W.P. Weijland

Synchrony with empty process

Computer Science/Department of Software Technology Report CS-R8832 August

69D41, 69F12, 69F32, 69F43

Synchrony with empty process

W.P. Weijland
Centre for mathematics and Computer Sciences
Kruislaan 413, 1098 SJ Amsterdam

abstract: In this paper the *Algebra of Synchronous Processes* - or ASP for short (see [13]) - is extended with an extra constant ε which serves as a neutral element for sequential composition. In contrast to the asynchronous algebra ACP, in the setting of ASP the empty process ε seems to behave in a very natural way.

Keywords and phrases: concurrency, process algebra, empty process, synchronous cooperation.
1985 Mathematics subject classification: 68N15, 68Q10, 68Q45, 68Q55.
1987 CR catogories: D.3.1, F.1.2, F.3.2, F.4.3.

1. INTRODUCTION

In many applications of process algebra it seems natural to have a constant which is the neutral element for sequential composition. Firstly, because many of the definitions and axioms can be formulated much more compactly once such a constant is available, and secondly because such a constant would provide us with a termination feature in process algebra.
However, in the case of the *Algebra of Communicating Processes* ACP of BERGSTRA & KLOP [1] the introduction of a neutral element - ε say - for sequential composition implies quite a few complications. For instance in KOYMANS & VRANCKEN [8], where the empty process was first introduced, it destroyed the associativity of the parallel composition operator. This deficiency was repaired by VRANCKEN [12] but after that still many people felt that the result was not satisfactory. Only recently a new version of ACP with ε was presented by BAETEN & VAN GLABBEEK [2], [3] in which a new treatment of ε was to give better insight in its behaviour.
As we will see, in the *Algebra of Synchronous Processes* ASP - introduced in WEIJLAND [13] - it turns out to be much easier to deal with ε than in ACP. It is quite surprising to see that all problems one has to deal with in ACP with ε, simply vanish once working in the setting of ASP. As a consequence, the introduction of ε in ASP proceeds in a straightforward way without many comments or restrictions as is shown in the remaining part of this paper.

2. THE ALGEBRA OF SYNCHRONOUS PROCESSES WITH EMPTY PROCESS

The main references to the theory ASP are BERGSTRA & KLOP [1] and WEIJLAND [13]. In this section we will give a brief overview of ASP and show how to introduce the new constant ε.
ASP is an equational theory with a signature that is much smaller than that of ACP. It contains binary operators +, $\cdot$ and $|$ for alternative, sequential and parallel composition respectively, a constant δ acting as a neutral element for + and as a zero element for $|$ and 1 which is the neutral element for $|$. Terms can be constructed from these operators and a set of constants A.

Report CS-R8832
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

In table 1 below one finds the equations of the theory ASP. The crucial equation, which is the essential difference between ASP and ACP, is the one for $|$ reading:

$$(ax|by) = (a|b)\cdot(x|y)$$

where a,b are constants from A and x and y are variables denoting arbitrary processes. As in regular algebra we will often leave out $\cdot$ and brackets ')' or '(' assuming that $\cdot$ binds stronger than $|$ which in turn binds stronger than +.

We will introduce ε as a new constant acting as a neutral element for $\cdot$, giving the equations:

$$\varepsilon\cdot x = x\cdot\varepsilon = x.$$

Note that $\varepsilon\notin A$ since otherwise we will find an inconsistency with the previous equation for parallel composition. Next the question arises how ε behaves with respect to $|$. To answer this it is important to observe that in ASP we have the equation

$$(ax|b) = (a|b)\cdot x$$

which in the presence of ε has an overlap with the equation for $|$ mentioned earlier (set $y=\varepsilon$). As a consequence the only correct choice for the interaction between ε and $|$ is to make ε act as a neutral element for $|$ as well, obtaining the axioms

$$(\varepsilon|x) = (x|\varepsilon) = x.$$

As in [13] we have $|$ as a commutative and associative operator on the set of constants A. Furthermore, A contains a neutral element 1 for $|$ and a zero element δ. So for all $a\in A$ we have:

$$(\delta|a) = (a|\delta) = \delta.$$
$$(1|a) = (a|1) = a.$$

Note that the second equation is not in conflict with the axioms for ε since $\varepsilon\notin A$.

Finally, recall that for every function f: $A\to A$ the unary operator ρ_f acts on processes as a renaming operator, renaming constants from A into new constants. Since the set A of constants is a parameter of the theory ASP_ε we will often write $ASP_\varepsilon(A)$ instead of just ASP_ε. In table 1 all axioms of ASP_ε are presented together.

