stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 235/83 SEPTEMBER

J.A. BERGSTRA & J.W. KLOP

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

Preprint

kruislaan 413 1098 SJ amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

CWI BIBLIOTHEEK
3 0054 00095 2151

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

The Mathematical Centre, founded 11 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.).

bgF11, bgF12, bgF32, bgF43

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

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

Copyright © 1983, Mathematisch Centrum, Amsterdam

The algebra of recursively defined processes and the algebra of regular processes *)

by

J.A. Bergstra & J.W. Klop

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. An interpretation of CSP in the regular processes is presented. As an example of recursively defined processes, bag and stack are discussed in detail. It is shown that the bag cannot be recursively defined without merge.

We introduce fixed point algebras which have interesting applications in several proofs. An example is presented of a fixed point algebra which has an undecidable word problem.

KEY WORDS & PHRASES: *concurrency, nondeterministic process, merge, process algebra, regular processes, recursively defined processes, fixed point algebra*

---

*) This report is not for review as it will be published elsewhere.

CONTENTS

INTRODUCTION

ACP, algebra of communicating processes, was introduced in BERGSTRA & KLOP [4]. It combines a purely algebraic formulation of a part of MILNER's CCS [13] with an algebraic presentation of the denotational semantics of processes as given by DE BAKKER & ZUCKER in [1,2]; moreover it includes two laws on communication of atomic actions which are also present in HENNESSY [8].

The ingredients of ACP are the following:

- A finite set A of so-called *atomic actions*, 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 these:

$x+y = y+x$	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
$x\|\|y = x\lfloor\!\lfloor y + y\lfloor\!\lfloor x + x\|y$	CM1
$a\lfloor\!\lfloor x = a.x$	CM2
$(ax)\lfloor\!\lfloor y = a(x\|\|y)$	CM3
$(x+y)\lfloor\!\lfloor z = x\lfloor\!\lfloor z + y\lfloor\!\lfloor z$	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

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

The operations $\lfloor\!\lfloor$ (*left merge*) and | (*communication merge*) are auxiliary ones. Our primary interest remains for +,•,||. 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 an enormous variety of models. In [3,4,5] we investigated the model A^∞. In the present paper we take into consideration graph theoretic models as well. Especially we consider *finitely branching* graphs.

Four types of models thus emerge:

(i) A_ω, the initial model of ACP seen as an equational specification over the signature with a constant for each atom.

(ii) A_ω mod n, (also written as A_n) for $n \geqslant 1$: a homomorphic image of A_ω obtained by identifying two processes in A_ω if their trees coincide up to depth n.

(iii) A^∞; this is the projective limit of the structures A_n.

(iv) graph theoretic models.

More information on these matters can be found in [3-6].

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 $\underline{A} = A - \{\delta\}$.

Under the assumption that $a|b = \delta$ for all $a,b \in A$, we may add $\|$ and $\between$ to the signature of these algebras, thus obtaining

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

Of course these structures can be constructed immediately without any reference to communication. Let PA be the following axiom system:

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

Then $\underline{A}_\omega(+, \cdot, \Vert, \between)$ is just the initial algebra of PA.

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 $+, \cdot, \Vert, \between, |, \partial_H, \delta$ two kinds of terms containing variables $X_1, \ldots, X_n$ are of particular importance:

(i) *Linear terms*. Linear terms 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 $T + T'$, $T \cdot T'$, $\partial_H(T)$, $T \Vert T'$, $T \between T'$, $T | T'$ (for every T').

A term T is guarded if it is not unguarded.

1.4. Process graphs.

Process graphs (or, as we will sometimes call them: transition diagrams) constitute a very useful tool for the description of processes. In this section we will consider finite process graphs (possibly containing cycles). Finite process graphs over A will find a semantics in A^∞ via a system of recursion

equations.

A *process graph* g for an action alphabet A is a rooted directed graph with edges labelled by elements of A. (Here g may be infinite and may contain 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 $U = \{(a,y) \mid g\colon s \xrightarrow{a} t \ \&\ t \text{ has positive outdegree}\}$,

$V = \{a \mid \exists t \ \ g\colon s \longrightarrow t \ \&\ t \text{ has outdegree } 0\}$.

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_1, g_2 be two process graphs for A. We define new process graphs as follows:

$g_1 + g_2$ results by glueing together the roots of g_1 and g_2,

$g_1 \cdot g_2$ results by glueing together the root of g_2 and all endpoints of g_1,

$\partial_H(g_1)$ results by replacing all labels $a \in H$ by δ in g_1,

$g_1 \| g_2$ is the cartesian product of the node sets of g_1 and g_2 provided with labeled edges as follows:

(i) $a\colon (s_1,s_2) \longrightarrow (s_1',s_2)$ if in g_1 we have $a\colon s_1 \longrightarrow s_1'$

(ii) $a\colon (s_1,s_2) \longrightarrow (s_1,s_2')$ if in g_2 we have $a\colon s_2 \longrightarrow s_2'$

(iii) $a\colon (s_1,s_2) \longrightarrow (s_1',s_2')$ if for some $b,c \in A$ we have $b|c = a$ and $b\colon s_1 \longrightarrow s_1'$ in g_1, $c\colon s_2 \longrightarrow s_2'$ in g_2.

