stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 218/83 JANUARI

J.A. BERGSTRA & J.W. KLOP

PROCESS ALGEBRA FOR COMMUNICATION AND MUTUAL EXCLUSION

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, Kruislaan 413, Amsterdam, The Netherlands.

The Mathematical Centre, founded 11th February 1946, is a non-profit institution for the promotion of pure and applied mathematics and computer science. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classification: 68B10, 68C01, 68D25, 68F20

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

Process algebra for communication and mutual exclusion*)

by

J.A. Bergstra & J.W. Klop

ABSTRACT

Within the context of an algebraic theory of processes we provide an equational specification of process coöperation. We consider three cases: process merging, merging with communication, and merging with mutual exclusion of critical sections. The term rewrite system behind the communication algebra is shown to be confluent and terminating (modulo its permutative reductions).

KEY WORDS & PHRASES: *nondeterministic processes, process algebra, merge, concurrency, communication, synchronisation, mutual exclusion, critical sections, term rewrite systems*

---

*) This report will be submitted for publication elsewhere.

0. INTRODUCTION

Let A be a finite collection (alphabet) of atomic actions, $\delta \in A$ a distinguished symbol denoting deadlock. Finite processes are generated from atomic processes in A using two operations

+ : nondeterministic choice and

. : sequential composition.

The following equational laws will hold for finite processes.

$x + y = y + x$	A1
$x + (y + z) = (x + y) + z$	A2
$x + x = x$	A3
$(x + y).z = x.z + y.z$	A4
$(x.y).z = x.(y.z)$	A5
$x + \delta = x$	A6
$\delta.x = \delta$	A7

The initial term algebra of these equations is $(A_\omega, +, ., \delta)$.

The main source of process algebra in this style is MILNER [9]. Exactly the above processes occur as finite uniform processes in DE BAKKER & ZUCKER [1], [2]. After adding an extra equation: $x.(y + z) = x.y + x.z$, one obtains a version of trace theory as described in REM [13].

For each $p \in A_\omega$ and $n \geqslant 1$ we have the approximation $(p)_n$ of p. This is inductively described by

$$(p + q)_n = (p)_n + (q)_n$$

$$(a)_n = a$$

$$(ax)_1 = a$$

$$(ax)_{n+1} = a(x)_n.$$

Interestingly, if $A_n = \{(p)_n \mid p \in A_\omega\}$ then $(A_n, +, ., \delta)$ is another model of the axioms A1,..,A7.

Infinite processes (A^∞) can be obtained as a projective limit of the structures A_n. Technically this means that A^∞ is the set of all sequences $p = (p_1, p_2, p_3, \ldots)$ with $p_i \in A_i$ and $p_i = (p_{i+1})_i$. The operations '+' and '.' are defined component-wise: $(p + q)_n = (p)_n + (q)_n$, $(p.q)_n = ((p)_n.(q)_n)_n$, thus obtaining the process algebra

$$(A^\infty, +, ., \delta).$$

On A^∞ a metric exists:

$$d(p,q) = \begin{cases} 0 \text{ if } p = q \\ 2^{-n} \text{ with n minimal such that } (p)_n \neq (q)_n \text{ if } p \neq q. \end{cases}$$

(A^∞, d) is a complete metric space, in fact it is the metric completion of (A_ω, d). The operations +,. are continuous.

(A^∞, d) was introduced in DE BAKKER & ZUCKER [1]. MILNER [10] uses charts modulo bisimulation (from PARK [12]) to obtain infinite processes from finite ones. Working with trace sets under the extra assumption $x.(y + z) = x.y + x.z$ this metric occurs in NIVAT [11]. In DE BAKKER et al. [3] the connections between (A^∞, d) and its corresponding trace space are investigated.

The processes discussed so far are provided with a bare minimum of structure. The crux of the algebraic method lies in algebraically defining new operators over the given process domains that will correspond to important process composition principles.