$x + y = y + x$	A1	$a\|b = b\|a$	C1
$x + (y + z) = (x + y) + z$	A2	$(a\|b)\|c = a\|(b\|c)$	C2
$x + x = x$	A3	$\delta\|a = \delta$	C3
$(x + y)z = xz + yz$	A4	$1\|a = a$	C4
$(xy)z = x(yz)$	A5		
$x + \delta = x$	A6		
$\delta x = \delta$	A7		
$\varepsilon\cdot x = x$	E1	$(ax\|by) = (a\|b)\cdot(x\|y)$	SC1
$x\cdot\varepsilon = x$	E2	$(x + y)\|z = x\|z + y\|z$	SC2
$\varepsilon\|x = x$	E3	$x\|(y + z) = x\|y + x\|z$	SC3
$\rho_f(\varepsilon) = \varepsilon$	R1	$\rho_f(a) = f(a) \quad (a\neq\delta,1)$	R4
$\rho_f(\delta) = \delta$	R2	$\rho_f(x + y) = \rho_f(x) + \rho_f(y)$	R5
$\rho_f(1) = 1$	R3	$\rho_f(xy) = \rho_f(x)\cdot\rho_f(y)$	R6

Table 1. $ASP_\varepsilon(A)$ $(a,b\in A;\ 1,\delta\in A,\ \varepsilon\notin A)$.

The axioms from table 1 can be transformed into a *term rewriting system* (TRS) by considering them to apply from left to right. Now, if we wish this TRS to be terminating we must exclude the rules for commutativity and associativity of + and $|$. Thus the TRS applies to ASP_ε-terms *modulo* the axioms A1, A2, C1 and C2 and we can think of them as *multisets* of summands and communications.
Apart from the axioms A1, A2, C1 and C2 we have to introduce new rules as well. The reason for this is that when using the axioms from table 1 from left right, applying axioms E1 or E2 we can only eliminate ε's. However, we often use E2 in two directions such as in the term $(a \mid b \cdot y)$ which has to be written as $(a \cdot \varepsilon \mid b \cdot y)$ before we can apply axiom SC1. Obviously, the new rewrite rule corresponds to an equation which is derivable from ASP_ε.
The resulting rewriting system will be called $\mathrm{RASP}_\varepsilon$, see table 2.

$x + x \to x$	RA1	$\delta \mid a \to \delta$	RC1
$(x + y)z \to xz + yz$	RA2	$1 \mid a \to a$	RC2
$(xy)z \to x(yz)$	RA3		
$x + \delta \to x$	RA4		
$\delta x \to \delta$	RA5		
$\varepsilon \cdot x \to x$	RE1	$(a \mid b \cdot y) \to (a \mid b) \cdot y$	RSC1
$x \cdot \varepsilon \to x$	RE2	$(ax \mid by) \to (a \mid b) \cdot (x \mid y)$	RSC2
$\varepsilon \mid x \to x$	RE3	$(x + y) \mid z \to x \mid z + y \mid z$	RSC3
$\rho_f(\delta) \to \delta$	RR1	$\rho_f(a) \to f(a) \quad (a \neq \delta, 1)$	RR4
$\rho_f(\varepsilon) \to \varepsilon$	RR2	$\rho_f(x + y) \to \rho_f(x) + \rho_f(y)$	RR5
$\rho_f(1) \to 1$	RR3	$\rho_f(xy) \to \rho_f(x) \cdot \rho_f(y)$	RR6

Table 2. $\mathrm{RASP}_\varepsilon(A)$ $(a, b \in A;\ 1, \delta \in A,\ \varepsilon \notin A)$.

Note that we want all $\mathrm{RASP}_\varepsilon$-reductions to correspond to ASP_ε-deductions. Every application of a $\mathrm{RASP}_\varepsilon$-rule corresponds to a derivation step in ASP_ε in an obvious manner, except that $\mathrm{RASP}_\varepsilon$ works on closed ASP_ε-terms *modulo* commutativity and associativity of + and $|$. In the case of $|$ these properties cannot be derived from ASP_ε for arbitrary processes, but it follows from an easy induction proof that they hold for all closed terms:

PROPOSITION 2.1 *For all closed* ASP_ε*-terms* s, t *and* u *we have:*
$\mathrm{ASP}_\varepsilon \vdash s \mid t = t \mid s$ *and* $\mathrm{ASP}_\varepsilon \vdash (s \mid t) \mid u = s \mid (t \mid u)$.

So $\mathrm{RASP}_\varepsilon$-reductions correspond to ASP-deduction steps. Now we will consider an important proposition that will be crucial in later proofs (see also [2] and [3]).