$g_1 \between g_2$ is defined like $g_1 \| g_2$, but leaving out all transitions of types (ii) and (iii) if s_1 is the root of g_1.

$g_1 | g_2$ is defined like $g_1 \| g_2$ but leaving out all transitions of types (i) and (ii) if s_1 resp. s_2 is the root of g_1, g_2.

Of course we have $[\![g_1 + g_2]\!] = [\![g_1]\!] + [\![g_2]\!]$ etc.

2. REGULAR PROCESSES

2.1. The algebra of regular processes.

For $p \in A^\infty$ we define the collection Sub(p) of *subprocesses* of p as follows:

$p \in Sub(p)$

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

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

DEFINITION. $p \in A^\infty$ is *regular* if Sub(p) is finite.

NOTATION. $r(A^\infty)$ denotes the collection of regular processes in A^∞.

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* $+, \cdot, \|, \between, |, \partial_H$. *Hence* $r(A^\infty)$ *is a subalgebra of* A^∞.
(iii) $r(A^\infty)$ *contains exactly the solutions of finite systems of guarded linear equations.*

PROOF. (i) and (iii) are standard; (ii) is an immediate consequence of the fact that the operations $+, \cdot, \|, \between, |, \partial_H$ acting on graphs preserve finiteness. (Cf. [6], Section 2.2) □

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 program* 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 true; b & p can only be performed if b is true. It is usual to abbreviate true & p to p. All variables x must occur in X. Further, C is an element of a set of channel names.

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

REMARK. 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 [9].)

However, from our point of view these restrictions are just guide-lines 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 *intermediate semantics* 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 $r(A^\infty_{CSP\text{-}P})$ and so is $[\![P]\!]$ because $r(A^\infty_{CSP\text{-}P})$ is a subalgebra of $A^\infty_{CSP\text{-}P}$.
(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 $+, \cdot, \|, \mathbin{\llcorner}, |, \partial_H$.

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 $(A^\infty)^n \longrightarrow (A^\infty)^n$ 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 $(p,q_1,\ldots,q_{n-1})$.

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. □

NOTATION. With $R(A^\infty)$ we denote the subalgebra of recursively defined processes.

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 $p_1,\ldots,p_n$ be processes in A^∞. Then $A_\omega(p_1,\ldots,p_n)$ will denote the subalgebra of A^∞ generated by $p_1,\ldots,p_n$.

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

PROOF. Let $p \in A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$. Then for some term T we have $p = T(\underline{X}_1,\ldots,\underline{X}_n)$; after substitutions corresponding to $X_i = T_i(X_1,\ldots,X_n)$ we may assume that t is guarded.

On the basis of ACP one can rewrite $T(X_1,\ldots,X_n)$ into the form

$$\sum a_i \cdot R_i(X_1,\ldots,X_n) + \sum b_i.$$

Consequently all immediate subprocesses of p, i.e. the $R_i(\underline{X}_1,\ldots,\underline{X}_n)$, are in $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ as well. □

This theorem gives a useful criterion for recursive definability (to be used in Section 5):

COROLLARY 3.2.2. (i) *Let* $p \in R(\underline{A}^\infty(+,\cdot,\|,\between\!\!\!\between))$. *Then* Sub(p) *is finitely generated using* $+,\cdot,\|,\between\!\!\!\between, a \in A$.
(ii) *Likewise for the restricted signature of* $+,\cdot,a \in A$. □

3.3. Finitely branching processes.

DEFINITION. Let $p \in A^\infty$. (i) Then $\mathcal{G}_p$ is the *canonical process graph* of p, defined as follows.

The set of *nodes* of $\mathcal{G}_p$ is Sub(p) ∪ {o}. Here o is a termination node. The *root* of $\mathcal{G}_p$ is p. The (labeled and directed) *edges* of $\mathcal{G}_p$ are given by:

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

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

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

(ii) Let $p \xrightarrow{a_0} p_1 \xrightarrow{a_1} \ldots$ be a maximal path in $\mathcal{G}_p$ (i.e. infinite or terminating in o). Then $a_0a_1\ldots$ is a *trace* of p.

(iii) p is *perpetual* if all its traces are infinite.

(iv) $\|p\|$, the *breadth* of p, is the outdegree of the root of $\mathcal{G}_p$. Here $\|p\| \in \mathbb{N}$, or $\|p\|$ is infinite.

(v) p is *finitely branching* if for all $q \in$ Sub(p), $\|q\|$ is finite.

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

The proof of the following proposition is routine and omitted.

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

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

THEOREM 3.3.1. (i) *Recursively defined processes are finitely branching.*
(ii) *Moreover, processes recursively defined using only* $+,\cdot$ *are uniformly finitely branching.*
(iii) *There exists a process* $p \in R(\underline{A}^\infty(+,\cdot,\|,\between))$ *which is not uniformly finitely branching.*

PROOF. (i),(ii): straightforward.
(iii): Consider the solution $\underline{X}$ of

$$X = a + b(Xc \parallel Xd).$$

Define with induction on n the following processes:

$$\begin{cases} p_0 = a \\ p_{n+1} = p_n \cdot c \parallel p_n \cdot d. \end{cases}$$

Claims: (1) $\forall n \in \mathbb{N}\ \exists q_n \in \mathrm{Sub}(\underline{X})\ \partial_{\{b\}}(q_n) = p_n$.

