stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 206/82 AUGUSTUS

J.A. BERGSTRA & J.W. KLOP

FIXED POINT SEMANTICS IN PROCESS ALGEBRAS

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

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

1980 Mathematics subject classification: 68D30, 68D35, 68F20

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

FIXED POINT SEMANTICS IN PROCESS ALGEBRAS *)

by

J.A. Bergstra **) & J.W. Klop

ABSTRACT

We consider nondeterministic uniform processes as introduced by de Bakker and Zucker, with composition, union, merge, μ-operator and semantics in metric spaces. We represent a collection of such processes as the projective limit of collections of finite processes. Here a finite process is generated from a set of atomic actions by means of the operations sequential composition and nondeterministic union. The process algebra thus obtained is augmented by an operation 'left merge' in terms of which the usual merge operator is defined.

We show the existence of solutions of equations $x = s(x)$, where $s(x)$ is a μ-free expression. This yields the existence of a fixed point semantics for process expressions containing the μ-operator. The proof amounts to a combinatorial analysis showing that certain iteration sequences stabilize on each finite level.

KEY WORDS & PHRASES: *nondeterministic processes, process algebra, merge operator, μ-expressions, fixed point semantics*

---

*) This report will be submitted for publication elsewhere.

**) Department of Computer Science, University of Leiden, Wassenaarseweg 80, 2300 RA LEIDEN, The Netherlands

0. INTRODUCTION

An important part of the development of programming languages is aimed at the *description of processes*; cf. HOARE [4] (CSP), MILNER [6] (CCS), PETRI [8], DENNIS e.a. [3], KAHN & MACQUEEN [5].

As a result, there has been quite some effort in the area of theoretical Computer Science to provide such descriptions of processes, or proces notations, of a *semantics*, denotational, operational or otherwise. This endeavour has led to a substantial body of general theory about processes; we refer to e.g. PRATT [10,11], NIVAT [7], DE BAKKER & ZUCKER [1,2].

In the present paper we will be concerned with the mechanism introduced and studied by DE BAKKER & ZUCKER [1,2]: from a set of atomic actions, finite processes are generated by means of composition and nondeterministic union; this set of finite processes is made into a metrical space and metrically completed to yield the collection of all processes. Finally the collection of processes is enriched with the merge operation.

In fact, the construction of DE BAKKER & ZUCKER is a solution of some domain equation, requiring a careful choice of the right metric. This sophisticated construction, related to work of NIVAT [7], is general in the sense that it applies to several domain equations, as described in [1,2]. Here we will only deal with the case of *uniform* processes as they are called in [1,2].

We follow another presentation, in which the collection of uniform processes results as a *projective limit* A_∞ instead of the metrical completion P in [1,2]. For uniform processes, the relation between the two approaches is clear (see our Concluding Remarks): this projective limit A_∞ is (isomorphic to) a certain compact subspace of P.

A_∞ is a projective limit of collections A_n consisting of finite (also in the sense of 'finitely branching') processes, of depth not exceeding n, which are generated from a set A of atomic actions using concatenation ($\cdot$) and union (+). For such finite processes, we obtain a *finite axiomatization* using a new primitive operation, 'left merge' ($\between$), in terms of which merge ($\|$) is defined. The use of $\between$ will be crucial, in a technical sense: not only it simplifies the algebraic presentation, but also it makes certain induction proofs possible.

In this framework we consider a specific problem, which arises as follows. DE BAKKER & ZUCKER [1,2] introduce μ-expressions $\mu x.s(x)$ to denote

least fixed points, or rather, to denote a certain solution of the equation $x = s(x)$. The problem is now to show that $\lim_{n\to\infty} s^n(q)$ exists, for certain 'starting' processes q. This is the problem which is solved here; it turns out to be sufficient to do this initially for the case that s(x) contains itself no μ-expressions. DE BAKKER & ZUCKER [1,2] give an indirect solution: instead of solving $x = s(x)$ they solve $x = \varepsilon s(x)$ where ε is a certain distinguished process, comparable with an 'idling' step of a process (cf. 'τ' in MILNER [6], or the 'hiaton' of PARK [9]). A certain solution of this last equation is then taken as the semantics of μx.s(x).

It is worth emphasizing that the sequel does not use the benefits of theory about complete partial orders (cpo's). This is so by necessity: there does not exist a suitable partial order on the set of uniform processes considered below. For the definition of 'suitable' and the simple proof, see Proposition 3.6.1.

Finally, we will give a summary of the paper.

1. PROCESS ALGEBRAS

In this section we introduce process algebras and their projections, fix some terminology and notations, and establish some useful algebraical identities valid in process algebras.

1.1. Process algebras: preliminaries.

1.1.1. DEFINITION. Let $A = \{a_i | i \in I\}$ be some set of atomic "actions".

A *process algebra over* A is a structure $\mathcal{A} = \langle A, +, \cdot, \between, \underline{a_i} (i \in I) \rangle$ where A is a set containing A, the $\underline{a_i}$ are constant symbols corresponding to the $a_i \in A$, and + (*union*), $\cdot$ (*concatenation* or *composition*), $\between$ (*left merge*) satisfy for all $x,y,z \in A$ and $a \in A$ the following axioms:

PA1 $x+y = y+x$

PA2 $x+(y+z) = (x+y)+z$

PA3 $x+x = x$

PA4 $(xy)z = x(yz)$

PA5 $(x+y)z = xz+yz$

PA6 $(x+y) \between z = x \between z + y \between z$

PA7 $ax \between y = a(x \between y + y \between x)$

PA8 $a \between y = ay$

1.1.1.1. NOTATION. We write xy instead of $x \cdot y$ and a instead of $\underline{a}$.

1.1.1.2. REMARK. Note the absorption law for + and note that there is no left distributive law $z(x+y) = zx+zy$. Also there is no '0' satisfying $x+0 = x$, $0x = x0 = x$, since this would lead to

$$xy = (x+0)y = xy+0y = xy+y,$$

contrary to our intentions (to have the 'isomorphism' described in Section 4). (However, see Section 3.)

