On Sequential Composition, Action Prefixes and Process Prefix

J.C.M. Baeten
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, The Netherlands

J.A. Bergstra
Programming Research Group, University of Amsterdam, P.O.Box 41882, 1009 DB Amsterdam, The Netherlands
and
Department of Philosophy, Utrecht University, Heidelberglaan 2, 3584 CS Utrecht, The Netherlands

We illustrate the difference between sequential composition in process algebra axiomatisations like ACP and action prefixing in process calculi like CCS. We define both early and late input in a general framework extending ACP, and consider various subalgebras, some very close to value passing CCS, another one close to CSP.

1980 Mathematics Subject Classification (1985 revision): 68Q45, 68Q55, 68Q65, 68Q50.
1987 CR Categories: F.4.3, D.2.10, D.3.1, D.3.3.
Key words & Phrases: process algebra, process calculus, sequential composition, action prefix, process prefix, CCS, ACP.
Note: Partial support received by ESPRIT basic research action 7166, CONCUR2. The second author also received partial support from ESPRIT basic research action 6454, CONFER.

1. INTRODUCTION.

The purpose of this paper is to present a family of axiom systems that illustrates the difference between sequential composition and action prefixing. In ACP [BK84], sequential composition is taken as a basic operation whereas CCS [MIL80] is based on action prefixing. CSP [HOA78] in turn uses sequential composition (being a programming language rather than a process algebra description).

We will survey a family of specifications starting with a parameter-free form of ACP. This specification is subsequently extended with a finite sort N of data and P^{N} of mappings from N to processes. The setting has been simplified by the assumption that a finite set of data is used for value matching and value passing. Two process constructors are introduced: late functional prefix and early functional prefix. Axioms are provided for these sorts and functions. Then, by a special choice of atomic actions, a specification is given of communication by means of value matching. This is the usual format used in ACP-based case studies, starting from [BK86]. Next, the early and late input prefixes $\mathsf{er}_{\mathsf{m}}(\mathsf{v});\_$ resp. $\mathsf{lr}_{\mathsf{m}}(\mathsf{v});\_$ are defined. These are action prefixes because of the variable binding effect. We notice that if $\cdot$ denotes associative sequential composition (as it is present in ACP), then $\mathsf{er}_{\mathsf{m}}(\mathsf{v})\cdot\mathsf{P}\cdot\mathsf{Q}$ cannot bind v in $\mathsf{P}\cdot\mathsf{Q}$ because of the scoping ambiguity (($\mathsf{er}_{\mathsf{m}}(\mathsf{v})\cdot\mathsf{P})\cdot\mathsf{Q}$ vs. $\mathsf{er}_{\mathsf{m}}(\mathsf{v})\cdot(\mathsf{P}\cdot\mathsf{Q})$). We conclude that binding actions must be prefixes.

We concentrate on finite processes, restricting attention tot a single finite datatype. This allows us to provide initial algebra specifications of various process algebras. Infinite processes can be introduced in the setting of initial algebra semantics along the lines of [BT84].

Having presented an extension of ACP, we determine four algebras of reducts of its initial algebra. Of each of these, a direct specification is given. By direct specification, we mean a specification not involving additional constants, operators and sorts. Two of these specifications, VMC and VPC describe different models of a language very close to finitary CCS.

Taking the minimal subalgebra of a reduct of an initial algebra is a known specification method. It occurs as a type II initial algebra specification in [BT87] and as the subalgebra interpretation of algebraic specifications in [KAM79]. We will refer to such specifications as subalgebra of reduced model specifications (abbreviated as SRM specifications).

It is our view that these specifications clarify the relationship between ACP and CCS in a quite satisfactory way. Subsequently, action prefix is extended to process prefix. As a consequence, a concise formulation of so-called parallel input is found. As a second application of process prefixing, we introduce exit actions and an iteration construct that allows the explicit solution of some recursion equations.