DEFINITION 2.1 The set of *basic terms* BT is recursively defined as follows:

1. $\varepsilon \in$ BT
2. if $a \in A \cup \{\delta\}$ and $x \in$ BT then $ax \in$ BT
3. if $x,y \in$ BT then $(x + y) \in$ BT.

PROPOSITION 2.2

1. RASP$_\varepsilon$ *is strongly terminating.*
2. If an ASP$_\varepsilon$*-term* t *is in normal form then it is a basic term.*

Proposition 2.2 (1) can be proved by using Kruskals termination theorem for trees or by using structural induction on ASP$_\varepsilon$-terms. It says that the term rewriting system RASP$_\varepsilon$ has no infinite reduction sequences. Proposition 2.2 (2) is proved directly by using structural induction on ASP$_\varepsilon$-terms that are *not* a basic term. It turns out that every non-basic term is an instance of a lefthand side of some rule in RASP$_\varepsilon$ and thus is not in normal form.

THEOREM 2.3 (elimination)

For every ASP$_\varepsilon$*-term* t *there exists a basic term* s *such that* ASP$_\varepsilon \vdash$ t=s.

PROOF Every ASP$_\varepsilon$-term t can be rewritten into a normal form s - using rewrite rules from RASP$_\varepsilon$ - which is a basic term (see proposition 2.2). Every application of a rewrite rule from RASP$_\varepsilon$ corresponds to a proof step in ASP$_\varepsilon$ and so for some basic term s we have ASP$_\varepsilon \vdash$ t=s. □

PROPOSITION 2.4 RASP$_\varepsilon$ *is confluent.*

PROOF For any rule r let us write lhs(r) for its lefthand side. Now recall that if r_1 and r_2 are rewrite rules such that lhs(r_1) unifies with a non-variable subterm of lhs(r_2), then lhs(r_2) can be reduced in two ways: by application of either r_1 or r_2. The pair of reducts resulting from every such pair of rules is called a *critical pair* (see [6] and [7]).

Obviously, a TRS is confluent if and only if every critical pair (s,t) is joinable, i.e. if there exist zero or more step reductions to a reduct that both terms in the pair have in common. The easiest way to prove RASP$_\varepsilon$ to be confluent is by simply checking all 29 critical pairs that emerge from the system. All 'critical pairs of rules' are listed below.

Recall that we considered RASP$_\varepsilon$ to act on ASP$_\varepsilon$-terms *modulo* A1, A2, C1 and C2. As a consequence lhs(RC1) and lhs(RC2) are unifiable terms although the arguments seem to appear in the wrong order. In the listing below we write R: A,B,C,.. to indicate that lhs(R) unifies with a proper subterm of lhs(A), lhs(B), lhs(C),... . So, from every pair of rules (R,A), (R,B), (R,C),... we can easily find a critical pair.

RA1:	RA2, RA4, RSC3;	RC1:	RC2;
RA2:	RA3, RE2, RR6;	RC2:	RC1;
RA3:	RA3, RE2, RR6;	RE1:	RA3, RE2, RR6;
RA4:	RA1, RA2, RA4, RR5, RSC3;	RE2:	RA2, RA3, RA5, RR6;
RA5:	RA3, RE2, RR6, RSC1, RSC2;	RE3:	RSC3;

The rules RSC1-RSC3, RR1-RR6 do not give rise to any critical pair. It is left to the reader to check that all critical pairs actually are joinable. To give an example: from RA2 and RA3 we find the 'critical term' $t = ((x + y)z)u$ (lhs(RA2) unifies with (xy) in lhs(RA3)!). Obviously, t has two reducts, $(xz + yz)u$ and $(x + y)(zu)$. Applying RA2 to the first term we obtain $(xz)u + (yz)u$ and by RA3 this becomes $x(zu) + y(zu)$. The same term is found by directly applying RA2 to the second term and so both terms have a common reduct: the critical pair is joinable. □

From proposition 2.4 we obtain an important theorem. Let us write BPA_ε for the theory consisting of the axioms A1-A7, E1, E2. So, BPA_ε is ASP_ε without parallel composition and without renaming (BPA stands for *Basic process algebra*). Observe that every *basic term* (see definition 2.1) is a term from the theory BPA_ε and so it follows from the elimination theorem that every ASP_ε-term can be proved equal to a term from BPA_ε. The question is now: does ASP_ε introduce new equalities on BPA_ε-terms? The answer is no, as follows from the following two theorems:

THEOREM 2.5 *Let* $RBPA_\varepsilon$ *be the TRS consisting of the rules* RA1-RA5, RE1 *and* RE2. *Then* RBPA *is confluent and terminating and every normal form with respect to* $RBPA_\varepsilon$ *is in normal form with respect to* $RASP_\varepsilon$ *as well.*

PROOF Obviously $RBPA_\varepsilon$ is terminating since $RASP_\varepsilon$ is. Moreover, observe that $RBPA_\varepsilon$ acts on BPA_ε-terms only (and produces BPA_ε-terms), whereas all other rules of $RASP_\varepsilon$ act on ASP_ε-terms with $|$ or ρ_f. It follows from this fact that every reduction of a BPA_ε-term in $RASP_\varepsilon$ is a reduction in $RBPA_\varepsilon$. Hence $RBPA_\varepsilon$ is confluent - since $RASP_\varepsilon$ is - and all its normal forms are $RASP_\varepsilon$-normal forms. □

THEOREM 2.6 ASP_ε *is a conservative extension of* BPA_ε.

PROOF Assume that $ASP_\varepsilon \vdash s = t$ for two BPA_ε-terms s and t. Then there exists a proof in ASP_ε consisting of equations $s=u_1$, $u_i=u_{i+1}$, $u_k=t$ $(0<i<k)$ that are closed contexts of instances of - possibly reversed - rules from ASP_ε. So, if u=v is a closed instance of a rule in ASP_ε and C[ ] is a closed context (a term with a 'hole' which has no variables) then C[u] = C[v] is a closed context of an instance of a rule from ASP_ε. Since the rules in ASP_ε have no direction, we allow u=v to be a reversed rule.
From proposition 2.2 it follows that every u_i has a unique normal form and since $RASP_\varepsilon$ is confluent (proposition 2.4) all u_i have the same normal form. Since s and t are BPA_ε-terms they have the same normal form with respect to $RBPA_\varepsilon$ and from the reductions in $RBPA_\varepsilon$ we can easily construct a proof of s = t within the theory BPA_ε. Hence $BPA_\varepsilon \vdash s=t$. □

The reason why theorem 2.6 is a crucial theorem in this paper, is because of the fact that in the presence of axiom E3 such a result cannot be obtained in the asynchronous algebra ACP (see BAETEN & VAN GLABBEEK [2]).
To see this, assume we have BPA_ε together with the following axioms:

$x \| y = x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x + x \mid y$	EM1		
$(x \| y) \| z = x \| (y \| z)$	EM2	$x \mid y = y \mid x$	EM6
$\varepsilon \mathbin{\lfloor\!\lfloor} x = \delta$	EM3	$x \mid \varepsilon = x$	EM7
$ax \mathbin{\lfloor\!\lfloor} y = a(x \| y)$	EM4	$x \mid ay = (x \mid a)y$	EM8
$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$	EM5	$x \mid (y + z) = x \mid y + x \mid z$	EM9

The presentation of these nine axioms are similar to the ones in [2] apart from EM1, EM2 and most of all EM7, which in [2] reads as $x \mid \varepsilon = \delta$. We will show that these axioms do not yield a conservative extension of BPA_ε by giving a counterexample which is a variant on one found by Karst Koymans and Rob van Glabbeek.
Consider the term $c \| (a + \varepsilon) \| b$, where a, b and c are constants from A not communicating with any constant. Then, using the axioms above, one easily finds:
$c \| (a + \varepsilon) = ac + c + c\cdot(a + \varepsilon)$
$(a + \varepsilon) \| b = ab + b + b\cdot(a + \varepsilon)$
and so we find two divergent reductions of $c \| (a + \varepsilon) \| b$ as follows:
$\underline{c \| (a + \varepsilon)} \| b \Rightarrow (ac + c + c\cdot(a + \varepsilon)) \| b =$
$= a\cdot(c \| b) + cb + c\cdot((a + \varepsilon) \| b) + b\cdot((a + \varepsilon) \| c)$
$= a\cdot(cb + bc) + cb + c\cdot(ab + b + b\cdot(a + \varepsilon)) + b\cdot(ac + c + c\cdot(a + \varepsilon))$
$c \| \underline{(a + \varepsilon) \| b} \Rightarrow c \| (ab + b + b\cdot(a + \varepsilon)) =$
$= a\cdot(bc + cb) + bc + b\cdot(ac + c + c\cdot(a + \varepsilon)) + c\cdot(ab + b + b\cdot(a + \varepsilon))$.
So, both expressions can be proved equal in the theory BPA_ε + EM1-EM9 but not in BPA_ε since the first contains a summand cb which does not appear in the second. Hence BPA_ε + EM1-EM9 is not a conservative extension of BPA_ε.
Apparently, the term rewriting system that one may associate with BPA_ε + EM1-EM9 is not confluent since we found two reducts of $c \| (a + \varepsilon) \| b$ that are not joinable. From theorem 2.6 it follows, however, that in ASP_ε we do not have this problem and we can simply add axiom E3 to our axiom system.