1.1.2. DEFINITION. The operator $\|$ (*merge*) is defined by

$$x \| y = x \between y + y \between x.$$

1.1.3. PROPOSITION. *For all process algebras:*

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

(ii) $\sum_{i=1}^{n} a_i x_i \,\|\, \sum_{j=1}^{m} b_j y_j = \sum_{i=1}^{n} a_i (x_i \| \sum_{j=1}^{m} b_j y_j) + \sum_{j=1}^{m} b_j (y_j \| \sum_{i=1}^{n} a_i x_i)$.

PROOF. Obvious. □

1.1.3.1. REMARK. Without $\between$, it does not seem possible to avoid the cumbersome explicit sum formula of 1.1.3(ii) in favour of simpler 'algebraic' axioms as for $\between$ (PA6,7,8).

Note that the set of axioms PA1,...,8 constitutes a finite axiomatization of process algebras (anyway if 'a' in PA7,8 is read as a metavariable over A; otherwise we need axioms PA7,8 for every $a \in A$), whereas an axiomatization in terms of $\|$ by means of 1.1.3(ii) would be infinite.

We conjecture that process algebras without $\between$ and using $\|$, are not finitely axiomatizable. (Not even in the case of finite A.)

Let us remark here in advance that the better algebraic properties of $\between$ as compared to $\|$, are only a side benefit; the real advantage is in the projection property displayed in Proposition 1.2.3(iv).

Mainly we will be interested in the *initial algebra* (or *term model*) $\mathcal{A}$ determined by PA1,...,8. $\mathcal{A}$ can be thought of as the set of closed terms built from the constants a_i and the operations $+, \cdot, \between$, after dividing out the equivalence relation generated by the axioms. Alternatively, $\mathcal{A}$ can be thought of as the set of terms with the property that PA3,5,6,7,8, as rewrite rules from left to right, can no longer be applied, modulo the equivalence relation generated by PA1,2,4. In fact, one easily establishes the following

1.1.4. PROPOSITION. (Representation of elements of $\mathcal{A}$)
Modulo equivalence, $\mathcal{A}$ is inductively generated as follows:

$$x_i \in \mathcal{A},\ a_i \in A\ (i=1,\ldots,n),\ b_j \in A\ (j=1,\ldots,m) \implies$$
$$(\sum_{j=1}^{m} b_j + \sum_{i=1}^{n} a_i x_i) \in \mathcal{A}.$$

1.1.5. REMARK. The elements of A can also be represented as finite 'commutative' trees, as displayed in the following example.

1.1.6. EXAMPLE. $bab||ab =$

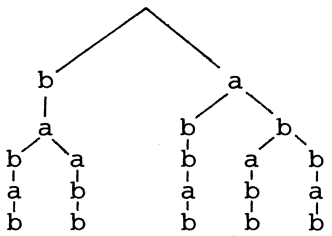
$bab\llcorner ab + ab\llcorner bab =$

$b(ab\llcorner ab + ab\llcorner ab) + a(b\llcorner bab + bab\llcorner b) =$

$b(ab\llcorner ab) + a(bbab + b(ab\llcorner b + b\llcorner ab)) =$

$b(a(b\llcorner ab + ab\llcorner b)) + a(bbab + b(abb + bab)) =$

$b(a(bab + abb)) + a(bbab + b(abb + bab)) =$



On the elements $x \in A$ we define the following norm $\nu(x)$, which intuitively is the minimum of the length of the 'branches' of the tree of x (as in the preceding example).

1.1.7. DEFINITION. For $x \in A$, we define $\nu(x)$ by:

(i) $\nu(a) = 1$

(ii) $\nu(x+y) = \min\{\nu(x), \nu(y)\}$

(iii) $\nu(ax) = 1 + \nu(x)$.

The following proposition says that merging will certainly not lead to shorter branches. (In fact, the proposition holds with '>' instead of '$\geq$', but we will not need that.)

1.1.8. PROPOSITION. $\nu(x||y) \geq \nu(x), \nu(y)$. □

In the next proposition we establish some useful identities valid in A (needed in Section 2), again without the routine proofs. First some more notation:

1.1.9. NOTATION. (i) $x^1 \equiv x$; $x^{n+1} \equiv xx^n$ $(n \geq 1)$

(ii) $x^{\underline{1}} \equiv x$; $x^{\underline{n+1}} \equiv x||x^{\underline{n}}$ $(n \geq 1)$

(This notation makes in fact only sense after Proposition 1.1.10(i).)

1.1.10. PROPOSITION. *In $\mathcal{A}$ the following identities are valid:*

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

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

(iii) $x \| y \| z = x \between (y \| z) + y \between (x \| z) + z \between (x \| y)$

(iv) $x_1 \| x_2 \| \ldots \| x_n =$
$x_1 \between (x_2 \| \ldots \| x_n) + x_2 \between (x_1 \| x_3 \| \ldots \| x_n) + \ldots + x_n \between (x_1 \| \ldots \| x_{n-1})$
$(n \geq 2)$

(v) $\underline{\underline{x^{n+1}}} = x \between \underline{\underline{x^n}}$ $(n \geq 1)$.