The contribution of this paper is certainly not a semantic one. We use strong early bisimulation semantics and strong late bisimulation semantics that are well-known from [PAR81] and the CCS literature [MPW91], [HL93]. We notice that strong (early) bisimulation was around already before 1980 in the literature on modal logics under different names. We hope to contribute to the design of algebraic specifications of process algebras. In particular, it is shown how action and process prefixing are very useful on top of ACP, in spite of the fact that ACP is based on the seemingly more general sequential composition mechanism.

Several remarks on notation are useful. In [BB90], a constant 0 has been introduced that satisfies the axioms $0 \cdot X = 0$, $X \cdot 0 = 0$ and $X + 0 = 0$. This constant can be identified with the constant 0 of [MIL89]. By doing so, an important difference between action prefixing $a;X$ and $a \cdot X$ (sequential composition of a and X) emerges:

$$a \cdot 0 = 0 \qquad \text{whereas} \qquad a;0 = a \cdot \delta.$$

Thus, sequential composition is not an extension of prefixing, not even in the case of basic actions, i.e. actions without binding effect. However, we will simplify our discussion by not considering 0. The role of CCS's nil will be played by δ of ACP. As said above, we will use ; to denote action prefixing rather than . as used in [MIL80]. Our motivation for this choice is as follows: ACP's · extends CCS's . in the setting of basic actions and in the absence of 0. That justifies its notation. Then, another notation is needed for action prefixing for which we choose ; like in LOTOS [BRI88].

Our value passing axioms VPA seem to be consistent with points of view put forward in [HI91], [HL93]. A difference is our emphasis on purely equational specification.

We have also considered CSP's communication primitives $m!t$ and $m?x$ (see [HOA78]). These require the presence of a global state from which a value for t is retrieved and into which a value for x is stored (like the register operator of [VER92]). When modeled as atomic actions, $m!t$ and $m?x$ have no variable binding effect and they can be used in the context of an associative sequential composition operator.

A distinction can be made between a process algebra specification and a process calculus specification. ACP is a process algebra specification, it explains the laws for algebraic manipulation of processes. VPC (value passing calculus) is the closest approximation we find of finite CCS with value passing. VPC is a process calculus, because it makes essential use of bound and free variables. Below, we will obtain a model of VPC as a subalgebra of a reduct of the initial algebra of a process algebra specification extending ACP. It is not clear to us whether or not such an approach is possible for mobile communication in the style of [EN86] and [MPW92] as well.

ACKNOWLEDGEMENT: We thank A. Ponse (University of Amsterdam), C. Verhoef (Eindhoven University of Technology) and the referee for their helpful remarks.

2. PARAMETER-FREE ACP AND FUNCTIONAL PREFIXES.

We consider two forms of functional prefix. Later on, these will be used to model early and late input. First, we give a version of ACP without parameters.

2.1 ACP WITHOUT PARAMETERS.

We describe a version of the process algebra ACP (Algebra of Communicating Processes) of [BK84] without parameters. The axioms are in table 1, the signature follows. These axioms have the form of a first order theory. For instance, axiom C1 should be read as

$$\forall a \in A\, \forall b \in A\, (a \mid b = b \mid a).$$

The signature of ACP is as follows:

Sorts:

P	sort of processes
$A \subseteq P$	subsort of atomic actions. Introduced in [BK82] (adapted from [BZ82]).
$A^c \subseteq A$	subsort of core atomic actions.

Constant:

$\delta \in A$	inaction (deadlock). From [BK84].

Functions:

$+ : P \times P \to P$	alternative composition, sum. From [MIL80].
$\cdot : P \times P \to P$	sequential composition, product. From [BZ82]. The laws A4, A5 are from [BK82], [BK84].
$\parallel : P \times P \to P$	parallel composition, merge. Laws CM1-9 from [BK84].
$\mathbin{\|\!\!\lfloor} : P \times P \to P$	left-merge. From [BK82].
$\mid : P \times P \to P$	communication merge. From [BK84].

$\partial: A^c \times P \to P$	encapsulation operator. We write $\partial_{\{d\}}(X)$ for $\partial(d, X)$. Notation from [BK84].