(2) $\|p_n\| = 2^n$

(3) $\|\partial_H(x)\| \leqslant \|x\|$

Here (1) states that the p_n are 'almost' subprocesses of $\underline{X}$. Claim (3) is the general and obvious fact that the projection operator ∂_H certainly cannot increase the breadth of its argument. Combined with the observation of Claim (2) that the breadths of the p_n are unbounded, the claim yields the result.

We will now prove Claim (1) and (2). (The proof of Claim (3) is straightforward.)

Proof of Claim (1).
Let

$$\begin{cases} q_0 = \underline{X} \\ q_{n+1} = q_n \cdot c \parallel q_n \cdot d. \end{cases}$$

We will prove that $\partial_{\{b\}}(q_n) = p_n$, for all $n \geq 0$.

$n = 0$: $\partial_{\{b\}}(q_0) = a + \delta(\ldots) = a = p_0$.

Induction hypothesis: $\partial_{\{b\}}(q_n) = p_n$ and $q_n \in Sub(\underline{X})$.

To prove: $\partial_{\{b\}}(q_{n+1}) = p_{n+1}$ and $q_{n+1} \in Sub(\underline{X})$.

Since $q_n \in Sub(\underline{X})$, we have $q_n c \in Sub(\underline{X}c)$ and $q_n d \in Sub(\underline{X}d)$.

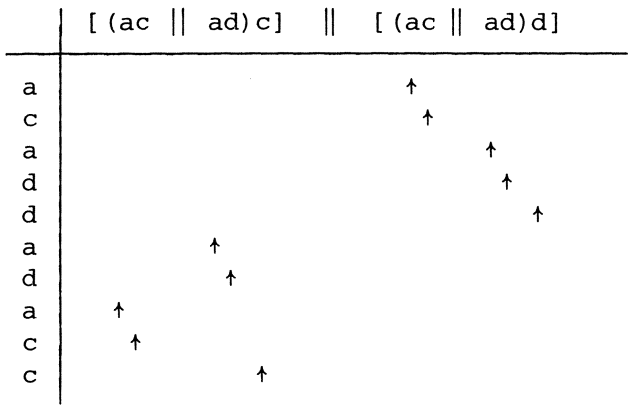
So $q_{n+1} = q_n c \parallel q_n d \in Sub(\underline{X}c \parallel \underline{X}d) \subseteq Sub(a + b(\underline{X}c \parallel \underline{X}d)) = Sub(\underline{X})$.

Furthermore, $\partial_{\{b\}}(q_{n+1}) = \partial_{\{b\}}(q_n c \parallel q_n d)$ = (since there is no nontrivial communication) $\partial_{\{b\}}(q_n c) \parallel \partial_{\{b\}}(q_n d) = (\partial_{\{b\}} q_n)c \parallel (\partial_{\{b\}} q_n)d = p_n c \parallel p_n d = p_{n+1}$.

Proof of Claim (2). We will give a sketch of the proof.
Define the set D of "inside-out traces" as follows:

$$\begin{cases} a \in D \\ \sigma, \tau \in D \Rightarrow \sigma c \tau d \ , \ \sigma d \tau c \in D. \end{cases}$$

Now consider e.g. $p_2 = [(ac \parallel ad)c] \parallel [(ac \parallel ad)d]$.
D contains some traces of p_2, such as acaddadacc. Traces in D arise from an "inside-out evaluation" of the merges in the unevaluated expression for p_2, as suggested by the following figure:



Moreover, traces in D can be evaluated in precisely one way. (This does not hold for traces not in D; e.g. aacdaadcdc may be obtained starting from each of the four occurrences of 'a' in the expression for p_2.) Hence the four summands of p_2, corresponding to the four occurrences of 'a' in the expression

for p_2, are different since they contain a trace which is characteristic for them. So $\|p_2\| = 4$. Likewise one proves the general statement in Claim (2). □

THEOREM 3.3.2. *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.*

PROOF. Since the $\underline{X}_i$ in $\underline{X} = \{\underline{X}_1,\ldots,\underline{X}_m\}$ 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). □

COROLLARY 3.3.3. *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.

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

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

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

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

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. □

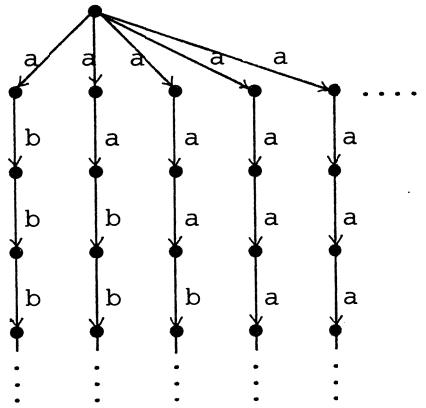
REMARK. The condition 'finitely branching' is necessary in this Corollary, as the following example shows. Consider

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

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 of p is



Now p is perpetual and $Sub(p) = \{p\} \cup \{a^n b^\omega \mid n \geqslant 0\}$, so Sub(p) is generated by its finite subset $\{p,b^\omega\}$; yet p is not regular.

3.4. Interesting examples of recursive definitions.

We will consider 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).