We will describe operators corresponding to the following three composition principles:

1. merging two processes
2. merging with communication
3. merging processes with mutual exclusion for critical sections.

1. MERGING TWO PROCESSES

The result of merging processes p and q is $p \| q$. For algebraic reasons (finite axiomatisability and ease of computation) an auxiliary operation $\lfloor\!\lfloor$ (left merge) is used. The process $p \lfloor\!\lfloor q$ stands for the result of merging p and q but taking the first step from p. Both operations $\|$ and $\lfloor\!\lfloor$ are specified on $(A_\omega, +, ., \delta)$ by this system of equations:

$x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$	M1
$a \lfloor\!\lfloor x = a.x$	M2
$ax \lfloor\!\lfloor y = a(x \| y)$	M3
$(x + y) \lfloor\!\lfloor z = x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	M4

Here x,y,z are variables over A_ω and a is a variable over A. Formally this is justified as a two-sorted logic with sorts A and A_ω where one sort is a subset of the other one.

The operations are extended to A^∞ as follows:

$$(p_1,p_2,\ldots) \| (q_1,q_2,\ldots) = ((p_1 \| q_1)_1, (p_2 \| q_2)_2, \ldots)$$

$$(p_1,p_2,\ldots) \lfloor\!\lfloor (q_1,q_2,\ldots) = ((p_1 \lfloor\!\lfloor q_1)_1, (p_2 \lfloor\!\lfloor q_2)_2, \ldots)$$

We omit the proof that these are indeed projective sequences (i.e. that $((p_{n+1} \| q_{n+1})_{n+1})_n = (p_n \| q_n)_n$). It also follows that $\|$ and $\lfloor\!\lfloor$ are continuous w.r.t. the metric d.

In BERGSTRA & KLOP [4] the operation $\lfloor\!\lfloor$ was introduced and a general existence theorem for solutions of equations of the form $x = \tau(x)$ in $(A^\infty, +, ., \|, \lfloor\!\lfloor, \delta)$ is derived.

2. MERGING WITH COMMUNICATION

Starting with $(A_\omega, +, ., \delta)$ plus a communication function $.|. : A \times A \to A$ which describes the effect of sharing (simultaneously executing) two atomic actions, three operations $\|$, $\lfloor\!\lfloor$ and $|$ are defined on A_ω. Here '|' extends

the given communication function. $\|$ and $\between$ coincide with the operators defined in Section 1 if the effect of $a|b$ is always δ (i.e. no two atomic actions communicate).

For the communication function three properties are required:

$a\|b = b\|a$	C1
$(a\|b)\|c = a\|(b\|c)$	C2
$\delta\|a = \delta$	C3

Then $\|$, $\between$ and $|$ are specified by:

$x\|\|y = x\between y + y\between x + x\|y$	CM1
$a\between x = a.x$	CM2
$(ax)\between y = a(x\|\|y)$	CM3
$(x+y)\between z = x\between z + y\between z$	CM4
$(ax)\|b = (a\|b).x$	CM5
$a\|(bx) = (a\|b).x$	CM6
$(ax)\|(by) = (a\|b).(x\|\|y)$	CM7
$(x+y)\|z = x\|z + y\|z$	CM8
$x\|(y+z) = x\|y + x\|z$	CM9

Let $C = \{a \in A \mid \exists b \in A \ \ (a|b) \neq \delta\}$. C contains all actions that must eventually communicate. So if process p runs on its own an action $c \in C$ cannot be executed and should be replaced by δ. This is modeled by the operation $\Delta: A_\omega \to A_\omega$ defined by

$$\Delta(a) = \begin{cases} a & \text{if } a \notin C \\ \delta & \text{if } a \in C \end{cases} \qquad \Delta 1$$

$$\Delta(x + y) = \Delta(x) + \Delta(y) \qquad \Delta 2$$

$$\Delta(x.y) = \Delta(x).\Delta(y) \qquad \Delta 3$$

Notice that $\Delta(x \| y) \neq \Delta(x) \| \Delta(y)$. Thus Δ is a homomorphism on $(A_\omega, +, ., \delta)$ but not on $(A_\omega, +, ., \|, \lfloor\!\lfloor, |, \delta)$.

