ to a configuration $C' = (C \cup \{\sigma * \langle H, x \rangle\}) - \{\varrho\}$. Note that C' is not less than C.

2. If $\varrho \in C$ is of the form $\sigma * \langle a + x \rangle$, respectively $\sigma * \langle ax + y \rangle$, with $a \in A$ and $a \notin H$ for each H in σ, then we can do a step a to configuration $B = C - \{\varrho\}$, respectively $B = (C \cup \{\sigma * \langle x \rangle\}) - \{\varrho\}$. Note $B < C$.

3. If $\varrho_1 \in C$ is of the form $\sigma_1 * \langle a + y \rangle$ or $\sigma_1 * \langle ax + y \rangle$, $\varrho_2 \in C$ of the form $\sigma_2 * \langle b + w \rangle$ or $\sigma_2 * \langle bz + w \rangle$, and the conditions of the communication rule are satisfied, then we can do a step $a \mid b$ to a configuration D, where D is

$$C - \{\varrho_1, \varrho_2\},\ (C - \{\varrho_1, \varrho_2\}) \cup \{\sigma_1 * \langle x \rangle\},\ (C - \{\varrho_1, \varrho_2\}) \cup (\sigma_2 * \langle z \rangle\}$$

or

$$(C - \{\varrho_1, \varrho_2\}) \cup \{\sigma_1 * \langle x \rangle, \sigma_2 * \langle z \rangle\}.$$

Note that in every case $D < C$.

4. If $\varrho \in C$ is of the form $\sigma * \langle x \| y \rangle$, with $x, y \neq \delta$, then we can do a step $i\|$ to configuration $E = (C - \{\varrho\}) \cup \{\sigma * \langle l, x \rangle, \sigma * \langle r, y \rangle\}$. Note $E < C$.

Now, we can use induction on configurations B, D, E. Note that, when we have $C \xrightarrow[i\partial]{} C'$, then the same steps of type 2 and 3 are possible from C' as from C, but possibly *more* or *less* steps $i\|$ of type 4. However, *after* abstracting from $i\|$ and $i\partial$, any step possible from a configuration E of type 4 is also possible from C as a step of type 2 or 3. In this way, we prove (b).

We illustrate these remarks by considering the term a. Of course, $\{\langle a \rangle\} \xrightarrow[i\partial]{} \{\langle H, a \rangle\}$ for each $H \subseteq A - \{a\}$. But also, if a is a communication step, say $a = b \mid c$, we have $\{\langle a \rangle\} \xrightarrow[i\partial]{} \{\langle H, b \| c \rangle\}$ for each H with $\{b, c\} \subseteq H \subseteq A - \{a\}$, and $i\|$ can be applied to the last configuration. For this example, we show the graph in Fig. 17, and indicate how claims (a) and (b) are proved.

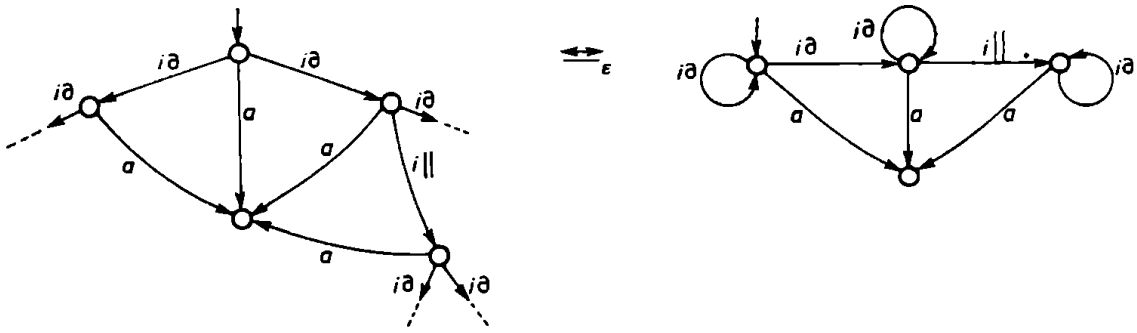


Fig. 17

The general proof will not be given here, but deferred to 4.11.

Now we prove the analogue of 2.21 for *ACP*.