Variables:

$X,Y,Z,... \in P$	process variables
$a,b,c,... \in A$	action variables
$d,e,... \in A^c$	core action variables.

$X + Y = Y + X$	A1
$(X + Y) + Z = X + (Y + Z)$	A2
$X + X = X$	A3
$(X + Y)\cdot Z = X\cdot Z + Y\cdot Z$	A4
$(X\cdot Y)\cdot Z = X\cdot(Y\cdot Z)$	A5
$X + \delta = X$	A6
$\delta\cdot X = \delta$	A7
$a \mid b = b \mid a$	C1
$(a \mid b) \mid c = a \mid (b \mid c)$	C2
$\delta \mid a = \delta$	C3
$X \parallel Y = X \mathbin{\underline{\|}} Y + Y \mathbin{\underline{\|}} X + X \mid Y$	CM1
$a \mathbin{\underline{\|}} X = a\cdot X$	CM2
$a\cdot X \mathbin{\underline{\|}} Y = a\cdot(X \parallel Y)$	CM3
$(X + Y) \mathbin{\underline{\|}} Z = X \mathbin{\underline{\|}} Z + Y \mathbin{\underline{\|}} Z$	CM4
$a\cdot X \mid b = (a \mid b)\cdot X$	CM5
$a \mid b\cdot X = (a \mid b)\cdot X$	CM6
$a\cdot X \mid b\cdot Y = (a \mid b)\cdot(X \parallel Y)$	CM7
$(X+Y) \mid Z = X \mid Z + Y \mid Z$	CM8
$X \mid (Y+Z) = X \mid Y + X \mid Z$	CM9
$\partial_{\{d\}}(\delta) = \delta$	D0
$d \neq e \Rightarrow \partial_{\{d\}}(e) = e$	D1
$\partial_{\{d\}}(d) = \delta$	D2
$\partial_{\{d\}}(X + Y) = \partial_{\{d\}}(X) + \partial_{\{d\}}(Y)$	D3
$\partial_{\{d\}}(X\cdot Y) = \partial_{\{d\}}(X)\cdot\partial_{\{d\}}(Y)$	D4
$a \mid b \in A$	CCA
$a \in A^c \vee a = \delta$	CAA

TABLE 1. ACP.

This presentation of ACP is an alternative to the presentation in [BW90], where there are two parameters, a finite set of constants A, and a communication function γ on the constants. Then, an axiom containing an a or b can be considered as an axiom schema, and we are dealing with an equational specification ACP(A,γ). In this case, we also have for each subset H of A an encapsulation operator ∂_H.

In the present setting, we have a subsort of core atomic actions. This subsort prevents δ from being a first argument for encapsulation. CAA (Core Atoms Axiom) expresses that atomic actions are either δ or core. Proper non-core actions will for example emerge in the form of multi-actions when step bisimulation semantics is considered (not in this paper however). CCA (Communication Closure Axiom) simply expresses that the communication of two atomic actions is again an atomic action. Since the subsort A^c may be infinite, we encapsulate only singleton sets. We can generalize by writing $\partial_{\{d_1,...,d_k\}}(X)$ for $\partial_{\{d_1\}}\circ...\circ\partial_{\{d_k\}}(X)$, if $d_1,...,d_k \in A^c$.

2.2 RENAMING.

A useful extension of ACP is obtained by the addition of renaming operators (see [BB88], notation from [VAA90]). Let f be a unary function from A^c to A^c. Then we add an operator ρ_f: $P \to P$ to ACP, that renames each $d \in A^c$ into $f(d)$. We have the axioms in table 2. The theory ACP plus renaming is called ACP + RN.

$f: A^c \to A^c$	renaming function
$\rho_f: P \to P$	renaming operator.

$$\rho_f(\delta) = \delta$$
$$\rho_f(d) = f(d)$$
$$\rho_f(X + Y) = \rho_f(X) + \rho_f(Y)$$
$$\rho_f(X \cdot Y) = \rho_f(X) \cdot \rho_f(Y)$$

TABLE 2. Renaming.