An important observation concerning the difference between processes and trace sets is exhibited in the following example. Let $A = \{a, c_1, c_2, c, \delta\}$ and let $c_1 | c_2 = c$. All other communications result in δ. Now

$$\Delta(a(c_1 + c_2) \| c_1) = ac, \text{ and}$$

$$\Delta((ac_1 + ac_2) \| c_1) = ac + a\delta$$

so the second process $ac_1 + ac_2$ has a deadlock possibility and the first one, $a(c_1 + c_2)$, not.

As before $\|$, $\lfloor\!\lfloor$ and Δ can be extended to continuous operations on (A^∞, d).

This formalism includes both message passing and synchronisation. In [9] and [1,2] synchronisation is modeled by having $a|b = \tau$ whenever $a|b \neq \delta$, τ denoting a silent move. In HENNESSY & PLOTKIN [6] a definition corresponding to the equation $x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x + x|y$ occurs.

The heart of the system consists of A1,...,A7,C1,...,C3,CM1,...,CM9. We will call this system ACP, for algebra of communicating processes.

ACP seems to provide a concise formulation of the algebraic essence of communication. In view of this we review its structure in detail here. We will show that the new operators are indeed well-defined by CM1,...,CM9 over A1,...,A7 + C1,...,C3. We will rearrange ACP into a TRS (term rewrite system) which is shown to be confluent and strongly terminating modulo the permutative reductions A1 and A2. As a consequence we find that each term built from A by +, ., $\|$, $\lfloor\!\lfloor$, | can be proved equal to a unique term in A_ω by ACP.

Finally we prove that $\|$ is associative, as well as several other interesting identities in Theorem 2.2.

For technical reasons we associate to each $a \in A$ a unary operator a^* which acts as follows:

$$a^*x = a.x \ .$$

On the term system generated by A, $+$, $.$, $\|$, $\between$, $|$, a^* ($a \in A$) we introduce two norms $|.|$ and $\|.\|$. Here intuitively $|S|$ computes an upper bound for the path lengths in S and $\|S\|$ computes an upper bound of the number of (nontrivial) summands in which S decomposes.

$$\begin{array}{ll}
|a| = 1 & \|a\| = 1 \\
|a^*x| = 1 + |x| & \|a^*x\| = 1 \\
|x.y| = |x| + |y| & \|x.y\| = \|x\| \\
|x + y| = \max(|x|, |y|) & \|x + y\| = \|x\| + \|y\| \\
|x|y| = |x| + |y| - 1 & \|x|y\| = \|x\|.\|y\| \\
|x \between y| = |x| + |y| & \|x \between y\| = \|x\| \\
|x \| y| = |x| + |y| & \|x \| y\| = \|x\| + \|y\| + \|x\|.\|y\| \ .
\end{array}$$

Now consider the following term rewrite system RACP:

$$\begin{array}{ll}
x + y \to y + x & \text{RA1} \\
x + (y + z) \to (x + y) + z & \text{RA2} \\
(x + y) + z \to x + (y + z) & \text{RA2'} \\
x + x \to x & \text{RA3} \\
(x + y).z \to x.z + y.z & \text{RA4} \\
a.x \to a^*x & \text{RA5'} \\
(a^*x).y \to a^*(x.y) & \text{RA5} \\
x + \delta \to x & \text{RA6} \\
\delta^*x \to \delta & \text{RA7} \\
x \| y \to x \between y + y \between x + x|y & \text{RCM1} \\
a \| x \to a^*x & \text{RCM2} \\
(a^*x) \between y \to a^*(x \| y) & \text{RCM3} \\
(x + y) \between z \to x \between z + y \between z & \text{RCM4} \\
a|b \to c_{a,b} & \text{RCM5'} \\
(a^*x)|b \to c^*_{a,b}.x & \text{RCM5} \\
a|b^*x \to c^*_{a,b}.x & \text{RCM6} \\
(a^*x)|(b^*y) \to c^*_{a,b}(x \| y) & \text{RCM7} \\
(x + y)|z \to x|z + y|z & \text{RCM8} \\
x|(y + z) \to x|y + x|z & \text{RCM9}
\end{array}$$