PROOF. (v) follows directly from (iv), which generalizes (iii); (iv) follows via simple algebraic manipulations from (i) and (ii). (i) and (ii) can be proved simultaneously using induction on the structure of $x,y,z \in \mathcal{A}$ according to Proposition 1.1.4. □

1.2. Projections and projective sequences.

1.2.1. DEFINITION. (i) On $\mathcal{A}$ we define for each $n \geq 1$ the *projection* $(\ )_n : \mathcal{A} \to \mathcal{A}$ as follows:

$$(a)_n = a$$
$$(ax)_1 = a; \quad (ax)_{n+1} = a(x)_n$$
$$(x+y)_n = (x)_n + (y)_n.$$

(ii) $\mathcal{A}_n = \{(x)_n \mid x \in \mathcal{A}\}$.

(iii) Instead of $(x)_n = (y)_n$ we will also say: $x = y$ *modulo* n.
(Intuitively, $(\ )_n$ cuts of the 'tree' of x at level n.)

1.2.2. EXAMPLE. Modulo 3 we have:

$$[(a^3 \| b^3) + a^3] \| b^3 = a^3 \| b^3 = (a+b)^3.$$

The following proposition is easily established. Note especially the occurrences of $n-1$ in (iii) and (iv):

1.2.3. PROPOSITION. *For all $x,y \in \mathcal{A}$:*

(i) $((x)_n)_m = (x)_{\min(n,m)}$ $(n,m \geq 1)$

(ii) $(x+y)_n = ((x)_n + (y)_n)_n$ $(n \geq 1)$

(iii) $(xy)_n = ((x)_n (y)_{n-1})_n$ $\quad (n \geq 2)$

(iv) $(x \between y)_n = ((x)_n \between (y)_{n-1})_n$ $\quad (n \geq 2)$

(v) $(x \| y)_n = ((x)_n \| (y)_n)_n$ $\quad (n \geq 1)$

(vi) $(xy)_1 = (x)_1$

(vii) $(x \between y)_1 = (x)_1$. □

1.2.3.1. REMARK. Note that the A_n are also process algebras over A, with the definition:

$$x \cdot_n y = (xy)_n$$

$$x \between_n y = (x \between y)_n.$$

1.2.4. DEFINITION. Let $q_i \in A$ $(i \geq 1)$. Then the sequence $q_1, q_2, \ldots$ is called *projective* iff for all i:

$$q_i = (q_{i+1})_i.$$

1.2.5. DEFINITION. A_∞ is the *projective limit* of the A_n $(n \geq 1)$; the elements of A_∞ are the projective sequences. A_∞ is a process algebra over A where the operations are defined component-wise.

1.2.6. EXAMPLE. (i) $(a, a+a^2, a+a^2+a^3, \ldots) \in A_\infty$

(ii) $(a, a^2, a^3, \ldots) \cdot (b, b^2, b^3, \ldots) = ((ab)_1, (a^2 b^2)_2, (a^3 b^3)_3, \ldots) =$
$(a, a^2, a^3, \ldots)$.

(iii) $(a, a+a^2, a+a^2+a^3, \ldots) \cdot (b, b+b^2, b+b^2+b^3, \ldots) =$
$(a, ab+a^2, a(b+b^2)+a^2 b+a^3, \ldots)$.

We will consider A_∞ again in Section 3,4.

2. ITERATION SEQUENCES

In this section we will show that every iteration sequence q, s(q), s(s(q)),... must eventually be constant ('stabilizes') modulo n, for every $n \geq 1$.

2.1. DEFINITION. The set EXP of *process expressions* (over A) is defined by:

$$s ::= \underline{a},\underline{b},\underline{c},\ldots \mid x,y,z,\ldots \mid s_1+s_2 \mid s_1 s_2 \mid s_1 \between s_2.$$

Here $a,b,c,\ldots \in A$ and $x,y,z,\ldots$ are variables.
(We will use the notation conventions already adopted in Section 1.)

2.2. DEFINITION. (i) Let $s(x) \in EXP$ be an expression containing no other variables than x. Let $q \in \mathcal{A}$. Then the sequence

$$q,\ s(q),\ s(s(q)),\ldots,s^k(q),\ldots$$

is called the *iteration sequence generated by* $s(x)$ *from* q.

(ii) The sequence $q_1, q_2,\ldots,q_k,\ldots$ $(q_i \in \mathcal{A},\ i \geq 1)$ is said to *stabilize modulo* n iff the sequence stabilizes in $\mathcal{A}_n$, i.e. iff

$$(q_1)_n,\ (q_2)_n,\ldots,\ (q_k)_n,\ldots$$

is eventually constant.

In order to prove the main theorem of this Section, we need some propositions.

2.3. PROPOSITION. *For every* $q \in \mathcal{A}$ *and* $n \geq 1$, *the iteration sequence*

$$q,\ q||q,\ q\|q||q,\ \ldots\ ,\ \overline{\overline{q}}^k,\ \ldots$$

stabilizes modulo n.

PROOF. Induction on n.
Basis: n=1. One easily computes:

$$(q)_1 = \Sigma a_i = (q||q)_1 = \ldots = (\overline{\overline{q}}^k)_1 = \ldots$$

for some sum Σa_i.
Induction step. Suppose the proposition is proved for n-1. By Proposition 1.1.10(v),

$$\overline{\overline{q}}^{k+1} = q \between \overline{\overline{q}}^k.$$

By Proposition 1.2.3(iv),

$$(\overline{\overline{q}}^{k+1})_n = (q \between \overline{\overline{q}}^k)_n = ((q)_n \between (\overline{\overline{q}}^k)_{n-1})_n.$$

By the induction hypothesis, $(\overline{\overline{q}}^k)_{n-1} = p$ for some fixed p for all but finitely many k.

Hence the sequence stabilizes indeed modulo n, viz. in $((q)_n \between p)_n$. □

The next two propositions generalize the preceding one considerably.

2.4. PROPOSITION. *Let* A *be finite. Let* $q_1, q_2, \ldots$ *be a sequence in* A *such that for all* $i \geq 1$:

$$q_{i+1} = q_i || r_i \text{ for some } r_i.$$

Then the sequence $q_1, q_2, \ldots$ *stabilizes modulo* n.