4.11. THEOREM. *In $BPA_{\varepsilon\delta}+$, we can prove for each closed ACP-term t, that if F specifies $\varphi(t)$, with variable x_0 for the root, then $\varepsilon_I(x_0) = t$ (with $I = \{i\|, i\partial\}$).*

Proof. Note that in each case, F will be a guarded recursive specification over $BPA_{\varepsilon\delta}$, with $i\|$ and $i\delta$ as extra atoms. As in 4.10, we use induction on nonempty multisets C, with $C = \{\sigma * \langle\delta\rangle\}$ or $t \neq \delta$ for each $t \in \mathrm{term}(C)$, to prove that if F specifies $\varphi(C)$, and $F = \{x = t_x\colon x \in X\}$ for some set of variables X, with $x_0 \in X$ denoting the root of $\varphi(C)$, then:

(a) if t_{x_1} has a summand $(i\partial)x_2$ $(x_1, x_2 \in X)$, then $\varepsilon_I(x_1) = \varepsilon_I(x_2)$;

(b) $\varepsilon_I(x_0) = \mathop{\|}_{t \in \mathrm{term}(C)} t$.

Case 1. $C = \{\sigma * \langle\delta\rangle\}$ for some tuple σ. Put $X = \{x_\varrho\colon \varrho$ is a tuple of subsets of $A\}$, $x_0 \equiv x_\lambda$ (λ is the empty tuple), then if we take variable x_ϱ for node $\{\varrho * \sigma * \langle\delta\rangle\}$ in graph $\varphi(C)$, we obtain the recursive specification

$$F = \{x_\varrho = \sum_{H \subseteq A} (i\partial)\, x_{\langle H\rangle * \varrho}\colon\ x_\varrho \in X\}$$

of $\varphi(C)$. Note that here, in order to have finite terms, we must require that the set of atomic actions A is finite. Now consider the guarded recursive specification $F' = \{y = (i\delta)y\}$. If process y satisfies F' (y exists by RDP), we infer that y satisfies all equations of F, so by RSP $y = x_\varrho$ for each $x_\varrho \in X$, and (a) is certainly satisfied. Finally, applying EAR to F', we get $\varepsilon_I(x_0) = \varepsilon_I(y) = \delta$.

Case 2. Otherwise. We have $t \neq \delta$ for each $t \in \mathrm{term}(C)$, and (a) and (b) hold for each $D < C$ reachable from C. Note that there only finitely many ways to write a $t \in \mathrm{term}(C)$ as $\partial_H(t')$, with $H \subseteq A$ and $|t| = |t'|$, because C is finite, we assume that A is finite and there are only finitely many terms of a given length. Suppose there are n such ways. If $\partial_H(t')$ is one such rewriting, then again there are at most n ways to write $t' = \partial_{H'}(t'')$, with $H' \subseteq A$ and $|t'| = |t''|$ (possibly less). Three kinds of steps are possible from C:

1. n distinct ways to apply $i\partial$, to configurations $A_1, \ldots, A_n$. Note that no A_i is less than C.

2. Suppose there are k distinct ways to apply an atomic step, say steps $a_1, \ldots, a_k$ to configurations $B_1, \ldots, B_k$. Note that each B_j is less than C. Moreover, if A' is a configuration reached from C by a number of applications of $i\partial$, then the same steps $a_1, \ldots, a_k$ are possible, to configurations $B'_1, \ldots, B'_k$ such that each B'_j is reachable from B_j by a number of applications of $i\partial$.

3. Suppose that there are $2l$ terms $t'_1, \ldots, t'_l, t''_1, \ldots, t''_l$ such that, for each $p \in \{1, \ldots, l\}$, $t'_p \neq \delta$, $t''_p \neq \delta$, there is a $t \in \mathrm{term}(C)$ with $|t| = |t'_p||t''_p|$ and there is a $H \subseteq A$ (possibly $H = \emptyset$) with $t = \partial_H(t'_p \| t''_p)$. Then, there is a subset S of $\{1, \ldots, l\}$ (where $p \in S \Leftrightarrow$ we can take $H = \emptyset$ above) such that for each $p \in S$ we can do a step $i\|$ from C to a configuration D_p. Note that each D_p is less than C. Moreover, if A' is a configuration reached from C by a number of applications of $i\partial$, then there is a subset S' of $\{1, \ldots, l\}$ such that for each $p \in S'$ we can do a step $i\|$ from A' to a configuration D'_p, and if $p \in S$, then D'_p is reachable from D_p by a number of applications of $i\partial$.

