THE ALGEBRA OF RECURSIVELY DEFINED PROCESSES AND THE ALGEBRA OF REGULAR PROCESSES

J.A. Bergstra and J.W. Klop

Centrum voor Wiskunde en Informatica, Kruislaan 413, AMSTERDAM

ABSTRACT. We introduce recursively defined processes and regular processes, both in presence and absence of communication. It is shown that both classes are process algebras. As an example of recursively defined processes, Bag and Stack are discussed in detail. It is shown that Bag cannot be recursively defined without merge. We introduce fixed point algebras which have useful applications in several proofs.

INTRODUCTION. ACP, Algebra of Communicating Processes, was introduced in Bergstra & Klop [3]. It combines a purely algebraic formulation of a part of Milner's CCS [9] with an algebraic presentation of the denotational semantics of processes as given by de Bakker & Zucker [1,2]; moreover it includes two laws on communication of atomic actions which are also present in Hennessy [6]. The ingredients of ACP are the following:

- a finite set A of so-called atomic actions a,b,c,... including a constant δ for deadlock (or failure). With $\underline{A}$ we denote $A - \{\delta\}$, the proper actions.
- a mapping $.|. : A \times A \to A$, called the communication function. If $a|b = c$ then c is the action that results from simultaneously executing a and b. Processes will cooperate by sharing actions rather than sharing data.
- a subset H of A (usually H contains the actions which must communicate with other actions in order to be executable). The elements of H are called subatomic actions.
- a signature of operations $\cdot, +, ||, \lfloor\!\lfloor, |, \delta, \partial_H$. (For $x \cdot y$ we will often write xy.)

The axioms of ACP are displayed in Table 1 on the next page.

These axioms reflect in an algebraic way that + represents choice, $\cdot$ represents sequential composition and $||$ the merge operator.

The operations $\lfloor\!\lfloor$ (left merge) and $|$ (communication merge) are auxiliary ones. Our primary interest remains for $+, \cdot, ||$. The process $x \lfloor\!\lfloor y$ is like $x||y$, but takes its first step from x, and $x|y$ is like $x||y$ but requires the first action to be a communication (between a first step of x and a first step of y).

1. PRELIMINARIES

1.1. Models of ACP. The axioms of ACP allow for a large variety of models ('process algebras'). In [3,5] we investigated the 'standard' model A^∞ for ACP which is used throughout this paper.

We will quickly describe the construction of the standard model A^∞. First one constructs A_ω, the initial model of ACP seen as an equational specification over the signature with a constant for each atom. The process algebra A_ω contains only finite processes and hence cannot solve fixed point (or recursion) equations, such as $X = aX + b$. One way of completing A_ω is as follows.

Let A_ω mod n (for short, A_n) be for $n \geqslant 1$, the homomorphic image of A_ω obtained by identifying two processes p,q in A_ω if their 'trees' coincide up to depth n. (More precisely, if their projections $(p)_n, (q)_n$ coincide. Here $(a)_n = a$, $(ax)_{n+1} = a(x)_n$, $(ax)_1 = a$, $(x+y)_n = (x)_n + (y)_n$.) The A_n are also process algebras with operations $+^n$ etc. defined as $(x +^n y) = (x+y)_n$ etc.

$x + y = y + x$	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
$a\|b = b\|a$	C1
$(a\|b)\|c = a\|(b\|c)$	C2
$\delta\|a = \delta$	C3
$x\|\|y = x \mathbin{\|\!\!\_} y + y \mathbin{\|\!\!\_} x + x\|y$	CM1
$a \mathbin{\|\!\!\_} x = a.x$	CM2
$(ax) \mathbin{\|\!\!\_} y = a(x\|\|y)$	CM3
$(x + y) \mathbin{\|\!\!\_} z = x \mathbin{\|\!\!\_} z + y \mathbin{\|\!\!\_} z$	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
$\partial_H(a) = a$ 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
$\partial_H(x.y) = \partial_H(x).\partial_H(y)$	D4

Table 1.

Now A^∞ is defined as the projective limit of the finite process algebras A_n, $n \geq 1$. That means that the elements of A^∞ are the projective sequences $(p_1, p_2, \ldots, p_n, \ldots)$ where $p_n \in A_n$ and such that $(p_{n+1})_n = p_n$; the operations are defined coordinate-wise.

All process algebras introduced in this paper will be subalgebras of A^∞.

Another way of completing the algebra A_ω of finite processes is as in De Bakker & Zucker [1,2] as a metrical completion. Furthermore one obtains a large collection of process algebras starting from process graphs (as in the sequel) and dividing out some notion of bisimulation. Such 'graph models' will not be considered in this paper; see [5].

1.2. Restricted signatures. It is useful to consider a smaller set of operations on processes, for instance: only + and $\cdot$. Then one may forget δ and consider structures

$$\underline{A}_\omega(+,\cdot),\ \underline{A}_n(+,\cdot),\ \underline{A}^\infty(+,\cdot)$$

where $\underline{A} = A - \{\delta\}$. Furthermore, under the assumption that $a|b = \delta$ for all $a,b \in A$, we may add $\|$ and $\mathbin{\|\!\!\_}$ to the signature of these algebras, thus obtaining

$$\underline{A}_\omega(+,\cdot,\|,\mathbin{\|\!\!\_}),\ \underline{A}_n(+,\cdot,\|,\mathbin{\|\!\!\_}) \text{ and } \underline{A}^\infty(+,\cdot,\|,\mathbin{\|\!\!\_}).$$

Of course these structures can be constructed immediately without any reference to communication. Let PA be the following axiom system (see Table 2). Then $\underline{A}_\omega(+,\cdot,\|,\mathbin{\|\!\!\_})$ is just the initial algebra of PA.