PROOF. By assumption, $q_k = q_1 || r_1 || r_2 || r_3 || \ldots || r_{k-1}$ $(k \geq 2)$, hence by Proposition 1.2.3(v):

$$(q_k)_n = ((q_1)_n || (r_1)_n || \ldots || (r_{k-1})_n)_n$$

Here all $(r_i)_n$ are elements of the *finite* A_n. (Obviously, since A is finite, every A_n is finite.) Say $A_n = \{p_1, \ldots, p_\ell\}$. Then, by associativity and commutativity of $||$, we can write

$$(q_k)_n = ((q_1)_n || p_1^{\underline{f_1(k)}} || p_2^{\underline{f_2(k)}} || \ldots || p_\ell^{\underline{f_\ell(k)}})_n$$

for some *monotone* functions f_i $(i=1,\ldots,\ell)$, with the understanding that if $f_i(k) = 0$, the corresponding 'mergend' vanishes. By Proposition 2.3, every

$$p_i^{\underline{f_i(k)}}$$

stabilizes modulo n, with growing k; whence the result follows. □

2.5. PROPOSITION. *Let* A *be finite. Let* $q_1, q_2, \ldots$ *be a sequence in* A *such that for all* $i \geq 1$, *either*

(i) $q_{i+1} = q_i || r_i$, *or*
(ii) $q_{i+1} = q_i r_i$

for some r_i. *Then the sequence* $q_1, q_2, \ldots$ *stabilizes modulo* n.

PROOF. We may suppose that for infinitely many i we are in case (ii); otherwise we are done at once using Proposition 2.4.

So by Proposition 1.1.8, $\nu(q_i) \geq n$, and hence $\nu((q_i)_n) = n$, for all but finitely many i. (Here we use also the obvious fact: $\nu(q_i r_i) \geq \nu(q_i)$.)

Now if $\nu((q_i)_n) = n$, and $q_{i+1} = q_i r_i$, then evidently $(q_{i+1})_n = (q_i)_n$. That is, modulo n, right concatenation has no effect from some i onwards. But then we are again in the case of the previous proposition. □

2.5.1. REMARK. If in Proposition 2.5, (ii) is replaced by (ii) $q_{i+1} = q_i + r_i$, then the resulting proposition is no longer true. Cfr. Example 1.2.2.

2.5.2. REMARK. A corollary of Proposition 2.5 is that in every A_n as well as in A_∞, if A is finite:

$$\exists x \forall y \quad x||y = x,$$

i.e. there exists an element which is "saturated" w.r.t. merges.

2.6. PROPOSITION. *Let* A *be finite. Let* $q_1, q_2, \ldots$ *be a sequence in* A *such that for all* $i \geq 1$, *either*

(i) $q_{i+1} = q_i \amalg r_i$, *or*
(ii) $q_{i+1} = q_i r_i$

for some r_i. *Then the sequence* $q_1, q_2, \ldots$ *stabilizes modulo* n.

PROOF. By Proposition 1.1.4, we have $q_1 = \Sigma a_i + \Sigma b_j x_j$ for some $a_i, b_j, x_j \in A$. Now if $q_2 = q_1 r_1$, then

$$q_2 = \Sigma a_i r_1 + \Sigma b_j x_j r_1,$$

and if $q_2 = q_1 \amalg r_1$, then

$$q_2 = \Sigma a_i r_1 + \Sigma b_j (x_j || r_1).$$

In both cases q_2 has the form say $\Sigma c_k p_k$ for some $c_k, p_k \in A$.

Now if e.g.

$$\begin{aligned} q_3 &= q_2 \amalg r_2 \\ q_4 &= q_3 r_3 \\ q_5 &= q_4 \amalg r_4 \\ q_6 &= q_5 \amalg r_5 \\ q_7 &= q_6 r_6 \\ &\vdots \end{aligned}$$

so that $q_7 = ((((q_2 \amalg r_2) r_3) \amalg r_4) \amalg r_5) r_6$, then, correspondingly,

$$q_3 = (\Sigma c_k p_k) \mathbin{\lfloor\!\lfloor} r_2 = \Sigma c_k (p_k \| r_2),$$

$$q_4 = (\Sigma c_k (p_k \| r_2)) r_3 = \Sigma c_k (p_k \| r_2) r_3$$

$$\vdots$$

and $q_7 = \Sigma c_k [((((p_k \| r_2) r_3) \| r_4) \| r_5) r_6]$. Hence an appeal to the previous proposition yields the result. □

2.6.1. REMARK. The generality in Propositions 2.5 and 2.6 w.r.t. the elements r_i, suggests looking at possible stabilization (modulo n) of general sequences of the forms:

(i) $q,\ s_1(q),\ s_1(s_2(q)),\ s_1(s_2(s_3(q))),\ \ldots$

(ii) $q,\ s_1(q),\ s_2(s_1(q)),\ s_3(s_2(s_1(q))),\ \ldots$

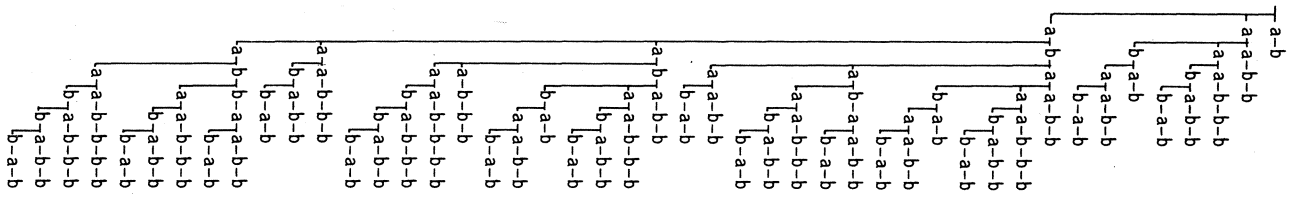
where $q \in \mathcal{A}$ and $s_i(x)$ $(i \geq 1)$ are arbitrary expressions $\in$ EXP having only x free.