3. THE TRANSITION MODEL

In this section we will study on a model - the *transition model* - for ASP_ε providing us with a clear operational semantics for ASP_ε. It turns out that the transition model can be constructed from the axioms of ASP_ε in a very natural way. Its construction first appeared in MILNER [10] for CCS, and in VAN GLABBEEK [5] for ACP.

a :	$a \xrightarrow{a} \varepsilon$	$\varepsilon\downarrow$		
+ :	$\dfrac{x \xrightarrow{a} x'}{(x + y) \xrightarrow{a} x'}$	$\dfrac{x\downarrow}{(x + y)\downarrow}$	$\dfrac{y \xrightarrow{a} y'}{(x + y) \xrightarrow{a} y'}$	$\dfrac{y\downarrow}{(x + y)\downarrow}$
· :	$\dfrac{x \xrightarrow{a} x'}{x\cdot y \xrightarrow{a} x'\cdot y}$	$\dfrac{x\downarrow,\ y \xrightarrow{a} y'}{x\cdot y \xrightarrow{a} y'}$	$\dfrac{x\downarrow,\ y\downarrow}{x\cdot y\downarrow}$	

Table 3. The transition predicates $\xrightarrow{a}$ and $\downarrow$ on BPA_ε-terms ($a \in A - \{\delta\}$).

On BPA_ε-terms, for each $a \in A-\{\delta\}$ we define binary predicates $\rightarrow^a$ and a unary predicate $\downarrow$. Intuitively, $x \rightarrow^a y$ means that process x can evolve into process y by executing a. $x\downarrow$ means that process x can terminate immediately. In table 3 the proof rules for these two predicates are presented. From now on we assume $\rightarrow^a$ and $\downarrow$ to be the minimal predicates that are closed under derivations from table 3.

DEFINITION 3.1 A *bisimulation* is a binary reflexive relation R on BPA_ε-terms with the following properties ($a \in A-\{\delta\}$):

1. If R(p,q) and $p \rightarrow^a p'$, then there exists q' such that $q \rightarrow^a q'$ and R(p',q')
2. If R(p,q) and $q \rightarrow^a q'$, then there exists p' such that $p \rightarrow^a p'$ and R(p',q')
3. If R(p,q), then $p\downarrow$ if and only if $q\downarrow$.

If there exists a bisimulation R between processes p and q, then we say p and q are *bisimilar*, notation: $p \leftrightarroweq q$.

THEOREM 3.1 $\leftrightarroweq$ *is a congruence relation on* BPA_ε*-terms.*

The proof of theorem 3.1 is left to the reader. Recall that a relation is called a *congruence* if it is an equivalence relation which respects function symbols.

DEFINITION 3.2 The *transition model* $\mathbb{T}$ is the set of closed BPA_ε-terms modulo $\leftrightarroweq$.

DEFINITION 3.3 The *depth* dp(t) of a basic term t is defined inductively as follows:

1. $dp(\varepsilon) = 0$, $dp(\delta) = 1$
2. for all $a \in A-\{\delta\}$ and basic terms s: $dp(a \cdot s) = 1 + dp(s)$
3. for any two basic terms s and t: $dp(s + t) = max(dp(t), dp(s))$.

PROPOSITION 3.2 *Let* t *be a basic term and* $a \in A-\{\delta\}$. *Then the following statements hold:*

1. If $t \rightarrow^a s$, *then* s *is a basic term and* $dp(s)<dp(t)$;

2. If $t \rightarrow^a s$, *then* $BPA_\varepsilon \vdash t = as + t$ *(i.e.* $a \cdot s$ *is a summand of* t*);*

3. If $t\downarrow$, *then* $BPA_\varepsilon \vdash t = \varepsilon + t$.

The proof of proposition 3.2 follows easily by structural induction on t and is left to the reader. The following theorem is an important result about the transition predicates of table 3.

THEOREM 3.3 *For all closed* BPA_ε*-terms* s *and* t *we have:* $BPA_\varepsilon \vdash s=t \Rightarrow s \leftrightarroweq t$.