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

Table 2.

1.3. Linear terms and guarded terms. Let $X_1,\ldots,X_n$ be variables ranging over processes. Given a (restricted) signature of operators from $+, \cdot, \|, ⌊⌊, |, \partial_H, \delta$ two kinds of terms containing variables $X_1,\ldots,X_n$ are of particular importance:

(i) Linear terms. These are inductively defined as follows:

- atoms a, δ and variables X_i are linear terms,
- if T_1 and T_2 are linear terms then so are $T_1 + T_2$ and aT_1 (for $a \in A$).

An equation $T_1 = T_2$ is called linear if T_1, T_2 are linear.

(ii) Guarded terms. The unguarded terms are inductively defined as follows:

- X_i is unguarded,
- if T is unguarded then so are $T + T'$, $T \cdot T'$, $\partial_H(T)$, $T \| T'$, $T ⌊⌊ T'$, $T | T'$ (for every T').

A term T is guarded if it is not unguarded.

1.4. Process graphs. A process graph g for an action alphabet A is a rooted directed multigraph with edges labeled by elements of A. Here g may be infinite and may contain cycles. Process graphs (or transition diagrams) constitute a very useful tool for the description of processes. In this section we will consider finite process graphs, possibly containing cycles.

Let g be a finite process graph over A. We show how to find a semantics of g in A^∞. To each node s of g with a positive outdegree, attach a process name X_s. Then the following system of guarded linear equations arises:

$$X_s = \sum_{(a,t) \in U} a \cdot X_t + \sum_{a \in V} a \qquad (E_X)$$

where $U = \{(a,t) \mid g: s \xrightarrow{a} t \ \&\ t \text{ has positive outdegree}\}$, $V = \{a \mid \exists t\ g: s \xrightarrow{a} t \ \&\ t$ has outdegree $0\}$. This system E_X has a unique solution in A^∞ and with s_0 the root of g, we define: $[\![g]\!] = p_{s_0}$, where $\langle p_s \rangle$ solves E_X.

1.5. Operations on process graphs. We assume that $.|.$ is defined as a communication function: $A \times A \to A$. Now let g,h be two process graphs for A. We define new process graphs as follows:

- $g + h$ results by glueing together the roots of g and h, provided these roots are acyclic, i.e. not lying on a cycle. (Otherwise g,h must be unwinded to make the roots acyclic; for a more precise account see [5].)
- $g \cdot h$ results by glueing together the root of h and all endpoints of g,
- $\partial_H(g)$ results by replacing all labels $a \in H$ by δ in g,
- $g \| h$ is the cartesian product of the node sets $\{s, s', ..\}$ and $\{t, t', ..\}$ of g resp. h provided with labeled edges as follows:
  - (i) $(s,t) \xrightarrow{a} (s',t)$ if in g we have $s \xrightarrow{a} s'$
  - (ii) $(s,t) \xrightarrow{a} (s,t')$ if in h we have $t \xrightarrow{a} t'$
  - (iii) $(s,t) \xrightarrow{a} (s',t')$ if for some $b,c \in A$ we have $b|c = a$ and $s \xrightarrow{b} s'$ in g, $t \xrightarrow{c} t'$ in h.
- $g \mathbin{|\!\!\lfloor} h$ is defined like $g \| h$, but leaving out all transitions of types (ii) and (iii) if s is the root of g,
- $g | h$ is defined like $g \| h$ but leaving out all transitions of types (i) and (ii) if s resp. t is the root of g resp. h.

Of course we have $[\![g + h]\!] = [\![g]\!] + [\![h]\!]$ etc. More precisely: $[\![\ ]\!]$ as in 1.4 is a homomorphism from the collection of finite process graphs (with acyclic roots) with operations as just described, to the process algebra A^∞.

2. REGULAR PROCESSES

2.1. The algebra of regular processes. For $p \in A^\infty$ the collection Sub(p) of subprocesses of p is defined by:

$p \in Sub(p)$

$ax \in Sub(p) \implies x \in Sub(p)$, provided $a \neq \delta$

$ax + y \in Sub(p) \implies x \in Sub(p)$, provided $a \neq \delta$.

We define $p \in A^\infty$ to be regular if Sub(p) is finite, and denote with $r(A^\infty)$ the collection of regular processes in A^∞. Now, noting that the operations in 1.5 on process graphs preserve finiteness, we have immediately the following facts:

THEOREM 2.1.1. (i) *If* p *is regular then there is a finite process graph* g *with* $[\![g]\!] = p$, *and conversely.*

(ii) *The class of regular processes is closed under the operations* $+, \cdot, \|, \mathbin{|\!\!\lfloor}, |, \partial_H$. *Hence* $r(A^\infty)$ *is a subalgebra of* A^∞.