In RCM5',...,RCM7 the symbol $c_{a,b}$ denotes $a|b \in A$. The axioms C1,C2,C3 translate into the commutativity and associativity of c and $c_{\delta,a} = \delta$ for all $a \in A$.

Convertibility in RACP is denoted by $=_R$.

2.1. THEOREM. For all ACP-terms without variables:

(i) $ACP \vdash S = T \iff S =_R T$

(ii) $ACP \vdash S = S'$ for some S' not containing $\|, \lfloor\!\lfloor, |$.

(iii) $ACP \vdash S' = S'' \iff A1,...,A7 \vdash S' = S''$ for S',S" not containing $\|, \lfloor\!\lfloor, |$.

(iv) $S.(T.U) =_R (S.T).U$

(v) RACP is weakly confluent (has the weak Church-Rosser property), working modulo A1 and A2.

(vi) RACP is strongly terminating, modulo A1 and A2.

(vii) RACP is confluent (has the Church-Rosser property).

PROOF. We start with (vi) and we introduce the auxiliary notion of the multiset of direct subterms DS(T) of a term T:

$$DS(a) = \emptyset$$
$$DS(a*x) = DS(x)$$
$$DS(x + y) = DS(x) \dot\cup DS(y)$$
$$DS(x \square y) = \{x \square y\} \dot\cup DS(x) \dot\cup DS(y) \quad \text{(here } \square \text{ is } ., \|, \lfloor\!\lfloor, \text{ or } |\text{)}$$

Here $\dot\cup$ denotes the multiset union. Let [S] be the mapping from terms to $\omega\times\omega$ defined by

$$[S] = (|S|, \|S\|).$$

This mapping is extended to multisets over terms, thus producing multisets over $\omega\times\omega$:

$$[V] = \{[S] \mid S \in V\}.$$

On $\omega \times \omega$ there is the lexicographic well-ordering $<$ which induces a well-ordering $\ll$ on finite multisets over $\omega \times \omega$. We now observe that along a reduction path

$$T_0 \xrightarrow[R_0]{} T_1 \xrightarrow[R_1]{} T_2 \xrightarrow[R_2]{} \cdots$$

we have

$$[DS(T_i)] \gg [DS(T_{i+1})] \text{ if } R_i \text{ is not RA1, RA2, RA2', and}$$

$$[DS(T_i)] = [DS(T_{i+1})] \text{ if } R_i \text{ is RA1, RA2 or RA2'.}$$

From this observation strong termination of RACP modulo A1 and A2 follows.

Instead of a proof of the observation we provide two characteristic examples.

(1) $a.x \to a*x$. Then:

$[DS(a.x)] = [a.x] \dot{\cup} [DS(x)]$ and $[DS(a*x) = [DS(x)]$.

Now $[a.x]$ majorizes each element of $[DS(x)]$ because

$[S] \in [DS(x)] \Rightarrow |S| \leqslant |x| \Rightarrow |S| < |a.x|$. Hence $[DS(a.x)] \gg [DS(a*x)]$.