Of course these descriptions are rather informal; a much more precise defi-

nition could be given along the lines of BERGSTRA & KLOP [7].

Here we are interested in recursive definitions for B,S and C:

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

REMARKS. The equation for B has been discussed in detail in [7].

The recursive definition of S is equivalent to one of HOARE [10].

The equations for 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 zero.

4. UNDECIDABILITY OF THE WORD PROBLEM IN FIXED POINT ALGEBRAS

As in 3.2, for $p_1,..,p_n \in A^\infty$ we denote with $A_\omega(p_1,...,p_n)$ the subalgebra of A^∞ generated by $p_1,...,p_n$.

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

REMARK. Let us denote with $A_\omega[X_1,...,X_n]$ the free ACP algebra generated over new names $X_1,...,X_n$. For each set of interpretations $\underline{X}_1,...,\underline{X}_n$ there is a homomorphism

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

Now suppose that E_X is a system of guarded fixed point equations for $X = \{X_1,...,X_n\}$. Then

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

is the algebra obtained by dividing out the congruence generated by E_X. On

the other hand, let $\underline{X}_1,\ldots,\underline{X}_n$ be the unique solutions of E_X in A^∞. There is again a homomorphism

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

Both algebras $A_\omega[X_1,\ldots,X_n]/E_X$ and $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ may be vastly different however. Being an initial algebra of a finite specification, $A_\omega[X_1,\ldots,X_n]/E_X$ is semicomputable. As we shall see, $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ is in general cosemicomputable and we will present an example where it is not computable indeed.

DEFINITION. A *fixed point algebra* is an algebra

$$A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$$

where the $\underline{X}_i$ are solutions in A^∞ of some system of guarded fixed point equations E_X.

DEFINITION. Consider an algebra $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$. The *word problem* for this algebra consists in deciding for given terms T and R over $+,\cdot,\|,\perp\!\!\!\perp,|,\partial_H,A,X_1,\ldots,X_n$ whether or not

$$A_\omega(\underline{X}_1,\ldots,\underline{X}_n) \models T = R.$$

THEOREM 4.1. *For each fixed point algebra* $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ *the word problem is co-r.e.*

PROOF. Consider $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ and let $\underline{X}_i$ solve E_X. Let T,R be two terms. Then

$$A_\omega(\underline{X}_1,\ldots,\underline{X}_n) \models T \neq R \iff$$

$$\exists k \quad A_\omega(\underline{X}_1,\ldots,\underline{X}_n)/\equiv_k \models T \neq R.$$

Here $\equiv_k$ is the congruence on A^∞ defined as follows: if $p = (p_1,p_2,\ldots)$ and $q = (q_1,q_2,\ldots)$ then $p \equiv_k q \iff p_i = q_i$ for $i = 1,\ldots,k$.

Now $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)/\equiv_k$ is a finite algebra which can be uniformly computed from E_X and k; in this algebra $T \neq R$ can be effectively decided.

Consequently inequality of T and R is semicomputable and equality is cosemicomputable (co-r.e.). □

THEOREM 4.2. *There is a fixed point algebra* $A_\omega(\underline{X}_1,\ldots,\underline{X}_n)$ *with undecidable word problem.*

PROOF. Let K be a recursively enumerable but not recursive subset of $\mathbb{N}$. The elements of K can be recognized by a register machine on three counters in which the argument n is initially stored in the first counter (see HOPCROFT & ULLMAN [11]).

The counters will be denoted by x,y,z; we use α as a metavariable ranging over x,y,z. The register machine has the following instructions:

$\alpha := \alpha + 1$; goto j

$\alpha := \alpha \dot{-} 1$; goto j

if $\alpha = 0$ then goto j_1 else goto j_2

stop.

A program for the register machine is a numbered sequence $I_1,\ldots,I_k$ of such instructions, where obviously $j,j_1,j_2 \in \{1,\ldots,k\}$ for all instructions.

Let P be a program which recognizes K in the sense that $P(n,0,0) \longrightarrow$ stop iff $n \in K$. Let ℓ be the number of instructions of P. We will define a fixed point algebra in which P can be represented. The alphabet A, the set of subatomic actions H and the set X of variables for the system of recursion equations for this fixed point algebra are as follows:

$$A = \{s_\alpha, p_\alpha, 0_\alpha, b, \underline{stop}, \delta\}$$
$$H = A - \{b, \underline{stop}, \delta\}$$
$$X = \{C_\alpha, H_\alpha, B, X_i \mid \alpha = x,y,z;\ i = 1,\ldots,\ell\}.$$

The communication function $.|.$ is given by

$$s_\alpha | s_\alpha = p_\alpha | p_\alpha = 0_\alpha | 0_\alpha = b \quad (\alpha = x,y,z)$$

All other communications equal δ.