2.3 SUBALGEBRA OF REDUCED MODEL SPECIFICATIONS.

Since the theory ACP of 2.1 has δ as only constant, the initial algebra is the one-point model. Thus, if we want to talk about subalgebras of initial ACP algebras, we need an additional set of constants. Extend ACP by a finite set $|A^c|$ of atomic constants, so for each $d \in |A^c|$ we have the declaration

$$d \in A^c$$

as a constant in the signature of ACP($|A^c|$). Next, if γ: $|A^c| \times |A^c| \to |A^c| \cup \{\delta\}$ is a commutative and associative function, then first, we replace axioms C1,2, CCA, CAA by axioms

$$d \mid e = \gamma(d,e)$$

for all $d,e \in |A^c|$. Secondly, we replace the axiom D1 by the axiom schema

$$\partial_{\{d\}}(e) = e \qquad \text{(for all } d,e \in |A^c| \text{ such that } d \neq e\text{).}$$

Thirdly, the axioms CM2,3,5,6,7, D2,3,4 are interpreted as axiom schemas, for all $d,e \in |A^c|$. These three steps lead to the finite equational specification ACP($|A^c|,\gamma$) of [BK84], [BW90]. Let γ_δ be the communication function always yielding δ. In order to exemplify the SRM specification method we will first consider four subsignatures of the signature of ACP($|A^c|,\gamma_\delta$):

- Σ(BPA) has sorts P,A, set of constants $|A^c|$ and functions $+,\cdot$

- $\Sigma(BPA_\delta)$ has sorts P,A, set of constants $\{\delta\} \cup |A^c|$ and functions $+,\cdot$
- $\Sigma(PA)$ has sorts P,A, set of constants $|A^c|$ and functions $+,\cdot, \|, \lfloor\!\lfloor$
- $\Sigma(PA_\delta)$ has sorts P,A, set of constants $\{\delta\} \cup |A^c|$ and functions $+,\cdot, \|, \lfloor\!\lfloor$

Let I be the initial algebra of ACP($|A^c|,\gamma_\delta$). For a subsignature Σ, consider $\langle I\rangle_\Sigma$. This is the minimal subalgebra of the Σ-reduct of I, $I|_\Sigma$. Then the following results paraphrase results that can be found e.g. in [BW90]:

- $\langle I\rangle_{\Sigma(BPA)}$ is axiomatised by axioms A1-5 of table 1 (in the sense of an initial algebra specification). They constitute the theory BPA.
- $\langle I\rangle_{\Sigma(BPA_\delta)}$ is axiomatised by axioms A1-7 of table 1. They constitute the theory BPA_δ.
- $\langle I\rangle_{\Sigma(PA)}$ is axiomatised by axioms A1-5 of table 1 and axioms M1-4 of table 3 below. They constitute the theory PA.
- $\langle I\rangle_{\Sigma(PA_\delta)}$ is axiomatised by axioms A1-7 of table 1 and axioms M1-4 of table 3 below. They constitute the theory PA_δ.

$X \| Y = X \lfloor\!\lfloor Y + Y \lfloor\!\lfloor X$	M1
$a \lfloor\!\lfloor X = a\cdot X$	M2
$a\cdot X \lfloor\!\lfloor Y = a\cdot(X \| Y)$	M3
$(X + Y) \lfloor\!\lfloor Z = X \lfloor\!\lfloor Z + Y \lfloor\!\lfloor Z$	M4

TABLE 3. Axioms for PA, PA_δ.

2.4 FUNCTIONAL PREFIXES.

Fix a natural number $n \geq 1$. This number will be a parameter of the further theory. Later on, n will be the cardinality of the message set that we consider as input or output messages. We have constants $\underline{i}$ in the language for all numbers $1,...,n$ (usually we do not make this difference explicitly, but in this case it makes things to follow more clear). Additional signature:

Sorts:

N	set of numbers $\{1,...,n\}$
P^N	functions from N to processes, represented by a sequence of length n.