(iii) $r(A^\infty)$ *contains exactly the solutions of finite systems of guarded linear equations.* □

2.2. CSP program algebras. In this subsection we illustrate the use of the algebras $r(A^\infty)$ by giving an interpretation of simplified CSP programs in such algebras.

Let Σ be an algebraic signature and let X be a set of variables. A CSP component

<u>program</u> S is defined by:

$$S ::= b \mid b \,\&\, x{:=}t \mid b \,\&\, C!t \mid b \,\&\, C?x \mid S_1;S_2 \mid S_1 \square S_2 \mid \underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}}.$$

Here b is a boolean (quantifier free) expression. The action b is a guard, which can only be passed when it evaluates to <u>true</u>; $b \,\&\, p$ can only be executed if b is true. It is usual to abbreviate <u>true</u> $\&\, p$ to p. All variables x must occur in X. Further, C is an element of a set of channel names.

A <u>CSP program</u> P is a construct of the form $[S_1 \| \ldots \| S_k]$ with the S_i CSP-component programs.

<u>Remark</u>. Originally the CSP syntax indicates restrictions: the S_i must work with different variables, the channels are used to interconnect specific pairs of components. (See Hoare [7,8].) However, from our point of view these restrictions are just guidelines on how to obtain a properly modularised system (semantically their meaning is not so clear).

Let a CSP program $P = [S_1 \| \ldots \| S_n]$ be given. We will evaluate an <u>intermediate semantics</u> for it by embedding it in a process algebra. First we fix a set of atomic actions; these are:

(i)	$b_1, \neg b_1, b_1 \wedge b_2$	if b_1, b_2 occur in P,
(ii)	$b \,\&\, x{:=}t$	if x and t occur in P, for all b from (i)
(iii)	$b \,\&\, C!t$	if C!t occurs in P, for all b from (i)
(iv)	$b \,\&\, C?x$	if C?x occurs in P, for all b from (i).

Let us call this alphabet of actions A_{CSP-P}. If we delete all actions of the form $b \,\&\, C!t$ or $b \,\&\, C?x$ we obtain A_P. So A_P contains the proper actions that evaluation of P can involve, while A_{CSP-P} contains the subatomic actions as well. H contains the actions of the form $b \,\&\, C!t$ and $b \,\&\, C?x$.

Next we fix a communication function. All communications lead to δ, except the following ones: $b_1 \,\&\, C!t \mid b_2 \,\&\, C?x \;=\; (b_1 \wedge b_2) \,\&\, x{:=}t$.
We will first find an image $[\![P]\!]$ of P in A^{∞}_{CSP-P}. This is done using the notation of μ-calculus. We use an inductive definition for subprograms of the component programs first:

$$[\![b]\!] = b$$

$$[\![b \,\&\, x{:=}t]\!] = b \,\&\, x{:=}t$$

$$[\![b \,\&\, C!t]\!] = b \,\&\, C!t$$

$$[\![b \,\&\, C?x]\!] = b \,\&\, C?x$$

$$[\![S_1;S_2]\!] = [\![S_1]\!] \cdot [\![S_2]\!]$$

$$[\![S_1 \square S_2]\!] = [\![S_1]\!] + [\![S_2]\!]$$

$$[\![\underline{\text{while}}\ b\ \underline{\text{do}}\ S\ \underline{\text{od}}]\!] = \mu x(b \cdot [\![S]\!] \cdot x + \neg b).$$

Here $\mu x(b \cdot [\![S]\!] \cdot x + \neg b)$ is the unique solution of the equation $X = b \cdot [\![S]\!] \cdot X + \neg b$. It is easily seen that the solution $\underline{X}$ is regular whenever $[\![S]\!]$ is regular.

Inductively one finds that $[\![S]\!]$ is regular for each component program S. Finally for the program P we obtain: $[\![P]\!] = [\![\,[S_1\|\ldots\|S_n]\,]\!] = \partial_H([\![S_1]\!]\|\ldots\|[\![S_n]\!])$.
We can now draw two interesting conclusions:

(i) $[\![P]\!]$ is regular;

(ii) $[\![P]\!]$ can just as well be (recursively) defined in $\underline{A}_P^\infty(+,\cdot)$ (so without any mention of communication).

Proof. (i) $[\![S_i]\!]$ is regular because it is defined using linear recursion equations only. Consequently the $[\![S_i]\!]$ are in $r(A^\infty_{CSP-P})$ and so is $[\![P]\!]$ because $r(A^\infty_{CSP-P})$ is a subalgebra of A^∞_{CSP-P}.
(ii) follows from (i) and Theorem 2.1.1(iii).

Remark. In general one must expect that a recursive definition of $[\![P]\!]$ not involving merge will be substantially more complex than the given one with merge.

3. RECURSIVELY DEFINED PROCESSES

3.1. The algebra of recursively defined processes. Let $X = \{X_1,\ldots,X_n\}$ be a set of process names (variables). We will consider terms over X composed from atoms $a \in A$ and the operators $+, \cdot, \|, \lfloor\!\lfloor, |, \partial_H$. A system E_X of guarded fixed point equations for X is a set of n equations $X_i = T_i(X_1,\ldots,X_n)$, $i = 1,\ldots,n$, with $T_i(X_1,\ldots,X_n)$ a guarded term.