Both types of sequences do not necessarily stabilize, however.
For (i): take $s_{2n+1}(x) = xa$, $s_{2n+2} = xb$ $(n \geq 0)$.
For (ii): take $s_{2n+1}(x) = x+a^3$, $s_{2n+2} = x \| b^3$ $(n \geq 0)$.
Then (i) does not stabilize modulo 1 on 0 (in fact, 0 will be introduced only in Section 3) and (ii) does not stabilize modulo 3 on $a^3 \| b^3$ as already remarked in 2.5.1.

We will now state and prove the main theorem of this paper, saying that every sequence $q,\ s(q),\ s^2(q),\ \ldots$ must eventually be constant modulo n. For expressions like e.g. $s(x) = ax+b(c+x)+d$ this is clear since iterating $s(x)$ yields a tree which develops itself in such a way that an increasing part of it is fixed. But even for simple terms as $s(x) = x \| x + ab$ the situation is at first sight not at all clear: in each step of the iteration the whole tree including the top is again in 'motion'. Moreover, the complexity of the expressions $q,\ s(q),\ s^2(q),\ \ldots$ may grow very fast; to get an impression, consider the following example:

2.6.2. EXAMPLE. Let $s(x) = x \| x + ab = x \mathbin{\lfloor\!\lfloor} x + ab$. Then $s(0) = ab$ (0 is in fact only in Section 3 introduced), $s^2(0) = s(ab) = a(bab+abb)+ab$, and

$s^3(0)$ =



2.7. THEOREM. *Let* $q \in A$ *and let* $s(x) \in$ EXP *have only* x *as free variable. Then the iteration sequence* $q, s(q), s(s(q)), \ldots, s^k(q), \ldots$ *stabilizes modulo* n, *for every* $n \geq 1$.

PROOF. The proof is by induction on n. Basis: $n = 1$. By Proposition 1.2.3, $(s(x))_1 = \Sigma a_i$ or $(s(x))_1 = (x)_1 + \Sigma a_i$.

(E.g. if $s(x) = x \amalg x + a \amalg x + bcx$, then

$$(s(x))_1 = (x \amalg x)_1 + (a \amalg x)_1 + (bcx)_1 = (x)_1 + a + b.)$$

In the first case the iteration sequence stabilizes modulo 1 at Σa_i, in the second case at $(q)_1 + \Sigma a_i$.

Induction step. Induction hypothesis: suppose the statement in the theorem is proved for n-1.

Consider s(x). It has the following form, possibly after some rewritings by means of axioms PA5,6:

$$\begin{aligned}
&x*t_1*t_2*\ldots*t_- \ + \\
&x*t_-*t_-*\ldots*t_- \ + \\
&\vdots \\
&x*t_-*\ldots\ldots*t_- \ + \\
&a_1*t_-*\ldots\ldots*t_- \ + \\
&a_1*t_-*\ldots\ldots*t_- \ + \\
&\vdots \\
&a_1*t_-*\ldots\ldots*t_- \ + \\
&\vdots \\
&a_k*t_-*\ldots\ldots*t_- \ + \\
&\vdots \\
&a_k*t_-*\ldots\ldots*t_-
\end{aligned}$$

Here * is either $\perp\!\!\!\perp$ or $\cdot$, $a_1,\ldots,a_k \in A$ and $t_1,t_2,t_\_,t_\_,\ldots \in EXP$. (The reader is invited to write the appropriate subscripts for the _ in $t_\_$.) In each summand brackets associate to the left.

In order to avoid awkward notation, we will give the remainder of the proof using as a typical example

$$s(x) = ((x \perp\!\!\!\perp t_1)t_2) \perp\!\!\!\perp t_3 + (x \perp\!\!\!\perp t_4) \parallel t_5 + a \perp\!\!\!\perp t_6.$$

Note that $t_1,\ldots,t_6$ may contain occurrences of x. To denote this, we will write $t_1(x),\ldots,t_6(x)$.

Now from Proposition 1.2.3 we have (using also the following fact which is easily derived from that Proposition: $(t(x))_n = (t((x)_n))_n$, $t \in EXP$):

$$\begin{aligned}(s(x))_n = {} & ((x_n \perp\!\!\!\perp (t_1(x_{n-1}))_{n-1})(t_2(x_{n-1}))_{n-1}) \perp\!\!\!\perp (t_3(x_{n-1}))_{n-1} + \\ & (x_n \perp\!\!\!\perp (t_4(x_{n-1}))_{n-1}) \perp\!\!\!\perp (t_5(x_{n-1}))_{n-1} + \\ & a \perp\!\!\!\perp (t_6(x_{n-1}))_{n-1}.\end{aligned}$$

By the induction hypothesis, the iteration sequence stabilizes modulo n-1, say at $Q \in A_{n-1}$. Hence for k sufficiently large we have, substituting $s^k(q)$ for x and Q for $(x)_{n-1}$:

$$\begin{aligned}(s(s^k(q)))_n = {} & (((s^k(q))_n \perp\!\!\!\perp (t_1(Q))_{n-1})(t_2(Q))_{n-1}) \perp\!\!\!\perp (t_3(Q))_{n-1} + \\ & ((s^k(q))_n \perp\!\!\!\perp (t_4(Q))_{n-1}) \perp\!\!\!\perp (t_5(Q))_{n-1} + \\ & a \perp\!\!\!\perp (t_6(Q))_{n-1}.\end{aligned}$$

Let us write t_i' instead of $t_i(Q)$, $i=1,\ldots,6$.