Constants:

$\underline{1}, ..., \underline{n} \in N$	numbers $1, ..., n$

Functions:

$\langle\_,...,\_\rangle: P \times ... \times P \to P^N$	sequence of processes of length n, representing element of P^N
$\_\hat{}\_: A \times N \to A$	late to early conversion (adapted from [MIL89])
$\bullet_N: A \times P^N \to P$	late functional prefix
$\circ_N: A \times P^N \to P$	early functional prefix

Variables:

$p,q,r \in N$

$F \in P^N$

Axioms are in table 4 (as before, $a,b \in A$, $X,Y,X_i \in P$). We notice that there are no restrictions on ^ except for the axiom $\delta\hat{}p = \delta$.

The last axiom in table 4 is the Early Communication Axiom ECA. It says that two late action prefixes cannot communicate. We call this extension of ACP with functional prefixes Functional Prefix Algebra, $FPA_{ECA}(N)$. Of course, at this point a modification is possible by allowing late prefixes to communicate. This requires that axiom ECA is dropped, of course. It happens for instance, when we consider mobile processes (see [EN86], [MPW92]).

$$a \circ_N \langle X_1, ..., X_n\rangle = (a\hat{}\underline{1})\cdot X_1 + ... + (a\hat{}\underline{n})\cdot X_n$$
$$\delta\hat{}p = \delta$$

$$\delta \bullet_N F = \delta$$
$$(a \bullet_N \langle X_1, ..., X_n\rangle)\cdot Y = a \bullet_N \langle X_1\cdot Y, ..., X_n\cdot Y\rangle$$
$$(a \bullet_N \langle X_1, ..., X_n\rangle) \mathbin{|\!\!\lfloor} Y = a \bullet_N \langle X_1 \| Y, ..., X_n \| Y\rangle$$
$$\partial_{\{a\}}(b \bullet_N \langle X_1, ..., X_n\rangle) = \partial_{\{a\}}(b) \bullet_N \langle \partial_{\{a\}}(X_1), ..., \partial_{\{a\}}(X_n)\rangle$$

$$(a \bullet_N F) \mid b = (a \circ_N F) \mid b$$
$$a \mid (b \bullet_N F) = a \mid (b \circ_N F)$$
$$(a \bullet_N F) \mid b\cdot X = (a \circ_N F) \mid b\cdot X$$
$$a\cdot X \mid (b \bullet_N F) = a\cdot X \mid (b \circ_N F)$$
$$(a \bullet_N F) \mid (b \bullet_N G) = \delta \qquad \text{ECA}$$

TABLE 4. Early and late functional prefixing.

3. FORMS OF COMMUNICATION.

We will specialise the general theory of the previous section to describe several forms of communication. Our specifications will be of the form $ACP(|A^c|,\gamma)$ with $|A^c|$ and γ being extended step by step.

3.1 READ-SEND COMMUNICATION.

We give a specific instantiation of the set of atomic actions A and communication on A following the notation of [BK86]. We assume the set of messages is given as N (perhaps after some appropriate coding), and we assume given a finite set of communication ports $\mathcal{M}$. For each $i \in N$ and $m \in \mathcal{M}$ we have the following atomic actions:

- $r_m(i)$ read message i at port m,
- $s_m(i)$ send message i at port m,
- $c_m(i)$ communicate message i at port m.

Thus, we have $|A^c| = \{r_m(i), s_m(i), c_m(i) \mid i \in N, m \in \mathcal{M}\}$, $|A| = |A^c| \cup \{\delta\}$.
On this action set, communication is defined by means of the axioms in table 5 ($a \in A$, $p \in N$). These axioms are actually axiom schemes, so for instance the first axiom exists for every $i \in N$ and $m \in \mathcal{M}$. Adding these axioms is consistent with axioms C0,1,2,3 of ACP. These notations and axioms have been introduced in [BK86].