Before giving the system of equations E_X we define a map $\{\!\!\{\ \}\!\!\}$ from register machine instructions to process algebra expressions over A,X:

$\{\!\!\{ \alpha := \alpha + 1;$ goto j $\}\!\!\} = s_\alpha \cdot X_j$

$\{\!\!\{ \alpha := \alpha \dot{-} 1;$ goto j $\}\!\!\} = (0_\alpha + p_\alpha) \cdot X_j$

$\{\!\!\{$ if $\alpha = 0$ then goto j else goto j' $\}\!\!\} = 0_\alpha \cdot X_j + p_\alpha \cdot s_\alpha \cdot X_{j'}$

$\{\!\!\{$ stop $\}\!\!\} =$ stop

Let E_X be the system of guarded recursion equations:

$$\begin{cases} C_\alpha = (0_\alpha + s_\alpha \cdot H) \cdot C & (\alpha = x,y,z) \\ H_\alpha = (p_\alpha + s_\alpha \cdot H) \cdot H & (\alpha = x,y,z) \\ B = b \cdot B & \\ X_j = \{I_j\} & (j = 1,\ldots,\ell) \end{cases}$$

(Note that the $\{I_j\}$ contain variables from $X_1,\ldots,X_\ell$.)

Further, note that the expression $H_\alpha^n \cdot C_\alpha$ denotes the state of counter α containing n, and consider the following term representing P:

$$\partial_H(X_1 \| H_x^n \cdot C_x \| C_y \| C_z).$$

We claim that

$$A_\omega(\underline{C}_x, \underline{H}_x, \underline{C}_y, \underline{H}_y, \underline{C}_z, \underline{H}_z, \underline{X}_1, \ldots, \underline{X}_\ell, \underline{B}) \models \partial_H(X_1 \| H_x^n C_x \| C_y \| C_z) = B$$

if and only if the computation P(n,0,0) diverges.

The straightforward proof of the claim is omitted.

Thus we find that the predicate $n \notin K$ is one-one reducible to the word problem of the fixed point algebra $A_\omega(\underline{C}_x, \ldots, \underline{B})$, which shows that this word problem is undecidable. □

5. TECHNICAL ASPECTS OF DIFFERENT RECURSIVE DEFINITION MECHANISMS

In this section we will provide information about particular recursive definition mechanisms. We summarize the results in a sequence of theorems:

THEOREM 5.1. C *(counter) and* S *(stack) cannot be defined by means of a single equation over* $A^\infty(+,\cdot)$.

THEOREM 5.2. B *(bag) cannot be recursively defined over* $A^\infty(+,\cdot)$ *(provided its domain of values contains at least two elements).*

THEOREM 5.3. *If* $\underline{X}$ *is recursively defined over* $A^\infty(+,\cdot,\|,\lfloor\!\lfloor)$ *and* $\underline{X} \notin A_\omega$ *then* $\underline{X}$ *has an infinite regular (i.e. eventually periodic) trace.*

THEOREM 5.4. *There is a process* $p \in \{a,b\}^\infty$ *which cannot be recursively defined in* $\{a,b\}^\infty(+,\cdot,\|,\between)$ *but which can be recursively defined in* $\{a,b,c,d,\delta\}^\infty(+,\cdot,\|,\between,|,\partial_H)$ *where* H *and the communication function are appropriately chosen.*

We will give the proofs of these theorems in the order of increasing length of the proof.

PROOF of Theorem 5.1. Immediately, by Theorem 3.3.2 and the fact that C and S are clearly not regular. □

PROOF of Theorem 5.4. Consider the alphabet $A = \{a,b,c,d,\delta\}$, with $H = \{c,d\}$ as set of subatomic actions and with communication function given by:

$$c|c = a;\ d|d = b;\ \text{other communications equal } \delta.$$

Let

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

and consider the system of equations

$$\begin{cases} X = cXc + d \\ Y = dXY \\ Z = dXcZ. \end{cases}$$

It turns out that $p = \partial_H(dcY\|Z)$. To prove this, consider the processes

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

for $n \geqslant 1$. Now we claim that for all $n \geqslant 1$:

$$p_n = ba^nba^{n+1}p_{n+1}$$

which immediately yields the result. Proof of the claim:

$$p_n = \partial_H(dc^nY\|Z) = \partial_H(dc^nY\|dXcZ) = ba^n\partial_H(Y\|Xc^ncZ) =$$

$$ba^n\partial_H(dXY\|(cXc+d)c^{n+1}Z) = ba^nb\,\partial_H(XY\|c^{n+1}Z) =$$

$$ba^nba^{n+1}\partial_H(Xc^{n+1}Y\|Z) = ba^nba^{n+1}\partial_H(dc^{n+1}Y\|Z) =$$

$$ba^nba^{n+1}p_{n+1}.$$

The fact that p cannot be recursively defined without communication

follows immediately from Theorem 5.3. whose proof will follow now. □

PROOF of Theorem 5.3.

To obtain information about traces of recursively defined processes, we need the concept of a *trace generator* of a term $T(X_1,\ldots,X_n)$. If T is closed, i.e. contains no variables X_i, the trace generators of T as defined below are just the usual traces; if T is open then its trace generators may also contain variables. First we need a 'normal form' of terms:

DEFINITION. (i) On the set of terms built from $+,\cdot,\|,\between,a\in A$ we define the following reduction rules (which may be applied in a context):

$$x+x \to x$$
$$(x+y)z \to xz+yz$$
$$z(x+y) \to zx+zy \quad (*)$$
$$x\|y \to x\between y + y\between x$$
$$a\between x \to ax$$
$$(ax)\between y \to a(x\|y)$$
$$(x+y)\between z \to x\between z + y\between z.$$

A term in which none of these reduction rules can be applied, is in *trace normal form*.