So in order to prove stabilization modulo n of the iteration sequence generated by s(x) with starting value q, it suffices to prove stabilization modulo n of the iteration sequence generated by

$$s'(x) = ((x \perp\!\!\!\perp t_1')t_2') \parallel t_3' + (x \perp\!\!\!\perp t_4') \parallel t_5' + a \perp\!\!\!\perp t_6',$$

with starting value $s^k(q) \equiv P$ for some k. The advantage obtained now is that the t_i' are closed terms, i.e. not containing x anymore.

Write

$$T_1(x) \equiv ((x \perp\!\!\!\perp t_1')t_2') \perp\!\!\!\perp t_3',$$

$$T_2(x) \equiv (x \perp\!\!\!\perp t_4') \perp\!\!\!\perp t_5',$$

$$T_3 \equiv a \perp\!\!\!\perp t_6'.$$

Then $s'(P) = T_1(P) + T_2(P) + T_3$, and

$$s'(s'(P)) = T_1(T_1(P)+T_2(P)+T_3) + T_2(T_1(P)+T_2(P)+T_3) + T_3$$
$$= T_1(T_1(P)) + T_1(T_2(P)) + T_1(T_3) +$$
$$T_2(T_1(P)) + T_2(T_2(P)) + T_2(T_3) + T_3.$$

Here the 'linearity' of T_1 and T_2 is due to the distributive laws for $\amalg$ and $\cdot$ (PA5,6). Continuing in this way we find

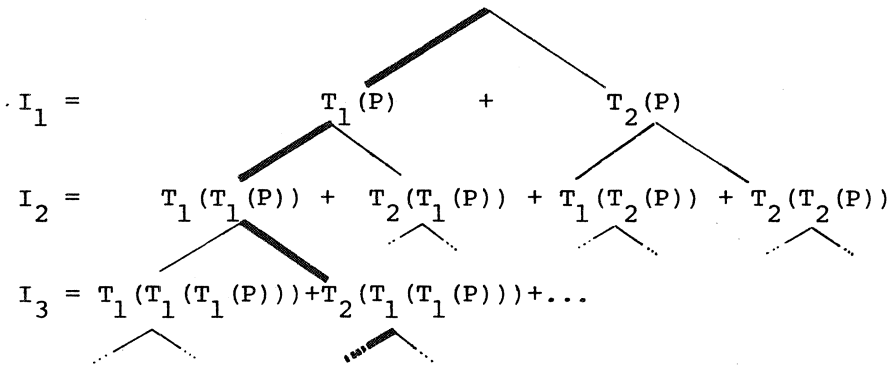
$$s'^k(P) = I_k + II_k + T_3$$

where

$$I_k = \sum_{i_1,..,i_k \in \{1,2\}} T_{i_1}(T_{i_2}(...(T_{i_k}(P))...))$$

$$II_k = \sum_{j_1,..,j_{k-1} \in \{1,2\}} T_{j_1}(T_{j_2}(...(T_{j_{k-1}}(T_3))...)).$$

Now both summands I_k and II_k stabilize modulo n for growing k. For, consider I_k:



Each "branch" in the tree thus obtained, e.g. the indicated branch

$$T_1(P),\ T_1(T_1(P)),\ T_2(T_1(T_1(P))),\ T_1(T_2(T_1(T_1(P)))),\ ...$$

stabilizes modulo n, according to Proposition 2.6, since the operations T_1, T_2 consist of some left merges on the right and some concatenations on the right.

Hence, by König's Lemma, there is some k such that all branches are stabilized (modulo n) at that level k, i.e. for all summands $T_{i_1}(T_{i_2}(...(T_{i_k}(P))...))$ in I_k further prefixing of T_1 or T_2 makes no

difference modulo n. So from that k onwards, I_k is stable, modulo n.

(Note that the finiteness condition on A, necessary for the application of Proposition 2.6, is satisfied since the only $a \in A$ playing a role here, occur in q and s(x).) □

2.8. REMARK. A fortiori we have stabilization of iteration sequences when in addition to PA1,...,8 left distributivity z(x+y) = zx+zy is assumed. This amounts to working with the set of branches ('traces') of the trees of $p \in \mathcal{A}$. (This result could however much faster be obtained than via the method above.)

2.9. COROLLARY. Let $s(x) \in EXP$ contain no other variables than x. Then the equation x = s(x) has a solution in $\mathcal{A}_n$, for every $n \geq 1$; and likewise in $\mathcal{A}_\infty$.

(Equivalently: every definable function has a fixed point in each $\mathcal{A}_n$ and $\mathcal{A}_\infty$.)

2.9.1. REMARK. Of course, in general x = s(x) will have more than one solution; consider e.g. s(x)= xa. (Or even s(x) = x.)

3. FIXED POINT SEMANTICS

3.1. For a closer correspondence between process algebras and the processes of DE BAKKER & ZUCKER [1,2] we will slightly change the definition of a process algebra, by adding a new distinguished element, 0. The axioms of a *process algebra with* 0 are:

$$x+y = y+x$$
$$x+(y+z) = (x+y)+z$$
$$x+x = x$$
$$x+0 = x$$
$$(xy)z = x(yz)$$
$$x0 = 0x = x$$
$$(x+y)z = xz+yz \text{ if } x,y \neq 0$$
$$(x+y) \between z = x \between z + y \between z$$
$$ax \between y = a(x \between y + y \between x)$$
$$0 \between y = 0$$
$$y \between 0 = y.$$

The proviso 'x,y ≠ 0' is necessary since otherwise

$$xy = (x+0)y = xy+0y = xy+y.$$

Axiom PA8: $a \between y = ay$ can now be derived:

$$a \between y = a0 \between y = a(0 \between y + y \between 0) = a(0+y) = ay.$$

The definitions of $\mathcal{A}$, $\mathcal{A}_n$, $\mathcal{A}_\infty$ and EXP are easily adapted to the case in which 0 is present. To indicate the presence of 0, we will write $\mathcal{A}'$, $\mathcal{A}_n'$, $\mathcal{A}_\infty'$, EXP'. The propositions and definitions leading up to Theorem 2.7 carry over with minor adaptations left to the reader. (E.g. include $(0)_n = 0$.) Let us look in some detail to the generalization of Theorem 2.7.