$r_m(i) \mid s_m(i) = c_m(i)$	
$r_m(i) \mid s_k(j) = \delta$	if $k \neq m$ or $i \neq j$
$r_m(i) \mid r_k(j) = \delta$	
$s_m(i) \mid s_k(j) = \delta$	
$c_m(i) \mid a = \delta$	
$r_m(i)$^$p = \delta$	
$s_m(i)$^$p = \delta$	
$c_m(i)$^$p = \delta$	

TABLE 5. Read-send communication

3.2 BOOLEANS.

In the sequel, we need the sort of Booleans. We use the axioms in table 6 ($p \in N$, $d \in A^c$). We use the *if...then...* operator (one-armed conditional) :→, see [BB92]. In terms of this operator, the *then...if...else...* operator (two-armed conditional) can be defined (notation taken from [HHJ+87]).

Sort:

B	booleans

Constants:

$T, F \in B$	true, false

Functions:

$\neg\_ \quad : B \to B$	negation
$\_\vee\_ \quad : B \times B \to B$	disjunction
$\_\wedge\_ \quad : B \times B \to B$	conjunction
$(\_=\_) : N \times N \to B$	data equality
$\_{:}{\to}\_ : B \times P \to P$	one-armed conditional, *if...then...*
$\_\lhd\_\rhd\_ : P \times B \times P \to P$	two-armed conditional, *then...if...else...*

Variables:

$\beta, \gamma \in B$.

$\neg T = F$	$F \vee \beta = \beta$
$\neg F = T$	$T \wedge \beta = \beta$
$T \vee \beta = T$	$F \wedge \beta = F$
$(p = p) = T$	
$(\underline{i} = \underline{j}) = F$	if $i \neq j$
$T :\rightarrow X = X$	
$F :\rightarrow X = \delta$	
$\beta :\rightarrow \delta = \delta$	
$\beta :\rightarrow (X + Y) = (\beta :\rightarrow X) + (\beta :\rightarrow Y)$	
$(\beta :\rightarrow X) \cdot Y = \beta :\rightarrow X \cdot Y$	
$(\beta :\rightarrow X) \mathbin{\|\!\lfloor} Y = \beta :\rightarrow X \mathbin{\|\!\lfloor} Y$	
$(\beta :\rightarrow X) \mid Y = \beta :\rightarrow X \mid Y$	
$X \mid (\beta :\rightarrow Y) = \beta :\rightarrow X \mid Y$	
$\partial_{\{d\}}(\beta :\rightarrow X) = \beta :\rightarrow \partial_{\{d\}}(X)$	
$\rho_f(\beta :\rightarrow X) = \beta :\rightarrow \rho_f(X)$	
$X \triangleleft \beta \triangleright Y = (\beta :\rightarrow X) + (\neg\beta :\rightarrow Y)$	

TABLE 6. Booleans.

3.3 SUBSTITUTION.

Now we introduce in the language variables over N in Boolean expressions. These are elements of the syntax, and can be considered as constants in the signature. Assume there is an infinite set of variables $V = \{v, w, v_1, ...\}$. As we have no functions defined on N, this makes that the only terms over N are the individual constants and the individual variables. Thus, each boolean expression $(k = m)$ has one of the forms $(\underline{i} = \underline{j})$, $(v = \underline{i})$, $(\underline{i} = v)$, $(v = w)$. In the presence of these variables, we can define subsitution by means of the axioms in table 7 ($a \in A$, $p,q,r \in N$, $X,Y,X_i \in P$, $\beta,\gamma \in B$).