(2) $x||y \to x \between y + y \between x + x|y$. Then:

$[DS(x||y)] = [x \| y] \dot{\cup} [DS(x)] \dot{\cup} [DS(y)]$ and

$$[DS(x \between y + y \between x + x|y)] = [x \between y] \dot{\cup} [DS(x)] \dot{\cup} [DS(y)] \dot{\cup}$$
$$[y \between x] \dot{\cup} [DS(x)] \dot{\cup} [DS(y)] \dot{\cup}$$
$$[x|y] \dot{\cup} [DS(x)] \dot{\cup} [DS(y)].$$

Again $[x||y]$ majorizes all of $[x \between y]$, $[y \between x]$, $[x|y]$, $[DS(x)]$, $[DS(y)]$, the first three in width and the second two in depth.

An alternative proof of termination can be given by ranking all occurrences of $||, \between, |$ by the $|.|$-norm of the term of which they are the leading operator. Using this extended set of operators a recursive path ordering can be found which is decreasing in all rewrite steps except the first three (RA1,RA2,RA2'). See DERSHOWITZ [5].

Proof of (v). RACP is weakly confluent modulo $\sim$, the congruence generated by A1 and A2. (We are here working in congruence classes and reductions have the form $[S]^{\sim} \to [S']^{\sim}$ whenever $S \to S'$.) This is a matter of some 400 straightforward verifications. (Of course left to the reader as an exercise.)

Proof of (vii). Working modulo $\sim$ RACP is strongly terminating in view of (vi). Now combining (v) and (vi) and using Newman's Lemma (see [8], Lemma 5.7.(1) or [7] where more information about reduction modulo equivalence can be found) we find that RACP is confluent modulo $\sim$ and consequently it is confluent because the reductions generating $\sim$ are symmetric.

Proof of (ii). This follows immediately from (vi).

Proof of (iv). First one proves the associativity of . for terms not containing $\|, \between, |$, using induction on the structure of S. The result then immediately follows using (ii).

Proof of (i). $S =_R T \Rightarrow ACP \vdash S = T$ is immediate. For the other direction one uses (iv).

Proof of (iii). If $ACP \vdash S' = S''$ then by (i) $S' =_R S''$ and by (vii) for some S"': $S' \twoheadrightarrow S'''$ and $S'' \twoheadrightarrow S'''$ (here $\twoheadrightarrow$ is the transitive reflexive closure of $\rightarrow$). Now because S' and S" are free of $\|, \between, |$ we see that $S' \twoheadrightarrow S''' \twoheadleftarrow S''$ is just a proof in A1,...,A7.

□

2.2. THEOREM. The following identities hold in $(A_\omega, +, ., \|, \between, |, \delta)$:

(1) $x|y = y|x$

(2) $x\|y = y\|x$

(3) $x|(y|z) = (x|y)|z$

(4) $(x \between y)\| z = x \between (y\|z)$

(5) $x|(y \between z) = (x|y) \between z$

(6) $x\|(y\|z) = (x\|y)\|z$.

PROOF. All proofs use induction on the structure of x,y,x written as a term over (A,+,.), which is justified by Theorem 2.1.(2). We write

$$x = \sum_i a_i x_i + \sum_j a'_j$$

$$y = \sum_k b_k y_k + \sum_\ell b'_\ell$$

$$z = \sum_m c_m z_m + \sum_n c'_n .$$

(1) and (2) are proved in a simultaneous induction.

$$x|y = \sum (a_i|b_k)(x_i\|y_k) + \sum (a_i|b'_\ell)x_i + \sum (a'_j|b_k)y_k + \sum (a'_j|b'_\ell) =$$

$$= \sum (b_k|a_i)(y_k\|x_i) + \sum (b'_\ell|a_i)x_i + \sum (b_k|a'_j)y_k + \sum (b'_\ell|a'_j) =$$

$$y|x.$$

Here we use C1 and the induction hypothesis for $x_i \| y_k = y_k \| x_i$.