(ii) Let $T \to \ldots \to T'$ be a reduction according to the rules above such that no further step is possible, i.e. T' is in trace normal form. Let

$$T' = \sum_{i=1}^{k} \tau_i$$

where the τ_i are indecomposable w.r.t. +.

Then the τ_i $(i=1,\ldots,k)$ are the *trace generators* of T. (So a trace normal form is a sum of trace generators.)

The reduction rules above correspond to the axioms A3,4 and M1-4, except for the rule (*). Note that we work modulo A1,2,5 (associativity of +,· and commutativity of +).

To show that the trace generators are well-defined by (ii) of the definition, one needs the following fact whose proof is standard and will be

omitted (cf. [4] for a similar proof):

PROPOSITION. (i) *Every reduction using the rules in the preceding definition must terminate.*
(ii) *All reductions with the same start terminate eventually in the same result.* □

EXAMPLE. (i) $a(b+b+ca)a$ reduces to the trace normal form $aba+acaa$, hence has trace generators (here also traces) aba, acaa.
(ii) $X(a+b) \parallel c$ reduces to

$$Xa \mathbin{\underline{\parallel}} c + Xb \mathbin{\underline{\parallel}} c + cXa + cXb,$$

hence has trace generators $Xa \mathbin{\underline{\parallel}} c$, $Xb \mathbin{\underline{\parallel}} c$, cXa and cXb.

In fact we are only interested in the prefix of a trace generator up to the first variable:

DEFINITION. Let τ be a trace generator of the form $w(\ldots(X\text{---}$ where $w \in A^*$ (i.e. w is a term built from $a \in A$ by $\cdot$ only) and where w is followed by some brackets (possibly none) followed by the variable X.

Then the trace generator wX is called a *prefix* of τ.

EXAMPLE. aaabbX is a prefix of the trace generator $aaabb((XX) \mathbin{\underline{\parallel}} b)$.

PROPOSITION. *Let* T,S *be terms such that* T *contains a trace generator with prefix* wX, *and* S *contains a trace generator* vY.

Then T[X:=S], *the term resulting from substituting* S *for the occurrences of* X *in* T, *contains a trace generator with prefix* wvY.

PROOF. Elementary. Note that the left-linearity of $\mathbin{\underline{\parallel}}$ is used as well as the auxiliary rule (*) needed for computing trace generators. □

EXAMPLE. $S(X) \equiv b^2(X^2 \mathbin{\underline{\parallel}} b) + c$ when substituted in $T(X) \equiv a^3[(X(X \mathbin{\underline{\parallel}} a)) \mathbin{\underline{\parallel}} a]$ yields as one of its trace generators

$$a^3b^2[((X^2 \mathbin{\underline{\parallel}} b)b^2((X^2 \mathbin{\underline{\parallel}} b) \mathbin{\underline{\parallel}} a)) \mathbin{\underline{\parallel}} a]$$

which has indeed a^3b^2X as a prefix.

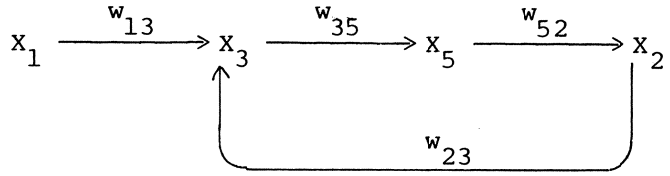
We can now finish the proof of Theorem 5.3. Let $E_X = \{X_i = T_i(X) \mid i = 1,..,n\}$

be a system of guarded equations defining $\underline{X} = \{\underline{X}_1,\ldots,\underline{X}_n\}$ where $\underline{X}_1$ has an infinite trace. Define a directed graph $\mathcal{G}$ on $X = \{X_1,\ldots,X_n\}$, with edges labeled by $w \in A^*$ as follows:

$X_i \xrightarrow{w_{ij}} X_j$ is a labeled edge of $\mathcal{G}$ if:

$w_{ij}X_j$ is a prefix of a trace generator of $T_i(X)$.

We may suppose that every $T_i(X)$ $(i = 1,\ldots,n)$ contains some variable (otherwise the trivial equation $X_i = T_i$ could be eliminated first). Hence $\mathcal{G}$ contains no endnodes. Therefore $\mathcal{G}$, being finite, must contain a path starting with X_1 and eventually cyclic, e.g.:



But then, by the previous Proposition, repeated substitution leads to a process $\underline{X}_1$ with an eventually periodic trace; in our example: $w_{13}(w_{35}w_{52}w_{23})^{\omega}$. □

EXAMPLE. If E_X is $\begin{cases} X_1 = a(X_2 \mathbin{\underline{\|}} X_3) + a \\ X_2 = bc(X_3 \| X_3) \\ X_3 = aaX_1X_3 \end{cases}$

then

$$X_1 \xrightarrow{a} X_2 \xrightarrow{bc} X_3 \xrightarrow{aa} X_1$$

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

PROOF of Theorem 5.2.

The behaviour of a bag B was defined above (Section 3.4):

$$B = \sum_{d \in D} d(\underline{d} \| B).$$

In this subsection we will consider the case that $D = \{a\}$, and the case $D = \{a,b\}$. (The results for the last case generalize at once to the case $D = \{a_1,\ldots,a_n\}$.)