Recursive specification F has variables x_σ for each sequence σ from $\{1, \ldots, n\}$, with $x_0 \equiv x_\lambda$ (λ is the empty sequence), variables y_j^σ for configurations B_j (and other configurations reachable from B_j by steps $i\partial$), and variables z_p^σ for configurations D_p (and other configurations reachable from D_p by steps $i\partial$). By induction hypothesis we have, for any two sequences σ_1, σ_2 from $\{1, \ldots, n\}$, that $\varepsilon_I(y_j^{\sigma_1}) = \varepsilon_I(y_j^{\sigma_2})$, $\varepsilon_I(z_p^{\sigma_1}) = \varepsilon_I(z_p^{\sigma_2})$ for each j, p. Then, F consists of equations

$$x_\sigma = \sum_{i \in S_\sigma} (i\partial) x_{\sigma * \langle i \rangle} + \sum_{j=1}^{k} a_j y_j^\sigma + \sum_{p \in T_\sigma} (i\|) z_p^\sigma$$

for appropriate sequences σ (reachable from x_λ) and certain $S_\sigma \subseteq \{1, \ldots, n\}$ and $T_\sigma \subseteq \{1, \ldots, l\}$. In order to do abstraction, using EAR, we need to consider two other specifications. Let t be an atom *not* occurring in F. Then, F^* will consist of equations

$$x_\sigma^* = \sum_{i \in S_\sigma} t x_{\sigma * \langle i \rangle}^* + \sum_{j=1}^{k} a_j y_j^\sigma + \sum_{p \in T_\sigma} (i\|) z_p^\sigma$$

for appropriate σ. Without proof, we mention that $\varepsilon_{I \cup \{t\}}(x_\sigma^*) = \varepsilon_I(x_\sigma)$. Now

$$\varepsilon_I(x_\sigma^*) = \sum_{i \in S_\sigma} t\varepsilon_I(x_{\sigma * \langle i \rangle}) + \sum_{j=1}^{k} a_j \varepsilon_I(y_j^\sigma) + \sum_{p \in T_\sigma} \varepsilon_I(z_p^\sigma).$$

Note that all actions from $\varepsilon_I(z_p^\sigma)$ will already occur in some $a_j \varepsilon_I(y_j^\sigma)$ so there is a process u such that for all σ

$$u = \sum_{j=1}^{k} a_j \varepsilon_I(y_j^\sigma) + \sum_{p \in T_\sigma} \varepsilon_I(z_p^\sigma),$$

and moreover, by induction we see $u = \|_{t \in C} t$, the desired answer. Next, consider $F_1 = \{g = tg + u\}$. We see that g (the solution of F_1) satisfies all equations of F^*, so by RSP $g = \varepsilon_I(x_\sigma^*)$ for each σ. Finally, by EAR $\varepsilon_{\{t\}}(g) = u$, so

$$\varepsilon_I(x_\sigma) = \varepsilon_{I \cup \{t\}}(x_\sigma^*) = \varepsilon_{\{t\}} \circ \varepsilon_I(x_\sigma^*) = \varepsilon_{\{t\}}(g) = u = \|_{t \in C} t,$$

in particular $\varepsilon_I(x_0) = \|_{t \in C} t$.

4.12. *Remark.* Thus, we see again that for ACP-terms, operational behaviour equals denotational behaviour. We just note, that operational rewrite rules can be formulated for the system ACP_θ (ACP with a priority ordering, used for instance to describe interrupts, introduced in Baeten, Bergstra and Klop [2]), and again the previous theorems will hold, operational behaviour will equal denotational behaviour. In the next section, we will describe a simple real-time semantics for the operational rewrite systems given in this and previous sections.

5. Real-time semantics

In this last section, we show how operational rewrite systems can be used to give a real-time semantics for concurrent, communicating processes, by giving a simple example of such a semantics. Of course, this example can be modified and elaborated upon, to give a more realistic semantics. Here, however, just showing the way was deemed sufficient for our purposes, and thus a number of simplifying assumptions will be encountered.

5.1. Definition. Let $((\Sigma, E), R)$ be one of the operational rewrite systems given in the previous sections, so BPA_ε (Section 2), $BPA_{\varepsilon\delta}$, PA, PA_δ (Section 3) or ACP (Section 4). Let t be a finite Σ-term. An *execution trace* of t is a series of transformation steps, starting with initial configuration $\{t\}$, (or $\{\langle t \rangle\}$ in the case of ACP) and ending in the empty configuration. Inductively, we define the *length* of an execution trace, the *amount of time* it takes to execute this trace.