(2): $x \| y = x \mathbin{\perp\!\!\!\perp} y + y \mathbin{\perp\!\!\!\perp} x + x | y = y \mathbin{\perp\!\!\!\perp} x + x \mathbin{\perp\!\!\!\perp} y + y | x = y \| x.$

The proof of (3),...,(6) is also done using one simultaneous induction.

(3): Write $x = x' + x''$ where $x' = \sum a_i x_i$ and $x'' = \sum a'_j$. Likewise $y = y' + y''$ and $z = z' + z''$. Then

$$x|(y|z) = x'|(y'|z') + x'|(y''|z') + x'|(y'|z'') + x'|(y''|z'') + x''|(y'|z') + x''|(y''|z') + x''|(y'|z'') + x''|(y''|z'').$$

Now $$x'|(y'|z') = \sum (a_i|(b_k|c_m))(x_i\|(y_k\|z_m)) = \sum ((a_i|b_k)|c_m)((x_i\|y_k)\|z_m) = (x'|y')|z'.$$

Here we used C2 and the induction hypothesis for (6). The other summands of $x|(y|z)$ are treated similarly. Hence $x|(y|z) = (x|y)|z$.

(4): $$(x \mathbin{\perp\!\!\!\perp} y) \mathbin{\perp\!\!\!\perp} z = ((\sum a_i x_i + \sum a'_j) \mathbin{\perp\!\!\!\perp} y) \mathbin{\perp\!\!\!\perp} z = (\sum a_i (x_i \| y) + \sum a'_j . y) \mathbin{\perp\!\!\!\perp} z =$$
$$\sum a_i ((x_i \| y) \| z) + \sum a'_j (y \| z) = \text{(induction hypothesis on (6))}$$
$$\sum a_i (x_i \| (y \| z)) + \sum a'_j (y \| z) =$$
$$(\sum a_i x_i + \sum a'_j) \mathbin{\perp\!\!\!\perp} (y \| z) =$$
$$x \mathbin{\perp\!\!\!\perp} (y \| z).$$

(5): Let $x = x' + x''$ and $y = y' + y''$ as in the proof of (3). Then

$$x|(y \mathbin{\perp\!\!\!\perp} z) = x'|(y' \mathbin{\perp\!\!\!\perp} z) + x'|(y'' \mathbin{\perp\!\!\!\perp} z) + x''|(y' \mathbin{\perp\!\!\!\perp} z) + x''|(y'' \mathbin{\perp\!\!\!\perp} z).$$

Now $$x'|(y' \mathbin{\perp\!\!\!\perp} z) = (\sum a_i x_i)|(\sum b_k y_k) \mathbin{\perp\!\!\!\perp} z) = (\sum a_i x_i)|(\sum b_k (y_k \| z)) =$$
$$\sum (a_i|b_k)(x_i \| (y_k \| z)) = \text{(induction hypothesis on (6))}$$
$$\sum (a_i|b_k)((x_i \| y_k) \| z) = (\sum (a_i|b_k)(x_i \| y_k)) \mathbin{\perp\!\!\!\perp} z =$$
$$(x'|y') \mathbin{\perp\!\!\!\perp} z.$$

The other three summands are treated similarly. Hence $x|(y \mathbin{\perp\!\!\!\perp} z) = (x|y) \mathbin{\perp\!\!\!\perp} z$.

(6) Write $A_x(y,z) = x \perp\!\!\!\perp (y \| z)$ and $B_x(y,z) = (y|z) \perp\!\!\!\perp x$. Then:

$x || (y \| z) = x \perp\!\!\!\perp (y \| z) + (y || z) \perp\!\!\!\perp x + x | (y || z) =$

$A_x(y,z) + (y \perp\!\!\!\perp z) \perp\!\!\!\perp x + (z \perp\!\!\!\perp y) \perp\!\!\!\perp x + (y|z) \perp\!\!\!\perp x + x | (y \perp\!\!\!\perp z) + x | (z \perp\!\!\!\perp y) + x | (y|z) =$