THEOREM 3.1.1. *Each system E_X of guarded fixed point equations has a unique solution in $(A^\infty)^n$.*

PROOF. See De Bakker & Zucker [1,2]; essentially E_X is seen as an operator $(A^\infty)^n \to (A^\infty)^n$ which under suitable metrics is a contraction and has exactly one fixed point, by Banach's fixed point theorem. □

Definition. $p \in A^\infty$ is called recursively definable if there exists a system E_X of guarded fixed point equations over X with solution $(p, q_1,\ldots,q_{n-1})$. With $R(A^\infty)$ (not to be confused with $r(A^\infty)$) we denote the subalgebra of recursively defined processes. This is indeed a process algebra:

PROPOSITION 3.1.2. *The recursively defined processes constitute a subalgebra of A^∞.*

PROOF. Let $E_X = \{X_i = T_i(X) \mid i = 1,\ldots,n\}$ and $E_Y = \{Y_j = S_j(Y) \mid j = 1,\ldots,m\}$. Let $E_Z = E_X \cup E_Y \cup \{Z = T_1(X) \| S_1(Y)\}$. Now if E_X defines p and E_Y defines q, then E_Z defines $p\|q$. Likewise for the other operations. □

Remark. For algebras with restricted signatures the above construction of a subalgebra of recursively defined processes is equally valid. Of course, the equations will then use the restricted signatures only. This leads to algebras like

$$R(\underline{A}^\infty(+,\cdot)) \text{ and } R(\underline{A}^\infty(+,\cdot,\|,\lfloor\!\lfloor)).$$

3.2. Recursive definitions and finitely generated process algebras. Let $p_1,\dots,p_n$ be processes in A^∞. Then $A_\omega(p_1,\dots,p_n)$ will denote the subalgebra of A^∞ generated by $p_1,\dots,p_n$.

Let $X_1,\dots,X_n$ be a set of new names for processes, and let $\underline{X}_1,\dots,\underline{X}_n$ be processes in A^∞. Then with $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ we denote an algebra as above but with the names $X_1,\dots,X_n$ added to the signature.

We define $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ to be a fixed point algebra if the $\underline{X}_i$ are the solutions in A^∞ of some system E_X of guarded fixed point equations where $X = \{X_1,\dots,X_n\}$.

Remark. Let us denote with $A_\omega[X_1,\dots,X_n]$ the free ACP algebra generated over new names $X_1,\dots,X_n$. For each set of interpretations $\underline{X}_1,\dots,\underline{X}_n$ there is a homomorphism ϕ: $A_\omega[X_1,\dots,X_n] \to A_\omega(\underline{X}_1,\dots,\underline{X}_n)$. Now suppose that E_X is a system of guarded fixed point equations for $X = \{X_1,\dots,X_n\}$. Then

$$A_\omega[X_1,\dots,X_n]/E_X$$

is the algebra obtained by dividing out the congruence generated by E_X. On the other hand, let $\underline{X}_1,\dots,\underline{X}_n$ be the unique solutions of E_X in A^∞. There is again a homomorphism

$$\phi\colon A_\omega[X_1,\dots,X_n]/E_X \to A_\omega(\underline{X}_1,\dots,\underline{X}_n).$$

Both algebras $A_\omega[X_1,\dots,X_n]/E_X$ and $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ may be vastly different however. Being an initial algebra of a finite specification, $A_\omega[X_1,\dots,X_n]/E_X$ is semicomputable. It can easily be proved that $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ is in general cosemicomputable. One can also give an example (see [4]) where $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ is not computable (has an undecidable word problem).

THEOREM 3.2.1. *Let* $\underline{X}_1,\dots,\underline{X}_n$ *be solutions of the system of guarded fixed point equations* E_X. *Then the fixed point algebra* $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ *is closed under taking subprocesses.*

PROOF. Let $p \in A_\omega(\underline{X}_1,\dots,\underline{X}_n)$. Then for some term T we have $p = T(\underline{X}_1,\dots,\underline{X}_n)$; after substitutions corresponding to $X_i = T_i(X_1,\dots,X_n)$ we may assume that T is guarded. Using the axioms of ACP one can rewrite $T(X_1,\dots,X_n)$ into the form $\Sigma a_i \cdot R_i(X_1,\dots,X_n) + \Sigma b_i$. Consequently all immediate subprocesses of p, i.e. the $R_i(\underline{X}_1,\dots,\underline{X}_n)$, are in $A_\omega(\underline{X}_1,\dots,\underline{X}_n)$ as well. □

This theorem gives a useful criterion for recursive definability (to be used in Section 4):

COROLLARY 3.2.2. (i) *Let* $p \in R(\underline{A}^\infty(+,\cdot,\|,\lfloor\!\lfloor))$. *Then* Sub(p) *is finitely generated using* $+,\cdot,\|,\lfloor\!\lfloor, a \in A$.

(ii) *Likewise for the restricted signature of* $+,\cdot, a \in A$. □

3.3. Finitely branching processes.

Definition. Let $p \in A^\infty$. (i) Then G_p is the canonical process graph of p, defined as follows. The set of nodes of G_p is $\mathrm{Sub}(p) \cup \{o\}$. Here o is a termination node. The root of G_p is p. The (labeled and directed) edges of G_p are given by:

(1) if $a \in \mathrm{Sub}(p)$ then $a \xrightarrow{a} o$ is an edge,

(2) if $ax \in \mathrm{Sub}(p)$ then $ax \xrightarrow{a} x$ is an edge,

(3) if $a+y \in \mathrm{Sub}(p)$ then $a+y \xrightarrow{a} o$ is an edge,

(4) if $ax+y \in \mathrm{Sub}(p)$ then $ax+y \xrightarrow{a} x$ is an edge.