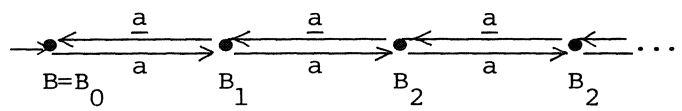
In the first case

$$B = a(\underline{a}\,||\,B) \qquad (*)$$

is equivalent to the following definition without $||$:

$$\begin{cases} B = aCB & (**) \\ C = \underline{a} + aCC \end{cases}$$

as can be seen by realizing that the behaviour of a bag with singleton value domain is identical to that of a stack over the same domain. Indeed, both recursive definitions yield the transition diagram (or process graph):



According to Theorem 3.2.1, the subprocesses B_n $(n \geqslant 0)$ of B are finitely generated (using (*)) by $+, \cdot, ||, \between, B, a, \underline{a}$. Indeed one easily verifies that

$$B_n = \underline{a}^n\,||\,B.$$

Using (**), the same theorem says for the restricted signature without $||, \between$, that the B_n are finitely generated by $+, \cdot, B, C, a, \underline{a}$. Indeed:

$$B_n = C^n \cdot B.$$

Before considering the case that D is not a singleton, say $D = \{a,b\}$ (the general case follows by a simple argument) and showing that the bag then needs $||, \between$ for its recursive definition, note that (*) can be rewritten as

$$B = (a\underline{a}) \between B.$$

The intuition here is that B is the 'ω-merge' of $a\underline{a}$, i.e.

$$B = a\underline{a} \,||\, a\underline{a} \,||\, a\underline{a} \,||\, a\underline{a} \,||\, \ldots$$

(For trace theory, the ω-merge occurs e.g. in [12].) To be precise:
Let p be a process. Then the ω-merge of p, written as $p^{\underline{\omega}}$, is the limit of the iteration sequence

$$p,\ p\|p,\ p\|p\|p,\ \ldots$$

as defined in [3], where the existence of this limit is shown. It is easy to prove that this limit is also obtained by the guarded fixed point equation

$$X = p \between X.$$

(Note that $X = p\|X$ would not do as it is not guarded and has no unique solution.)

Next consider the bag B over {a,b}, that is:

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

Some alternative definitions are

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

or

$$B = (a\underline{a} + b\underline{b}) \between B,$$

or

$$B = (a\underline{a}\|b\underline{b}) \between B,$$

or

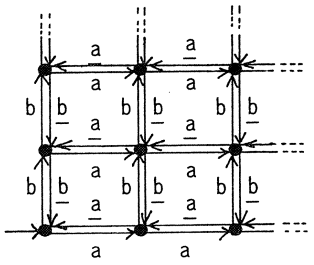
$$\begin{cases} B = X\|Y \\ X = a(\underline{a}\|X) \\ Y = b(\underline{b}\|Y) \end{cases}$$

or

$$\begin{cases} B = X_1\|Y_1 \\ X_1 = aX_2X_1 \\ X_2 = \underline{a} + aX_2X_2 \\ Y_1 = bY_2Y_1 \\ Y_2 = \underline{b} + bY_2Y_2. \end{cases}$$

(The last two systems are guarded after an appropriate substitution.) The last system of equations is of interest since it shows that $R(A^{\infty}(+,\cdot))$ is not closed under $\|$ (after the result below is proved).

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:



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

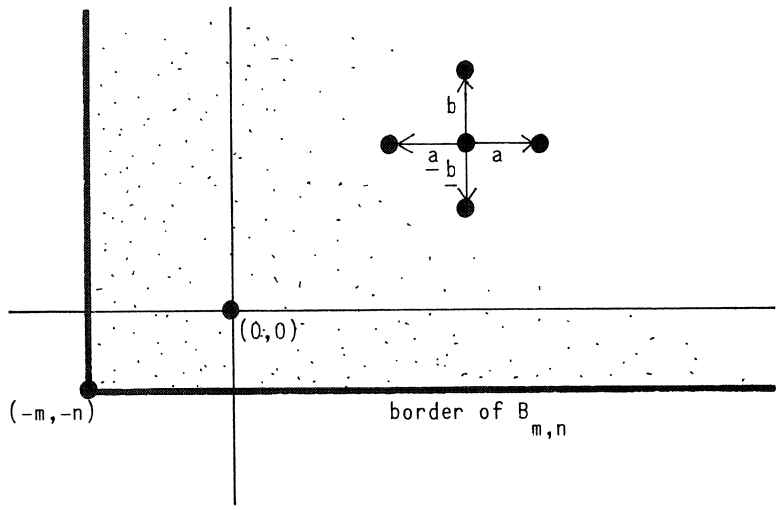
$$B_{m,n} = aB_{m+1,n} + \underline{a}B_{m-1,n} + bB_{m,n+1} + \underline{b}B_{m,n-1}$$

with the understanding that summands in which a negative subscript appears, must vanish.

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

$$B_{m,n} = a^m||b^n||B.$$

Graphically we display the $B_{m,n}$ in the "a,b-plane":



in which the starting node of $B_{m,n}$ is at (0,0) and all traces of $B_{m,n}$ stay confined in the indicated quadrant.