$A_x(y,z) + y \perp\!\!\!\perp (z \| x) + z \perp\!\!\!\perp (y \| x) + B_x(y,z) + (x|y) \perp\!\!\!\perp z + (x|z) \perp\!\!\!\perp y + x | (y|z) =$

$A_x(y,z) + A_y(z,x) + A_z(y,x) + B_x(y,z) + B_z(y,x) + B_y(x,z) + x | (y|z).$ (*)

Also $(x || y) \| z = z || (x \| y) = z || (y \| x) =$

$A_z(y,x) + A_y(x,z) + A_x(y,z) + B_z(y,x) + B_x(y,z) + B_y(z,x) + z | (y|x) =$

$A_x(y,z) + A_y(x,z) + A_z(y,x) + B_x(y,z) + B_y(z,x) + B_z(y,x) + (x|y) | z$

which equals (*) using the commutativity of the A's and B's and the induction hypothesis on $(x|y)|z$.

3. MERGING WITH MUTUAL EXCLUSION OF CRITICAL REGIONS

From $(A_\omega, +, ., \delta)$ we derive $((A \cup \underset{\rightarrow}{A})_\omega, +, ., \delta)$ by adding for each atomic action a A, a new copy $\underset{\rightarrow}{a}$. This $\underset{\rightarrow}{a}$ stands for: do a and, if possible, immediately proceed with an action of the same process (i.e. do not allow interference of other processes until the "lowest" process containing a has terminated).

Now this algebra is enriched with operators $\|$ and $\lfloor\!\lfloor$ and a really new one: $p \mapsto \underline{p}$. Here $\underline{p}$ stands for: execute p as a critical region (i.e. do not allow interference of other steps).

The axioms for $((A \cup \underset{\rightarrow}{A})_\omega, +, ., \|, \lfloor\!\lfloor, \_, \delta)$ are as follows:

$x \| y = x \lfloor\!\lfloor y + y \lfloor\!\lfloor x$	ME1
$a \lfloor\!\lfloor x = a.x$	ME2
$\underset{\rightarrow}{a} \lfloor\!\lfloor x = a.x$	ME3
$(ax) \lfloor\!\lfloor y = a.(x \| y)$	ME4
$(\underset{\rightarrow}{a}x) \lfloor\!\lfloor y = \underset{\rightarrow}{a}.(x \lfloor\!\lfloor y)$	ME5
$\underline{a} = a$	ME6
$\underset{\rightarrow}{\underline{a}} = a$	ME7
$\underline{a.x} = \underline{a}.\underline{x}$	ME8
$\underset{\rightarrow}{\underline{a}}.x = \underset{\rightarrow}{\underline{a}}.\underline{x}$	ME9
$\underline{x + y} = \underline{x} + \underline{y}$	ME10
$\underset{\rightarrow}{\underline{\delta}} = \delta$	ME11

We omit a prooftheoretic analysis of these equations, but state without proof these useful facts:

$$\underline{x.y} = \underline{\underline{x}.y}$$

$$\underline{x \| y} = \underline{x}.y + \underline{y}.x \ .$$

In general $\underline{x \| y} \neq \underline{x} \| \underline{y}$ however.

Suppose one considers terms generated by A, +, ., || and _. These have an intuitively clear meaning and should be given a semantics in $(A_\omega, +, ., \delta)$.

Note that $\underline{x}$ frequently is denoted by <x> in the litterature on parallel programs. On the other hand, atomic actions of the form $\underset{\rightarrow}{a}$ are not common and should not enter the definition of a programming system; they are here only used as a means of calculation and are eliminated in the second step below (i.e. by applying the homomorphism Φ).