PROOF We only need to prove that $\leftrightarroweq$ respects all axioms of BPA_ε. For instance consider axiom (A1) $(s + t) \leftrightarroweq (t + s)$. Set $R = I \cup \{(s + t, t + s)\} \cup \{(t + s, s + t)\}$, where I is the binary identity relation. Assume we have R(p,q) then we find that either p and q are identical or $p = (s + t)$ and $q = (t + s)$. Now suppose $(s + t) \rightarrow^a u$ then this transition is an instance of one of the +-rules in table 3. Therefore it follows, that either $s \rightarrow^a u$ or $t \rightarrow^a u$ and so $(t + s) \rightarrow^a u$ (applying the +-rule

again) and by definition we have R(u,u). In the same way it follows from $(s + t)\downarrow$ that $(t + s)\downarrow$. Hence we find that R is a bisimulation between (s + t) and (t + s).
In the same way we find that (A2) $((s + t) + u) \leftrightarrow (s + (t + u))$ and (A3) $(s + s) \leftrightarrow s$. In order to prove (A4) $(s + t)u \leftrightarrow (su + tu)$ set $R = I \cup \{((s + t)u, su + tu)\} \cup \{(su + tu, (s + t)u)\}$, then it easily follows that R is a bisimulation between (s + t)u and (su + tu). Similarly we can prove (A5) $(st)u \leftrightarrow s(tu)$. Note that δ nor $\delta \cdot x$ can be the left hand side of any transition, and therefore we have (A6) $s + \delta \leftrightarrow s$ and (A7) $\delta s \leftrightarrow \delta$.
Finally, from the $\cdot$-rules in table 3 we find immediately that $\varepsilon \cdot x$, $x \cdot \varepsilon$ and x all have the same transitions and so: (E1) $\varepsilon \cdot x \leftrightarrow x$ and (E2) $x \cdot \varepsilon \leftrightarrow x$. □

The converse of 3.3 holds as well:

THEOREM 3.4 $\mathbb{T}$ *is an initial algebra for* BPA_ε.

PROOF So we have to prove $BPA_\varepsilon \vdash s=t \Leftrightarrow s \leftrightarrow t$. By proposition 3.2 it is sufficient to prove this for basic terms s and t only (using transitivity of $\leftrightarrow$).
$\Rightarrow$ by theorem 3.3.
$\Leftarrow$ This is done by induction on dp(s) + dp(t), as follows.
If dp(s) + dp(t) = 0 it directly follows that both s,t are sums of ε's, and so we have both $s \leftrightarrow t$ and $BPA_\varepsilon \vdash s=t$. Now assume $s \leftrightarrow t$ for BPA_ε-terms s and t and for all s', t' dp(s') + dp(t') < dp(s) + dp(t) with $s' \leftrightarrow t'$ it is already proved that $BPA_\varepsilon \vdash s'=t'$.
It is enough to prove that any summand ε or $a \cdot s'$ of s is also a summand of t (and vice versa) since then it follows that both $BPA_\varepsilon \vdash s = s + t$ and $BPA_\varepsilon \vdash t = t + s$, from which it follows that $BPA_\varepsilon \vdash s=t$.
(1) Assume ε is a summand of s, then either $s \equiv \varepsilon + r$ or $s \equiv \varepsilon$. Clearly $s\downarrow$ and hence $t\downarrow$ since $s \leftrightarrow t$, and therefore by proposition 3.2 it follows that ε is a summand of t.
(2) Assume $a \cdot s'$ is a summand of s, i.e. $s \equiv a \cdot s' + r$ or $s \equiv a \cdot s'$. Then $s \rightarrow^a s'$, and so $t \rightarrow^a t'$ for some t' with $s' \leftrightarrow t'$. By proposition 3.3 it follows that t' is a basic term with dp(t')<dp(t) and at' is a summand of t, i.e. $BPA_\varepsilon \vdash t = at' + t$. Furthermore, dp(s')<dp(s) so by induction we conclude that $BPA_\varepsilon \vdash s'=t'$. Hence $BPA_\varepsilon \vdash at'=as'$ and therefore $BPA_\varepsilon \vdash t = as' + t$. □