1. An atomic action (an application of operational rewrite rule $a \in A$) takes 1 unit of time, $[a] = 1$;

2. An internal action (an application of operational rewrite rules $i+$, $i\|$, $i\partial$ or ε) takes 0.1 unit of time, we write

$$[i+] = [i\|] = [i\partial] = [\varepsilon] = 0.1;$$

3. Action r_1 followed by r_2 takes max $\{[r_1], [r_2]\}$ units of time if r_1 and r_2 work on *disjoint* subconfigurations (as explained in 3.19) and $[r_1]+[r_2]$ units of time otherwise;

4. In general, if we have an execution trace ϱ followed by an action $r \in R$, we look at the subconfiguration that r is working on (the subconfiguration that is matched with the top configuration in rule r). This subconfiguration resulted from applications of certain rules in ϱ, but is disjoint from modifications of other rules in ϱ. Now the execution time of ϱ followed by r, $[\varrho * \langle r \rangle]$, is the maximum of $[\varrho]$ and $[\varrho'] + [r]$, where ϱ' is the subtrace of ϱ obtained by leaving out all actions disjoint from r, in the sense explained above.

5.2. EXAMPLE 1. Consider the PA-term aa. Execution trace ϱ_1 is given by $\{aa\} \xrightarrow{}_a \{a\} \xrightarrow{}_a \emptyset$, and we see $[\varrho_1] = 2$. Execution trace ϱ_2 is given by $\{a \| a\} \xrightarrow{}_{i\|} \{a, a\} \xrightarrow{}_a \{a\} \xrightarrow{}_a \emptyset$, and we see $[\varrho_2] = 1.1$. Thus, we see that the parallel execution in trace ϱ_2 is 0.9 units of time *faster* than the sequential execution of trace ϱ_1. We can display these traces on a time axis by putting elements of a configuration below each other. Thus ϱ_1 is

Fig. 18

and ϱ_2 is

Fig. 19

5.3. EXAMPLE 2. A little bit more involved example is given by the ACP-term a^4 ($= aaaa$), if we assume $a = b \mid b$ for some $b \in A$ (of course $a \neq b$). We consider three execution traces: ϱ_1 is the sequential trace, applying rule a four times, so $[\varrho_1] = 4.0$; for ϱ_2, we write $a^4 = \partial_{\{b\}}(aba \| ba)$ and do

$$\{\langle a^4 \rangle\} \xrightarrow{}_{i\partial} \{\langle \{b\}, aba \| ba \rangle\} \xrightarrow{}_{i\|} \{\langle \{b\}, l, aba \rangle, \langle \{b\}, r, ba \rangle\} \xrightarrow{}_a$$
$$\{\langle \{b\}, l, ba \rangle, \langle \{b\}, r, ba \rangle\} \xrightarrow{}_a \{\langle \{b\}, l, a \rangle, \langle \{b\}, r, a \rangle\} \xrightarrow{}_a \{\langle \{b\}, l, a \rangle\} \xrightarrow{}_a \emptyset;$$

and ϱ_3 uses maximal parallelism, writing $a^4 = (a \| a) \| (a \| a)$. We display traces ϱ_2 and ϱ_3.

ϱ_2 is

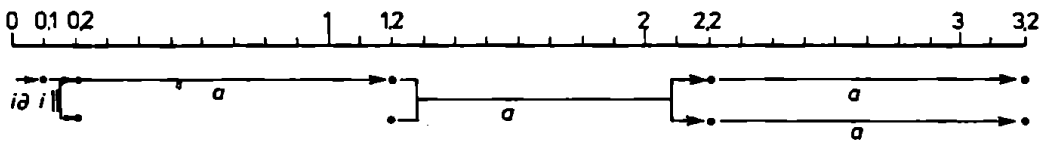


Fig. 20

and ϱ_3 is

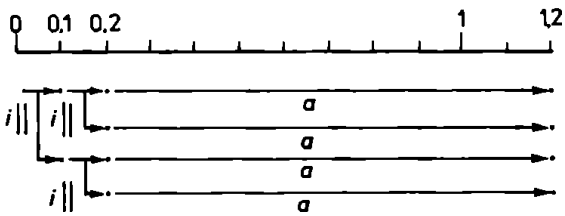


Fig. 21

We see $[\varrho_2] = 3.2$ and $[\varrho_3] = 1.2$. It can be seen that trace ϱ_3 is the fastest way to execute term a^4, and in general that term a^{2^n} can be executed in $1+n/10$ units of time. In order to give an example that is a little bit more realistic, we will extend Definition 5.1 to infinite processes.