(If p has only infinite branches, the termination node o can be discarded.)

(ii) Let $p \xrightarrow{a_0} p_1 \xrightarrow{a_1} \dots$ be a maximal path in G_p (i.e. infinite or terminating in o). Then $a_0 a_1 \dots$ is a <u>trace</u> of p.

(iii) p is <u>perpetual</u> if all its traces are infinite.

(iv) $\|p\|$, the <u>breadth</u> of p, is the outdegree of the root of G_p. Here $\|p\| \in \mathbb{N}$, or $\|p\|$ is infinite.

(v) p is <u>finitely branching</u> if for all $q \in Sub(p)$, $\|q\|$ is finite.

(vi) p is <u>uniformly finitely branching</u> if $\exists n \in \mathbb{N}\ \forall q \in Sub(p)\ \|q\| < n$.

The proof of the following proposition is routine and omitted.

<u>PROPOSITION</u>. *The uniformly finitely branching processes constitute a subalgebra of* A^∞.□

The next theorem gives further criteria for recursive definability of processes.

<u>THEOREM 3.3.1</u>. (i) *Recursively defined processes are finitely branching.*

(ii) *Moreover, processes recursively defined using only* $+,\cdot$ *are uniformly finitely branching.*

(iii) *There exists a process* $p \in R(\underline{A}^\infty(+,\cdot,\|,\underline{\|}))$ *which is not uniformly finitely branching.*

<u>PROOF</u>. (i),(ii): straightforward. (iii): Consider the solution $\underline{X}$ of $X = a + b(Xc \| Xd)$. It is proved in [4] that $\underline{X}$ is not uniformly finitely branching. □

<u>THEOREM 3.3.2</u>. *Let* E_X *be a system of guarded fixed point equations over* $+,\cdot,A,X$. *Suppose the solutions* $\underline{X}$ *are perpetual. Then they are regular.*

<u>PROOF</u>. Since the $\underline{X}_i$ in $\underline{X} = \{\underline{X}_1,\dots,\underline{X}_m\}$ are perpetual, we have $\underline{X}_i \cdot p = \underline{X}_i$ for every $p \in A^\infty$. Therefore every product $X_i \cdot t$ in E_X may be replaced by X_i without altering the solution vector $\underline{X}$. This leads to a system E'_X where only prefix multiplication is used, or in other words, containing only linear equations (see 1.3). Hence the solutions $\underline{X}$ of E'_X are regular, by Theorem 2.1.1(i). □

<u>COROLLARY 3.3.3</u>. *Let* p *be a finitely branching and perpetual process. Let* Sub(p) *be generated using* $+,\cdot$ *by a finite subset* $X \subseteq Sub(p)$. *Then* p *is regular.*

<u>PROOF</u>. Say $X = \{q_1,\dots,q_m\}$. Since p is finitely branching, and hence also the q_i are finitely branching, we can find guarded expressions (using $+,\cdot$ only) $T(X_1,\dots,X_n)$ and $T_i(X_1,\dots,X_{m_i})$ such that

$$\begin{cases} p = T(p_1,\dots,p_n) \\ q_i = T_i(q_{i1},\dots,q_{im_i}),\ i=1,\dots,m. \end{cases}$$

Here the p_k $(k=1,\dots,n)$ and q_{ij} $(i=1,\dots,m;\ j=1,\dots,m_i)$ are by definition in Sub(p); therefore the p_k and q_{ij} can be expressed in $q_1,\dots,q_m$. So there are guarded $+,\cdot$-terms T' and T'_i such that

$$\begin{cases} p = T'(q_1,\dots,q_m) \\ q_i = T'_i(q_1,\dots,q_m),\ i=1,\dots,m. \end{cases}$$

Since p is perpetual, every subprocess of p is perpetual; in particular the q_i

$(i=1,\ldots,m)$. By the preceding theorem p and the q_i are now regular. □

Remark. The condition 'finitely branching' is necessary in this Corollary, as the following example shows. Consider

$$p = \sum_{i=1} a^i b^\omega,$$

or more precisely, p is the projective sequence $(p_1,p_2,\ldots,p_n,\ldots)$ with

$$p_n = \sum_{i=1}^{n} a^i b^{n-i}.$$

Then the canonical transition diagram G_p is as in Figure 1. Now p is perpetual and

$$\mathrm{Sub}(p) = \{p\} \cup \{a^n b^\omega \mid n \geq 0\},$$

so $\mathrm{Sub}(p)$ is generated by its finite subset $\{p, b^\omega\}$; yet p is not regular.

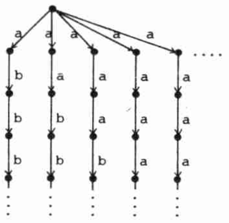

Figure 1.