3.2. THEOREM. *Let* $q \in \mathcal{A}'$ *and let* $s(x) \in$ EXP' *contain no other variables than* x. *Then the iteration sequence* q, s(q), s(s(q)), ... *stabilizes modulo* n, *for every* $n \geq 1$.

PROOF. Distinguish the cases

1. s(x) is constant
2. s(x) is not constant
   - 2.1. q = 0
     - 2.1.1. 0 = s(0)
     - 2.1.2. 0 ≠ s(0)
   - 2.2. q ≠ 0

Case 1 is trivial. Case 2.1.1 is also trivial. In case 2.1.2 we take s(0) as new starting value and then we are in case 2.2. Now rewrite s(x) such that the term s(x) contains no occurrence of 0; for a non-constant s(x) this is clearly possible. But then iteration of such a 0-free s(x) on a starting value ≠ 0 takes place entirely in $\mathcal{A}$, the process algebra without 0 for which Theorem 2.7 holds. Hence stabilization follows. □

We will now extend the set EXP' of expressions in the language corresponding to a process algebra with 0, to the set EXP_μ' of 'μ-expressions':

3.3. DEFINITION. The set EXP_μ' of μ-expressions is defined by

$$s ::= a,0 \mid x,y,\ldots \mid s_1+s_2 \mid s_1 s_2 \mid s_1 \between s_2 \mid \mu x.s$$

We will now define the semantics $[\![s]\!]_n$ and $[\![s]\!]$ of $s \in EXP'_\mu$, resp. in $\mathcal{A}'_n$ and $\mathcal{A}'_\infty$. We will give an informal discussion first.

It will be convenient to introduce the following "*stabilization functions*", whose existence is guaranteed by Theorem 3.2:

3.4. NOTATION. $EXP'(\vec{x}) \subseteq EXP'$ is the set of μ-free expressions containing no other variables than $\vec{x}$.

3.5. DEFINITION. (i) σ: $EXP'(x) \times \mathbb{N} \to \mathbb{N}$ is the function denoting the number of steps it takes the iteration sequence $0, s(0), \ldots$ to stabilize modulo n. I.e.:

$$\sigma(s(x),n) = \min \{k \mid (s^k(0))_n = (s^{k+1}(0))_n\}.$$

(ii) Let A be finite (so $\mathcal{A}'_n$ is finite). Then σ^*: $EXP'(x,\vec{y}) \times \mathbb{N} \to \mathbb{N}$ is defined by

$$\sigma^*(s(x,\vec{y}),n) = \max \{\sigma(s(x,\vec{p}),n) \mid \vec{p} \in \mathcal{A}'_n\}.$$

So σ^* gives an uniform bound for the number of steps it takes an instantiation $s(x,\vec{p})$ of $s(x,\vec{y})$ to stabilize modulo n.

We proceed with the discussion of $[\![s]\!]_n$ and $[\![s]\!]$. For μ-free s the definition of $[\![s]\!]_n$ is clear; and $[\![s]\!]$ will be the projective sequence $([\![s]\!]_1, [\![s]\!]_2, \ldots)$.

The case of $\mu x.s(x)$ where $s \in EXP'(x)$, is also clear by virtue of Theorem 3.2:

$$[\![\mu x.s(x)]\!]_n = (s^{\sigma(s(x),n)}(0))_n.$$

It is not hard to check that with this definition $[\![\mu x.s(x)]\!]$ is indeed a projective sequence.

Next, consider $\mu y.t(\mu x.s(x,y))$ where t and s are μ-free. Intuitively, $[\![\mu y.t(\mu x.s(x,y))]\!]_n$ will be computed as follows:

$$p_1 = t(\mu x.s(x,0)) = t(s^{\sigma(s(x,0),n)}(0,0))$$

$$p_2 = t(\mu x.s(x,p_1)) = t(s^{\sigma(s(x,p_1),n)}(0,p_1))$$

$$p_3 = t(\mu x.s(x,p_2)) = t(s^{\sigma(s(x,p_2),n)}(0,p_2))$$

$$\vdots$$

The problem is whether p_1, p_2, p_3, ... stabilizes modulo n. Indeed this is the case, as we will show using σ^*. (The finiteness condition on A is fullfilled: only the $a \in A$ occurring in $\mu y.t(\mu x.s(x,y))$ play a role.)

Let $M = \sigma^*(s(x,y),n)$. Then by Definition 3.5, for all $i \geq 1$:

$$\sigma(s(x,p_i),n) \leq \sigma^*(s(x,y),n) = M.$$

Hence modulo n:

$$p_1 = t(s^M(0,0))$$
$$p_2 = t(s^M(0,p_1))$$
$$p_3 = t(s^M(0,p_2))$$
$$\vdots$$

and this is an *iteration* sequence, with generator $h(y) = t(s^M(0,y))$. Hence p_1, p_2, ... stabilizes modulo n. The result is by definition $[\![\mu y.t(\mu x.s(x,y))]\!]_n$.