5.4. Definition. Let $((\Sigma, E), R)$ be one of the operational rewrite systems BPA_ε, PA, PA_δ, $BPA_{\varepsilon\delta}$ or ACP. Let x be a process over Σ (by which we mean, that x is given as the unique solution of some guarded recursive specification, see 2.16 and 2.17). An *execution trace* of x is a series of transformation steps, starting with initial configuration $\{x\}$ (or $\{\langle x\rangle\}$ in the case of ACP) and ending in the empty configuration *or* ending in a configuration that appeared earlier in the trace. The *length* of such an execution trace is defined as in 5.1. Thus, to give a simple example, if a^ω is the solution of $\{x = ax\}$ (see 2.18), then a^ω has trace $\{a^\omega\} \xrightarrow{a} \{a^\omega\}$, of length 1.

5.5. Definition. As a nontrivial example we will consider a communication protocol. We consider the following network, given in Fig. 22.

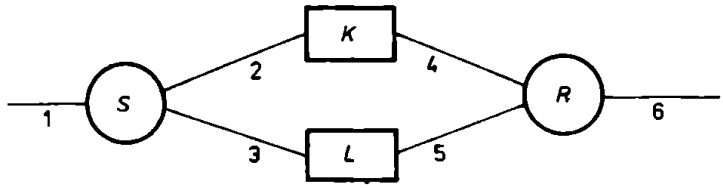


Fig. 22

Such a network consists of *locations* (given by processes, here S, K, L and R) and *channels* between them (here 1, 2, 3, 4, 5, 6). Processes can

communicate along channels, and these communications will consist of the transferal of a piece of data. So, if D is finite set of data, we have the following atomic actions:

$$r_i(d) = \textit{read } d \textit{ along channel } i \ (1 \leqslant i \leqslant 6);$$

$$s_i(d) = \textit{send } d \textit{ along channel } i \ (1 \leqslant i \leqslant 6);$$

$$c_i(d) = \textit{communicate } d \textit{ along } i \ (1 \leqslant i \leqslant 6).$$

Then, the communication function is defined by: $r_i(d) \mid s_i(d) = c_i(d)$, and all other communications give δ (for this terminology, also see e.g. Baeten, Bergstra and Klop [3]). Here, S is a sender, who will send data to receiver R, alternatively through K and buffer L. S, K, L and R are given by guarded recursive specifications, as follows:

$$K = \sum_{d \in D} r_2(d) s_4(d) K$$

$$L = \sum_{d \in D} r_3(d) s_5(d) L$$

$$S = S^0 S^1 S$$

$$S^i = \sum_{d \in D} r_1(d) s_{2+i}(d) \quad \text{for } i = 0,1$$

$$R = R^0 R^1 R$$

$$R^i = \sum_{d \in D} r_{4+i}(d) s_6(d) \quad \text{for } i = 0,1$$

Table 7

Note that the recursive specifications of S and R are not really guarded, but can easily by rewritten in a guarded form. K and L are one-bit buffers. The process we are interested in is

$$X = \partial_H(S \| K \| L \| R)$$

where $H = \{s_i(d), r_i(d)\colon 2 \leqslant i \leqslant 5,\ d \in D\}$.

5.6. *Remark.* X is a correct communication protocol. By this statement, we mean that each action $r_1(d)$ (for a certain $d \in D$) will eventually be followed by $s_6(d)$. In between, however, besides actions from $I = \{c_i(d) \mid 2 \leqslant i \leqslant 5,\ d \in D\}$, also another $r_1(e)$ could be performed, but not another $s_6(e)$, so that actions will be sent along channel 6 in the same order as they were received along channel 1.