As an intermediate step a semantics in $((A \cup \underset{\rightarrow}{A})_\omega, +, ., \delta)$ is found:

$$\mathcal{M}^*(a) = a$$
$$\mathcal{M}^*(p + q) = \mathcal{M}^*(p) + \mathcal{M}^*(q)$$
$$\mathcal{M}^*(p.q) = \mathcal{M}^*(p).\mathcal{M}^*(q)$$
$$\mathcal{M}^*(\underline{p}) = \underline{\mathcal{M}^*(p)}$$
$$\mathcal{M}^*(p||q) = \mathcal{M}^*(p) \,||\, \mathcal{M}^*(q)$$

Then the canonical mapping (homomorphism)

$$\Phi: ((A \cup \underset{\rightarrow}{A})_\omega, +, ., \delta) \longrightarrow (A_\omega, +, ., \delta)$$

generated by $\Phi(a) = \Phi(\underset{\rightarrow}{a}) = a$ can be used to provide a semantics for +,.,||,_ terms over A:

$$\mathcal{M}(p) = \Phi(\mathcal{M}^*(p)).$$

Note that $\mathcal{M}^*$ is a homomorphism w.r.t. <u>all</u> program constructions +,.,||,_, as should be expected from a denotational (or rather compositional) semantics; Φ is a homomorphism only w.r.t. +,.,δ. Hence $\mathcal{M}$ is a homomorphism w.r.t +,.,δ. However, $\mathcal{M}$ does not act as a homomorphism w.r.t. ||. (I.e. $\mathcal{M}(p\|q) \neq \mathcal{M}(p)\|\,\mathcal{M}(q)$, in general.)

REFERENCES

[1] DE BAKKER, J.W. & J.I. ZUCKER,
Denotational semantics of concurrency,
Proc. 14th ACM Symp. on Theory of Computing, pp.153-158, 1982.

[2] DE BAKKER, J.W. & J.I. ZUCKER,
Processes and the denotational semantics of concurrency,
Department of Computer Science Technical Report IW 209/82,
Mathematisch Centrum, Amsterdam 1982.

[3] DE BAKKER, J.W., J.A. BERGSTRA, J.W. KLOP & J.-J.CH. MEYER,
Linear time and branching time semantics for recursion with merge,
Department of Computer Science Technical Report IW 211/82,
Mathematisch Centrum, Amsterdam 1982.

[4] BERGSTRA, J.A. & J.W. KLOP,
Fixed point semantics in process algebras,
Department of Computer Science Technical Report IW 206/82,
Mathematisch Centrum, Amsterdam 1982.

[5] DERSHOWITZ, N.,
Orderings for term-rewriting systems,
Theoretical Computer Science 17 (1982)pp.279-301.

[6] HENNESSY, M., & G. PLOTKIN,
A term model for CCS,
Proc. 9th MFCS, Poland (1980), LNCS 88.

[7] HUET, G.,
Confluent reductions: Abstract properties and applications to term rewriting systems,
J.ACM 27 (4) (1980) pp.797-821.

[8] KLOP, J.W.,
Combinatory Reduction Systems,
Mathematical Centre Tracts 127, Mathematisch Centrum, Amsterdam
1980.

[9] MILNER, R.,
A Calculus of Communicating Systems,
Springer LNCS 92, 1980.

[10] MILNER, R.,
A Complete Inference System for a class of Regular Behaviours,
Preprint, Edinburgh 1982.

[11] NIVAT, M.,
Infinite words, infinite trees, infinite computations,
Foundations of Computer Science III.2 (J.W. de Bakker & J. van
Leeuwen, eds.) pp.3-52, Mathematical Centre Tracts 109, Mathe-
matisch Centrum, Amsterdam 1979.

[12] PARK, D.M.R.,
Concurrency and automata on finite sequences,
Computer Science Department, University of Warwick (1981).

[13] REM, M.,
Partially ordered computations, with applications to VLSI design,
To appear in the proc. of the 4th Advanced Course on Foundations of Computer Science, Amsterdam, June 1982.

6gF11
6gF12
6gF32
6gF43