3.4. Recursive definitions for Bag, Counter and Stack. Let D be a finite set of data values. Let $A = D \cup \underline{D}$, where $\underline{D} = \{\underline{d} \mid d \in D\}$. Let us first consider a bag B over D; its actions are:

d: add d to the bag

$\underline{d}$: take d from the bag.

The initial state of B is empty. Thus the behaviour of B is some process in A^∞.

Similarly the stack S is represented by a process in A^∞.

A counter C is a process in $\{0,p,s\}^\infty$ where the actions $0,p,s$ have the following meaning:

0: assert that C has value 0

p: add one to the counter

s: subtract one from the counter (if possible).

Now the following recursive definitions of B, C and S can be given (see Table 3):

$B = \sum_{d \in D} d \cdot (\underline{d} \parallel B)$
$\begin{cases} S = \sum_{d \in D} d \cdot T_d \cdot S \\ T_d = \underline{d} + \sum_{b \in D} b \cdot T_b \cdot T_d \quad \text{for all } d \in D \end{cases}$
$\begin{cases} C = (0 + s \cdot H) \cdot C \\ H = p + s \cdot H \cdot H \end{cases}$

Table 3.

For a discussion of the equation for Bag B in Table 3, see [5]. The recursive definition of Stack S is equivalent to one of Hoare [8]. The equations for Counter C are similar to those for S when $D = \{s\}$ and p stands for $\underline{s}$. It only has the extra option for testing on value 0. In the following section some further information on these recursive definitions will be given.

4. TECHNICAL ASPECTS OF DIFFERENT RECURSIVE DEFINITION MECHANISMS

In this final section we will provide some information about particular recursive definition mechanisms. Namely: systems of equations (over $+,\cdot$) have greater expressive power than single recursion equations (Theorem 4.1); adding $\|$ to $+,\cdot$ yields more expressive power (Theorem 4.2); adding communication yields more expressive power.

THEOREM 4.1. C *(Counter) and* S *(Stack) as in Table 3 cannot be defined by means of a single equation over* $A^{\infty}(+,\cdot)$.

PROOF. Immediately, by Theorem 3.3.2 and the fact that C and S are clearly not regular. □

THEOREM 4.2. B *(Bag) cannot be recursively defined over* $A^{\infty}(+,\cdot)$ *(provided its domain of values contains at least two elements).*

PROOF. First let us note that the proviso in the statement of the theorem is necessary: If the domain of values $D = \{a\}$ then B as in Table 3 is recursively defined by $B = a(\underline{a}\|B)$. Now it is not hard to see that an equivalent definition for B can be given without $\|$:

$$\begin{cases} B = aCB \\ C = \underline{a} + aCC. \end{cases}$$

(This can be seen by constructing the process graph. Or: note that the behaviour of Bag with singleton value domain is identical to that of a Stack over the same domain, and use the recursive definition for S in Table 3.)

Let D be the domain of values and suppose $D = \{a,b\}$. (The case $D = \{a_1,\ldots,a_n\}$, $n \geqslant 2$, follows easily.) Then Bag B over $\{a,b\}$ is defined by

$$B = a(\underline{a} \| B) + b(\underline{b} \| B).$$

(Some alternative and equivalent definitions are: $B = a(\underline{a}\|B) \| b(\underline{b}\|B)$, or $B = (a\underline{a} + b\underline{b}) \lfloor\!\lfloor B$, or $B = (a\underline{a}\|b\underline{b}) \lfloor\!\lfloor B$, or $B = \{X\|Y,\ X = a(\underline{a}\|X),\ Y = b(\underline{b}\|Y)\}$, or the system of recursion equations $\{B = X_1\|Y_1,\ X_1 = aX_2X_1,\ X_2 = \underline{a} + aX_2X_2,\ Y_1 = bY_2Y_1,\ Y_2 = \underline{b} + bY_2Y_2\}$. The last system is of interest since it shows - after the present theorem is proved - that the algebra $R(A^{\infty}(+,\cdot))$ is not closed under $\|$.)

We will show that B cannot recursively be defined over $+,\cdot$, i.e. $B \notin R(A^{\infty}(+,\cdot))$. We start with some observations about B. Its canonical process graph is as in Figure 2(a):

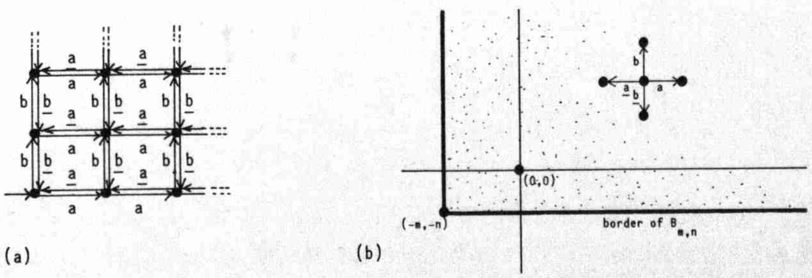

Figure 2 (a) (b)

The subprocesses of B are the $B_{m,n}$ $(m,n \geqslant 0)$ where $B = B_{0,0}$; the $B_{m,n}$ satisfy for all $m,n \geqslant 0$:

$$B_{m,n} = aB_{m+1,n} + \underline{a}B_{m-1,n} + bB_{m,n+1} + \underline{b}B_{m,n-1}$$

with the understanding that summands in which a negative subscript appears, must vanish. (E.g.: $B_{1,0} = aB_{2,0} + \underline{a}B_{0,0} + bB_{1,1}$.) Graphically we display the $B_{m,n}$ in the "a-b-plane" as in Figure 2(b) on the preceding page. Here the root of the displayed subprocess $B_{m,n}$ is at (0,0) and all traces of $B_{m,n}$ stay confined in the indicated quadrant.