5.7. Process X has execution traces ϱ_{de} $(d, e \in D)$ as shown in Fig. 23.

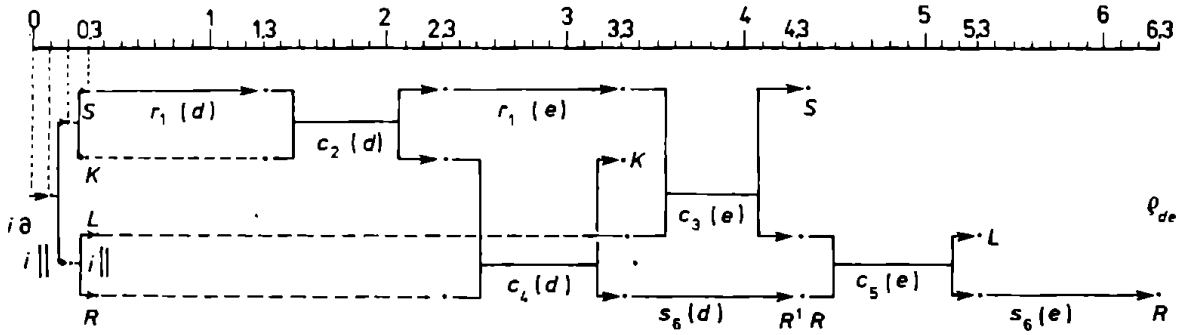


Fig. 23

We see that after 6.3 units of time, the same configuration is reached as after 0.3 units of time (namely $\{\langle H, l, l, S\rangle, \langle H, l, r, K\rangle, \langle H, r, l, L\rangle, \langle H, r, r, R\rangle\}$). Thus, X has execution traces of length 6.3. Note that two executions would take 10.3 units of time, since a copy of the piece of ϱ_{de} between 0.3 and 6.3 can be fitted in between 4.3 and 10.3 units of time. In general, n executions of this protocol, using parallelism as above, take $2.3+4n$ units of time. Since, as indicated above, any execution trace must involve 8 atomic actions, a sequential execution trace (having configurations of only one element) will take at least 8 units of time, and thus n executions will take $8n$ units. So we see that a parallel execution of this protocol is approximately twice as fast as a sequential execution.

5.8. In other papers, we have given considerable attention to the verification of protocols, i.e., determining the behaviour of the process after abstraction from internal actions (in this case, the set of internal actions is $I = \{c_i(d) \mid 2 \leqslant i \leqslant 5, d \in D\}$). We do not want to do that here, so we will just state without proof what $\varepsilon_I(X)$ is. Define a *four-bit buffer* B with input channel 1 and output channel 6 by:

$$B = B_\lambda = \sum_{d \in D} r_1(d) B_{\langle d\rangle} \quad (\lambda \text{ is the empty sequence});$$

$$B_{\sigma * \langle d\rangle} = s_6(d) B_\sigma + \sum_{e \in D} r_1(e) B_{\langle e\rangle * \sigma * \langle d\rangle}.$$

if $\sigma = \lambda$ or $\sigma \in D^1 \cup D^2$ (a sequence from D of length 0, 1, or 2)

$$B_{\sigma * \langle d\rangle} = s_6(d) B_\sigma \quad \text{if} \quad \sigma \in D^3$$

(a sequence from D of length 3).

We claim $\varepsilon_I(X) = B$ (and also $\tau_I(X) = B$), and remark that the architecture of X is an efficient parallel implementation of a four-bit buffer.

References

[1] D. Austry, G. Boudol, *Algèbre de processus et synchronisation*, Theor. Comp. Sci. 30, 1984.

[2] J. C. M. Baeten, J. A. Bergstra, J. W. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fund. Inf. IX (2), 1986, 127–168.

[3] –, –, –, *On the consistency of Koomen's Fair Abstraction Rule*, Theor. Comp. Sci. 51 (1/2), 1987, 129–176.

[4] J. W. De Bakker, *Mathematical theory of program correctness*, Prentice Hall International, 1980.

[5] J. A. Bergstra, J. Heering, J. W. Klop, *Object-oriented algebraic specification: proposal for a notation and 12 examples*, report CS-R8411, Centrum voor Wiskunde en Informatica, Amsterdam 1984.

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

[7] –, –, *Algebra of communicating processes*, Proceedings of the CWI Symposium Mathematics and Comp. Sci., (J. W. de Bakker, M. Hazewinkel and J. K. Lenstra, eds.), North-Holland, Amsterdam, 1986, 89–136.

[8] G. Boudol, G. Roucairol, R. De Simone, *Petri nets and algebraic calculi of processes*, report 292, INRIA, Sophia Antipolis, 1984.

[9] J. A. Goguen, *Order sorted algebra*, Semantics and Theory of Computations Report no. 14 UCLA Comp. Sci. Dept. 1978.

[10] C. J. P. Koymans, J. L. M. Vrancken, *Extending process algebra with the empty process* ε, Logic Group Preprint Series no. 1, Dept. of Phil., State University of Utrecht, 1985.

[11] W. Reisig, *Petri nets*, EATCS monograph on Theoretical Computer Science, Springer, 1985.

[12] J. C. M. Baeten, R. J. van Glabbeek, *Merge and termination in process algebra*, in: Proc. 7th Conf. on Foundations of Software Technology & Theor. Comp. Sci., Pune, India (K. V. Nori, ed.), Springer LNCS 287, 1987, 153–172.

Presented to the semester
Mathematical Problems in Computation Theory
September 16–December 14, 1985