Theorem 3.4 makes clear that BPA_ε is in fact a full axiomatisation of bisimulation equivalence on closed BPA_ε-terms. In order to extend BPA_ε to the larger theory ASP_ε, consider the additional rules in table 4:

$$| : \quad \frac{x \rightarrow^a x',\ y \rightarrow^b y'}{x|y \rightarrow^{a|b} x'|y'} \qquad \frac{x \rightarrow^a x',\ y\downarrow}{x|y \rightarrow^a x'} \qquad (a|b \neq \delta)$$

$$\frac{x\downarrow,\ y \rightarrow^a y'}{x|y \rightarrow^a y'} \qquad \frac{x\downarrow, y\downarrow}{x|y\downarrow}$$

$$\rho_f : \quad \frac{x \rightarrow^a x'}{\rho_f(x) \rightarrow^{f(a)} \rho_f(x')} \qquad \frac{x \rightarrow^1 x'}{\rho_f(x) \rightarrow^1 \rho_f(x')} \qquad \frac{x\downarrow}{\rho_f(x)\downarrow} \qquad (a \neq \delta, 1;\ f(a) \neq \delta)$$

Table 4. The transition predicates $\rightarrow^a$ and $\downarrow$ on ASP_ε-terms ($a \in A - \{\delta\}$).

Again, we will assume these transition predicates to be the minimal interpretation which is closed under the rules of table 3 and table 4.

DEFINITION 3.4 The *transition model with communication* $\mathbb{TC}$ is the set of ASP_ε-terms modulo bisimulation equivalence.

THEOREM 3.5 $\mathbb{TC}$ is an initial algebra for ASP_ε.

PROOF It is straightforward to prove that $ASP_\varepsilon \vdash s=t \Rightarrow s \leftrightarrow t$ (*).
So assume $s \leftrightarrow t$ for some ASP_ε-terms s and t. By theorem 2.3 it follows that for some BPA_ε-terms s' and t' we have that $ASP_\varepsilon \vdash s=s'$ and $ASP_\varepsilon \vdash t=t'$. Now using (*) it follows that $s \leftrightarrow s'$ and $t \leftrightarrow t'$, so using the transitivity of $\leftrightarrow$ we find that $s' \leftrightarrow t'$. Since both s' and t' are BPA_ε-terms, it then follows from theorem 3.4 that $BPA_\varepsilon \vdash s'=t'$, hence $ASP_\varepsilon \vdash s'=t'$.
From $ASP_\varepsilon \vdash s=s'$ and $ASP_\varepsilon \vdash t=t'$ it then follows that $ASP_\varepsilon \vdash s=t$. □

4. RECURSION

The transition model, introduced in the previous section, works on closed ASP_ε-terms. In practice, however, we will often need methods to specify infinite processes as well. In the following we will consider the construction of recursion which enables us to describe infinite processes algebraically.

DEFINITION 4.1 A *recursive specification* over ASP_ε is a set of equations $E = \{x = t_x : x \in V\}$, where V is a set of variables and t_x are ASP_ε-terms only containing variables from V.

DEFINITION 4.2 Let t be an ASP_ε-term and x a variable from t. The occurrence of x in t is called *guarded* if x is preceded by an atomic action from A, i.e. if t has a subterm of the form $a \cdot s$ with $a \in A$, and this x occurs in s. If not, the occurrence of x is called *unguarded.*
A recursive specification is called guarded if each occurrence of a variable is guarded.

Note that $\varepsilon \notin A$ and therefore it follows from definition 4.2 that ε cannot serve as a guard in a recursive specification.

DEFINITION 4.3 The *Recursive Definition Principle* (RDP) is the rule saying that every guarded recursive specification has a solution.

We will write $\mathcal{A} \models$ RDP if the recursive definition principle RDP holds in the algebra $\mathcal{A}$.
Recursive specifications are used to specify processes. If a recursive specification E is satisfied in a model and $x \in V$, then $<x \mid E>$ will denote the x-component of some solution of E. So if E has more than just one solution, $<x \mid E>$ will denote some kind of quantified variable ranging over all E's witnesses (see VAN GLABBEEK [5]). If E has no solution, then $<x \mid E>$ remains undefined. Finally,

$\langle t \mid E\rangle$ denotes the term t in which each occurrence of a variable $x \in V$ is replaced by $\langle x \mid E\rangle$. The fact that $\langle x \mid E\rangle$ is a solution of E can simply be expressed by: $\langle x \mid E\rangle = \langle t_x \mid E\rangle$.

$$\text{recursion:}\qquad \frac{\langle t_x \mid E\rangle \longrightarrow^{a} y}{\langle x \mid E\rangle \longrightarrow^{a} y} \qquad \frac{\langle t_x \mid E\rangle\downarrow}{\langle x \mid E\rangle\downarrow}$$

Table 5. Additional transitions for recursion.

In table 5 we find the transitions which hold for the new terms $\langle t_x \mid E\rangle$ for every recursive specification E in ASP_ε. Note that we allow E to be infinite. Restricting to finite specifications only will give us a smaller model.