(The subprocesses $B_{m,n}$ are by Theorem 3.2.1 generated by $B,a,b,\underline{a},\underline{b}$ via $+,\cdot,\|,\lfloor\!\lfloor$; indeed it is easy to compute that $B_{m,n} = a^m \| b^n \| B$.)

Now suppose for a proof by contradiction that $B \in R(A^{\infty}(+,\cdot))$. Then, by Corollary 3.2.2, the collection of subprocesses $B_{m,n}$ $(m,n \geqslant 0)$ is finitely generated using $+,\cdot$ only by say $\underline{X}_1,\ldots,\underline{X}_k$. Let the $B_{m,n}$ therefore be given by

$$B_{m,n} = T_{m,n}(\underline{X})$$

where $T_{m,n}(X)$ are terms involving only $+,\cdot,a,\underline{a},b,\underline{b},X$. (Here $X = (X_1,\ldots,X_k)$ contains the variables of the system of recursive definitions yielding solutions $\underline{X}$ and used to define B.)

We may assume that every occurrence of X_i in $T_{m,n}$ is immediately preceded by some $u \in A = \{a,\underline{a},b,\underline{b}\}$. If not, we expand the corresponding $\underline{X}_i$ as

$$\underline{X}_i = a\underline{X}_{i1} + \underline{a}\,\underline{X}_{i2} + b\underline{X}_{i3} + \underline{b}\,\underline{X}_{i4}$$

(some summands possibly vanishing) and replace $\underline{X}_i$ by its subprocesses $\underline{X}_{i1},\ldots,\underline{X}_{i4}$ in the set of generators $\underline{X}$.

Further, we may take $T_{m,n}$ to be in normal form w.r.t. rewritings $(x+y)z \to xz+yz$. Now consider an occurrence of X_i in $T_{m,n}$. Then X_i is contained in a subterm of the form uX_iP, $u \in A$, P maybe vanishing. Take P maximal so, i.e. uX_iP is not a proper subterm of some uX_iPQ.

Then it is easy to see that $\underline{X}_i\underline{P}$ (where $\underline{P}$ is P after substituting $\underline{X}_j$ for X_j, $j = 1,\ldots,k$) is a subprocess of $B_{m,n}$, i.e. $\underline{X}_i\underline{P} = B_{k,\ell}$ for some k,ℓ.

Thus we find that all generators are left-factors of some subprocess of B. If such a left-factor $\underline{X}_i$ is perpetual, then clearly in the factorization $\underline{X}_i\underline{P} = B_{k,\ell}$ we have already $\underline{X}_i = B_{k,\ell}$. For proper factorizations (i.e. where $\underline{X}_i$ is not perpetual) we have the following remarkable properties:

<u>CLAIM</u>. *Let* $PQ = B_{m,n}$ *be a factorization of a subprocess of* B. *Suppose* P *is not perpetual. Then:*

(i) all finite traces of P *end in the same point of the* a,b-*plane;*

(ii) P *determines* n,m *and* Q *uniquely (i.e. if moreover* $PQ' = B_{m',n'}$, *then* $Q = Q'$ *and* $n,m = n',m'$*).*

<u>Proof of the claim</u>. (i) Consider Figure 3(a) on the next page. Suppose P has traces σ,σ' ending in different points (k,ℓ) and (k',ℓ'). Then Q has a trace ρ such that $\sigma\rho$ leads

to the border of $B_{m,n}$. However, then the trace $\sigma'\rho$ exceeds this border, contradicting the assumption $PQ = B_{m,n}$.

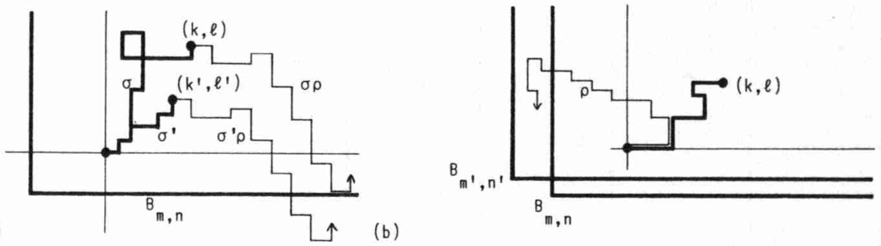

Figure 3 (a) (b)

(ii) To see that $B_{m,n}$ is uniquely determined, consider Figure 3(b) above and let $PQ' = B_{m',n'}$. Say that P's finite traces terminate in (k,ℓ). Now consider a trace ρ in P which avoids this 'exit point'. (Here the argument breaks down for the case of a singleton value domain D = {a}.) Since (k,ℓ) is P's only exit point (by (i)), ρ is confined to stay in P as long as it avoids (k,ℓ). But then a trace ρ as in Figure 3(b) which enters the symmetrical difference of the areas occupied in the a,b-plane by $B_{m,n}$ and $B_{m',n'}$ leads to an immediate contradiction.