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.

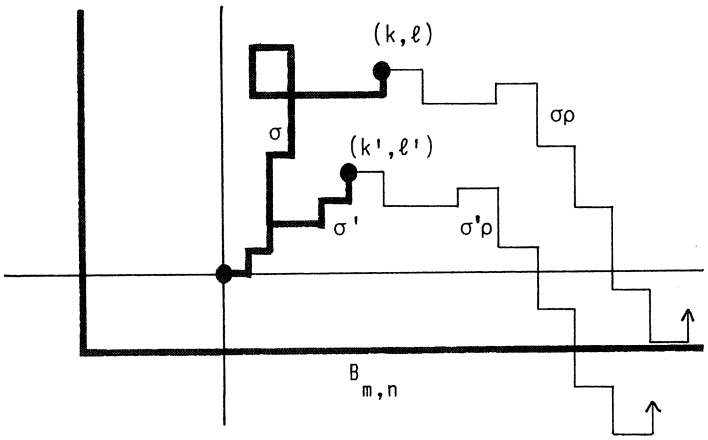
$$\underline{X}_i\underline{P} = B_{k,\ell}$$

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:

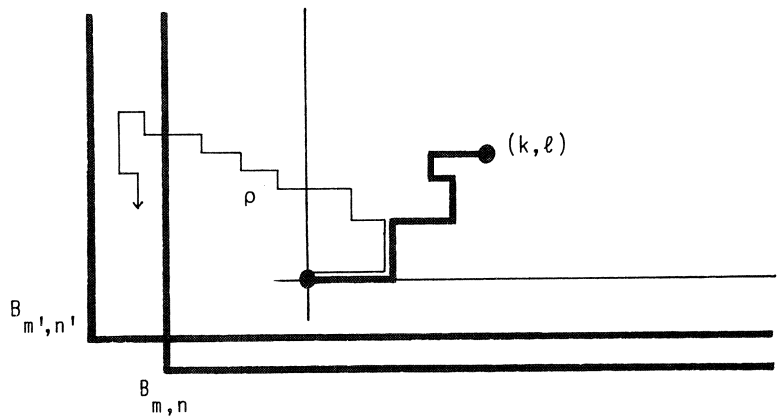
PROPOSITION. *Let* $PQ = B_{m,n}$ *be a factorization of a subprocess of* B. *Suppose* P *is not perpetual. Then:*
(i) all finite traces of P *end in the same point of the* a,b*-plane;*
(ii) P *determines* n,m *and* Q *uniquely (i.e. if moreover* $PQ' = B_{m',n'}$, *then* $Q = Q'$ *and* $n,m = n',m'$*).*

PROOF. (i) Consider the following figure:



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

(ii) To see that $B_{m,n}$ is uniquely determined, let $PQ' = B_{m',n'}$. Say 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, ρ is confined to stay in P as long as it avoids (k,ℓ). But then a trace ρ as in the figure 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.)

A corollary of the preceding Proposition 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} *depending on* i *alone*. 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. □

REMARK. For the case of the 'general' bag defined by

$$B = \sum_{d\in D} d(\underline{d}\,||\,B)$$

where D contains at least two elements, the non-eliminability of $||, \lfloor\!\lfloor$ follows from the above by the following argument. Let ϕ: D → {a,b} be a surjection. Then ϕ extends in the obvious way to a map from $(D \cup \underline{D})^\infty$ to $\{a,\underline{a},b,\underline{b}\}^\infty$, by replacing each atom $u \in D \cup \underline{D}$ by $\phi(u)$. This extended mapping is easily shown to be a homomorphism w.r.t. all operations. Hence a recursive definition without $||, \lfloor\!\lfloor$ for the general bag would by this 'collapsing' mapping yield a similar recursive definition for the bag over $\{a,\underline{a},b,\underline{b}\}$.

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,
Fixed point semantics in process algebras,
Department of Computer Science Technical Report IW 206/82,
Mathematisch Centrum, Amsterdam 1982.

[4] BERGSTRA, J.A. & J.W. KLOP,
Process algebra for communication and mutual exclusion,
Department of Computer Science Technical Report IW 218/83,
Mathematisch Centrum, Amsterdam 1983.

[5] BERGSTRA, J.A. & J.W. KLOP,
A process algebra for the operational semantics of static data flow networks,
Department of Computer Science Technical Report IW 222/83,
Mathematisch Centrum, Amsterdam 1983.

[6] BERGSTRA, J.A. & J.W. KLOP,
An abstraction mechanism for process algebras,
Department of Computer Science Technical Report IW 231/83,
Mathematisch Centrum, Amsterdam 1983.

[7] BERGSTRA, J.A. & J.W. KLOP,
An algebraic specification method for processes over a finite action set,
Department of Computer Science Technical Report IW 232/83,
Mathematisch Centrum, Amsterdam 1983.

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

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

[10] 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).

[11] HOPCROFT, J.E. & J.D. ULLMAN,
Introduction to automata theory, languages and computation,
Addison-Wesley 1979.

[12] ITO, T. & Y. NISHITANI,
On universality of concurrent expressions with synchronization primitives,
TCS 19 (1982), 105-115.

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

ONTVANGEN 3 0 SEP. 1983