DEFINITION 4.4 The *Recursive Specification Principle* (**RSP**) says that every guarded recursive specification has at most one solution.

So, in a model with both **RDP** and **RSP** every guarded recursive specification has precisely one solution. Observe that ε cannot serve as a guard because of the fact that in any model for ASP_ε the equation $x = \varepsilon \cdot x$ has every process x as a solution.
Now suppose $\mathbb{TC}_R$ is the extension of $\mathbb{TC}$ with new terms $\langle t_x \mid E\rangle$ for every recursive specification E and every variable x in E, satisfying the transitions of table 5.

PROPOSITION 4.1 $\mathbb{TC}_R \models$ **RDP, RSP**

The proof of proposition 4.1 is left to the reader.
Clearly $\mathbb{TC} \not\models$ **RDP** since no closed term has infinitely many transitions, whereas a process satisfying $x = a \cdot x$ can do infinitely many a-steps.
It is important to observe that there exists a canonical construction which turns a transition model like $\mathbb{T}$, $\mathbb{TC}$ or $\mathbb{TC}_R$ into a model with *process graphs* as its domain (see PARK [11], MILNER [9] and BAETEN, BERGSTRA & KLOP [4]). Because of the close relationship between the two kinds of models in this paper it seemed sufficient to present the transition model only.

REMARK: Since in [13] one can find an example of how to use ASP in practical applications, we chose not to include such an example in this paper. The concept of using *atomic vectors* (i.e. vectors of atomic actions, the set of which can be denoted by $A^{\mathcal{P}}$ for any set of *ports* $\mathcal{P}$) proceeds in a similar way for ASP_ε by simply working within $ASP_\varepsilon(A^{\mathcal{P}})$. Note that vectors in $A^{\mathcal{P}}$ do not contain ε's in their components. So, apart from a general constant ε (which informally can be interpreted as $(\varepsilon\, \varepsilon \cdots \varepsilon)$), the signature of $ASP_\varepsilon(A^{\mathcal{P}})$ is without ε's.

REFERENCES

[1] J.A.BERGSTRA & J.W.KLOP, *Process algebra for synchronous communcation*, Information & Control 60 (1/3), pp.109-137, 1984.

[2] J.C.M.BAETEN & R.J.VAN GLABBEEK, *Merge and termination in process algebra*, Proc. 7th Conf. on FST&TCS, Pune, India (K.V.Nori, ed.), Springer LNCS 287, pp. 153-172, 1987.

[3] J.C.M.BAETEN & VAN GLABBEEK, *Abstraction and empty process in process algebra*, report CS-R8721, Centre for Mathematics and Computer Sciences, Amsterdam, 1987; to appear in: Fundamenta Informaticae.

[4] J.C.M.BAETEN, J.A.BERGSTRA & J.W.KLOP, *On the consistency of Koomen's fair abstraction rule*, TCS 51 (1/2), pp.129-176, 1987.

[5] R.J.VAN GLABBEEK, *Bounded nondeterminism and the approximation induction principle in process algebra*, proc. STACS (F.J.Brandenburg, G.Vidal-Naquet & M.Wirsing eds.), Springer LNCS 247, pp.336-347, 1987.

[6] D.E.KNUTH & P.B.BENDIX, *Simple word problems in universal algebras*, in: Computational Problems in Abstract Algebra (ed. J.Leech), Pergamon Press, pp.263-297, 1970

[7] J.W.KLOP, *Term rewriting systems: a tutorial*, Bulletin of the EATCS, pp.143-182, 1987.

[8] C.J.P.KOYMANS & J.L.M.VRANCKEN, *Extending process algebra with the empty process ε*, report LGPS 1, Dept. of Philosophy, State University of Utrecht, The Netherlands, 1985.

[9] R.MILNER, *A calculus for communicating processes*, Springer LNCS 92, 1980.

[10] R.MILNER, *Lectures on a calculus for communicating systems*, conf. proc. Seminar on Concurrency (Brookes, Roscoe & Winskel eds.), LNCS 197, Springer Verlag, pp.197-220, 1985.

[11] D.M.R.PARK, *Concurrency and automata on infinite sequences*, proc. 5th GI conference, Springer LNCS 104, 1981.

[12] J.L.M.VRANCKEN, *The algebra of communicating processes with empty process*, report FVI 86-01, Dept. of Computer Science, University of Amsterdam, 1986.

[13] W.P.WEIJLAND, *The algebra of synchronous processes*, report CS-R8807, Centre for Mathematics and Computer Sciences, Amsterdam, 1988; to appear in Fundamenta Informaticae.