The unicity of Q is proved by similar arguments. (Note that Q is itself a subprocess of B.)

This ends the proof of the Claim. A corollary of the Claim is that in the equations $B_{m,n} = T_{m,n}(\underline{X})$ every $\underline{X}_i\underline{P}$ (as defined above) can be replaced by B_{k_i,ℓ_i} <u>depending on i alone</u>. Therefore the set of generators can be taken to consist of a finite subset of the collection of $B_{m,n}$, say $\{B_{k_i,\ell_i} \mid i = 1,\ldots,p\}$.

However, by Corollary 3.3.3, B must then be regular, an evident contradiction. Hence B cannot be recursively defined with + and • alone. □

We conclude this paper with the observation that communication yields strictly more expressive power. As a preparation we need another criterion for recursive definability:

<u>THEOREM 4.3</u>. *Let* $\underline{X}$ *be recursively defined over* $A^{\infty}(+,\cdot,\|,\llcorner\!\llcorner)$ *and suppose* $\underline{X}$ *is not finite* ($\underline{X} \notin A_{\omega}$). *Then* $\underline{X}$ *has an infinite regular (i.e. eventually periodic) trace.* □

The proof requires a syntactical analysis for which we refer to [4]. The intuition of the proof can be hinted at by the following example; here we write for variables X_i, X_j in a system $E_X = \{X_i = T_i(X) \mid i = 1,\ldots,n\}$:

$X_i \overset{w}{\dashrightarrow} X_j$ if X_j occurs in $T_i(X)$ and the 'path' w 'leads to' this occurrence of X_j.

<u>Example</u>. Let E_X be $\{X_1 = a(X_2 \llcorner\!\llcorner X_3) + a,\ X_2 = bc(X_3 \| X_3),\ X_3 = aaX_1X_3\}$, then

$$X_1 \overset{a}{\dashrightarrow} X_2 \overset{bc}{\dashrightarrow} X_3 \overset{aa}{\dashrightarrow} X_1,$$

hence $\underline{X}_1$ contains a trace $(abcaa)^{\omega}$.

THEOREM 4.4. *There is a process* $p \in \{a,b\}^{\infty}$ *which cannot be recursively defined in* $\{a,b\}^{\infty}(+,\cdot,\|,⌊\!\!\lfloor)$ *but which can be recursively defined in* $\{a,b,c,d,\delta\}^{\infty}(+,\cdot,\|,⌊\!\!\lfloor,|,\partial_H)$ *where* H *and the communication function are appropriately chosen.*

PROOF. Consider the alphabet $A = \{a,b,c,d,\delta\}$, with $H = \{c,d\}$ as set of subatomic actions and with communication function given by: $c|c = a$; $d|d = b$; other communications equal δ. Now let

$$p = ba(ba^2)^2(ba^3)^2(ba^4)^2\ldots$$

and consider the system of equations $\{X = cXc + d,\ Y = dXY,\ Z = dXcZ\}$. It turns out that $p = \partial_H(d\cdot cY\|Z)$. To prove this, consider the processes

$$p_n = \partial_H(dc^nY\|Z)$$

for $n \geq 1$. Now we claim that for all $n \geq 1$: $p_n = ba^nba^{n+1}p_{n+1}$, which immediately yields the result. Proof of the claim:

$$\begin{aligned}p_n &= \partial_H(dc^nY\|Z) = \partial_H(dc^nY\|dXcZ) = ba^n\partial_H(Y\|Xc^ncZ) = \\ &ba^n\partial_H(dXY\|(cXc+d)c^{n+1}Z) = ba^nb\partial_H(XY\|c^{n+1}Z) = \\ &ba^nba^{n+1}\partial_H(Xc^{n+1}Y\|Z) = ba^nba^{n+1}\partial_H(dc^{n+1}Y\|Z) = \\ &ba^nba^{n+1}p_{n+1}.\end{aligned}$$

The fact that p cannot be recursively defined without communication is an immediate consequence of Theorem 4.3. □

REFERENCES

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

[2] DE BAKKER, J.W. & J.I. ZUCKER, Processes and the denotational semantics of concurrency, Information and Control, Vol.54, No.1/2, p.70-120, 1982.

[3] BERGSTRA, J.A. & J.W. KLOP, Process algebra for communication and mutual exclusion, Report IW 218/83, Mathematisch Centrum, Amsterdam 1983.

[4] BERGSTRA, J.A. & J.W. KLOP, The algebra of recursively defined processes and the algebra of regular processes, Report IW 235/83, Mathematisch Centrum, Amsterdam 1983.

[5] BERGSTRA, J.A. & J.W. KLOP, Algebra of Communicating Processes, in: Proceedings of the CWI Symposium Mathematics and Computer Science (eds. J.W. de Bakker, M. Hazewinkel and J.K. Lenstra), CWI Monograph Series, North-Holland. To appear.

[6] HENNESSY, M., A term model for synchronous processes, Information and Control, Vol.51, No.1(1981), p.58-75.

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

[8] HOARE, C.A.R., A Model for Communicating Sequential Processes, in: "On the Construction of Programs" (ed. R.M. McKeag and A.M. McNaghton), Cambridge University Press, 1980 (p.229-243).

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