3.6. REMARK. Note that μ is a *selection* operator on the solution set of the equation $x = s(x)$ rather than a *minimal* fixed point operator, since we have no partial ordering. In fact:

3.6.1. PROPOSITION. *There does not exist a suitable p.o.* $\leq$ *on* A', *the set of finite processes with* 0.

(Here $\leq$ *is called suitable iff:*

(i) $0 \leq p$

(ii) $p \leq q \implies s(p) \leq s(q)$

for all $p,q \in A'$ *and expressions* $s(x) \in EXP'(x)$.)

PROOF. Let $a \in A$ and suppose a suitable $\leq$ exists. Then

$$aa = aa+0 \leq aa+a = aa+a0 \leq aa+aa = aa.$$

Hence, since $\leq$ is a p.o., $aa = aa+a$; a contradiction. □

3.6.2. REMARK. Note that Proposition 3.6.1 also holds if the set of axioms for A' is extended with left distributivity: $z(x+y) = zx+zy$ if $x,y \neq 0$. So, even when dealing with the set of branches of the trees of elements from A', a suitable p.o. does not exist.

4. CONCLUDING REMARKS

It is not hard to formulate the correspondence between the process algebra $\mathcal{A}_\infty$ introduced above as a projective limit of the sets $\mathcal{A}_n$ of finite processes, and a solution P of the domain equation for uniform processes (see DE BAKKER & ZUCKER [1,2]):

$$P \cong \{p_0\} \cup \mathcal{P}_c(A \times P) \qquad (*)$$

Here A is as above, p_0 corresponds to 0 in Section 3, and $\mathcal{P}_c(A \times P)$ denotes the collection of all closed subsets of $A \times P$ (closed w.r.t. the Hausdorff metric: see DE BAKKER & ZUCKER [1,2]) and $\cong$ denotes isometric equivalence.

The correspondence between $\mathcal{A}_\infty$ and P is as follows: $\mathcal{A}_\infty$ is (up to an isometry) the set of those processes which are the limits of finitely deep and 'finitely branching' processes. Here $\mathcal{A}_\infty$ can be enriched with the following metric δ:

$$\delta(p,q) = 2^{-\min\{n \mid p_n \neq q_n\}}$$

for $p = (p_0, p_1, \ldots)$ and $q = (q_0, q_1, \ldots)$.

Note that pair formation in P has vanished in $\mathcal{A}_\infty$: e.g. $\{\langle a,q\rangle\}$ in P corresponds in $\mathcal{A}_\infty$ with aq (or better, with $a\tilde{q}$ if q corresponds with $\tilde{q} \in \mathcal{A}_\infty$).

It is routine to establish the isometry between $\mathcal{A}_\infty$ and the above mentioned subspace of P; we will leave this to the reader. It will also be clear that the main theorem (3.2), stating that iteration sequences $\{s^k(q)\}_k$ stabilize modulo n for all $n \geq 1$, can be rephrased as stating that the sequence $\{s^k(q)\}_k$ is a Cauchy sequence in the sense of the metric δ; i.e. $\lim_{k\to\infty} s^k(q)$ exists. Moreover, by our discussion in Section 3, this holds even when s(x) contains μ-expressions ($s(x) \in EXP'_\mu$).

The domain equation (*) is only one of a number of domain equations studied in the framework of metric spaces in DE BAKKER & ZUCKER [1,2], where also non-uniform processes are considered as well as value-passing and synchronization mechanisms. (In fact, the topological treatment of DE BAKKER & ZUCKER [1,2] aims at dealing in a uniform way with a whole class of such domain equations.) It will be interesting to study versions of process algebras corresponding to such extensions. For one of these extensions, with a 'next'-operator as in data flow, we remark that

Theorem 3.2 does not generalize. Let 'next' satisfy

$$next(x+y) = next(x) + next(y)$$

$$next(ax) = x$$

$$next(a) = 0$$

$$next(0) = 0.$$

Now $s(x) = next(x) + xa$ generates the iteration sequence on 0:

$$a,\ a^2,\ a+a^3,\ a^2+a^4,\ a+a^3+a^5,\ a^2+a^4+a^6,\ a+a^3+a^5+a^7,\ldots$$

or, modulo 2:

$$a,\ a^2,\ a+a^2,\ a^2,\ a+a^2,\ldots\ .$$

This counterexample to Theorem 3.2 for a process algebra containing 'next' holds also in the presence of left distributivity "$z(x+y) = zx+zy$ if $x,y\neq 0$".

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,
To appear in the Proc. of the 4th Advanced Course on Foundations of Computer Science, Amsterdam, June 1982.

[3] DENNIS, J., J. FOSSEEN & J. LINDERMAN,
Data flow schemes,
Springer LNCS 5, 1974.

[4] HOARE, C.A.R.,
Communicating Sequential Processes,
Comm.ACM, 21 (1978), 666-677.

[5] KAHN, G. & D. MACQUEEN,
Coroutines and networks of parallel processes,
Proc. IFIP Congress, pp.993-998, North-Holland, 1977.

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

[7] NIVAT, M.,
Synchronization of concurrent processes,
Formal Language Theory (R.V. Book, ed.), 429-454, Academic Press, 1980.

[8] PETRI, C.,
Introduction to General Net Theory,
Advanced Course on General Net Theory of Processes and Systems,
Springer LNCS, 1980.

[9] PARK, D.,
Nondeterministic networks: notes on some anomalies,
To appear in the Proc. of the 4th Advanced Course on Foundations of Computer Science, Amsterdam, June 1982.

[10] PRATT, V.R.,
Process Logic,
Proc. 6th Ann. ACM Symp. on Principles of Programming Languages, 93-100, San Antonio, Texas, Jan.1979.

[11] PRATT, V.R.,
On the composition of processes,
Proc. 9th ACM Symp. on Principles of Programming Languages, 213-223, 1982.

MC NR

35228

ONTVANGEN 8 SEP. 1982

09F11
09F12
09F43