Functions:

$\_[\_/\_]: P \times N \times V \rightarrow P$	substitution on P
$\_[\_/\_]: B \times N \times V \rightarrow B$	substitution on B
$\_[\_/\_]: N \times N \times V \rightarrow N$	substitution on N

It is possible to extend this set-up, and allow functions on N, so that we can form terms over N, consisting of variables, constants and function symbols. We do not do so for reasons of simplicity.

$$\delta[p/v] = \delta$$
$$a \in A^c \Rightarrow a[p/v] = a$$
$$(X + Y)[p/v] = X[p/v] + Y[p/v]$$
$$(X \cdot Y)[p/v] = X[p/v] \cdot Y[p/v]$$
$$(a \bullet_N \langle X_1, ..., X_n\rangle)[p/v] = a[p/v] \bullet_N \langle X_1[p/v], ..., X_n[p/v]\rangle$$
$$(X \lhd \beta \rhd Y)[p/v] = X[p/v] \lhd \beta[p/v] \rhd Y[p/v]$$

$$T[p/v] = T$$
$$F[p/v] = F$$
$$(\neg \beta)[p/v] = \neg \beta[p/v]$$
$$(\beta \wedge \gamma)[p/v] = \beta[p/v] \wedge \gamma[p/v]$$
$$(\beta \vee \gamma)[p/v] = \beta[p/v] \vee \gamma[p/v]$$
$$(q = r)[p/v] = (q[p/v] = r[p/v])$$

$$\underline{i}[p/v] = \underline{i}$$
$$v[p/v] = p$$
$$w[p/v] = w \qquad \text{if } v \neq w$$

TABLE 7. Substitution.

3.4 EARLY AND LATE ACTION PREFIXING.

Now we add a number of unary operators on P, based on [MIL80]. We also add a new input action, and a functional abstraction operator. Let $m \in \mathcal{M}$, $i \in N$ and $v \in V$. Axioms are in table 8 ($a \in A$, $X \in P$). Extra signature:

$er_m(v);\_: P \to P$	early input prefix
$lr_m(v);\_: P \to P$	late input prefix
$s_m(i);\_: P \to P$	output prefix
$c_m(i);\_: P \to P$	communication prefix
$r_m \in A^c$	input action
$\lambda v.\_: P \to P^N$	functional abstraction

We are not dealing with free and bound variables here because of the way we treat variables over N. This is why we do not need machinery to distinguish free and bound variables and α-conversion.

$s_m(i);X = s_m(i) \cdot X$
$c_m(i);X = c_m(i) \cdot X$
$er_m(v);X = r_m \circ_N \lambda v.X$
$lr_m(v);X = r_m \bullet_N \lambda v.X$
$r_m \hat{} \underline{i} = r_m(i)$
$r_m \mid a = \delta$
$\lambda v.X = \langle X[\underline{1}/v], ..., X[\underline{n}/v] \rangle$

TABLE 8. Early and late input.

Notice that we can write $er_m(v);X = r_m(1) \cdot X[\underline{1}/v] + ... + r_m(n) \cdot X[\underline{n}/v]$, so if one focuses on early input prefixes, both functional input prefixing and functional abstraction are not necessary and just serve as a notational convenience.

3.5 EXAMPLE.

The following expression denotes a process that reads in a value i in late semantics and then outputs the value $i+1 \pmod n$ along the same port:

$$lr_m(v); (s_m(2) \lhd v=\underline{1} \rhd \delta + ... + s_m(1) \lhd v=\underline{n} \rhd \delta).$$

3.6 RESTRICTED INPUT.

We can define input actions that work with a restricted domain, a subset of N. Let $D \subseteq N$.

Extra syntax:

$er^D_m(v);\_: P \to P$	restricted early input prefix
$lr^D_m(v);\_: P \to P$	restricted late input prefix
$r^D_m \in A^c$	restricted input action.

$er^D_m(v);X = r^D_m \circ_N \lambda v.X$	
$lr^D_m(v);X = r^D_m \bullet_N \lambda v.X$	
$r^D_m \hat{} \underline{i} = r_m(i)$	if $i \in D$
$r^D_m \hat{} \underline{i} = \delta$	if $i \notin D$
$r^D_m \mid a = \delta$	

TABLE 9. Restricted input.

3.8 ELIMINATION.

We can obtain an elimination theorem for the theory developed in this section, even for terms containing data variables. Thus, call a term process closed if it does not contain process variables. Then we claim that for each process closed term over this theory, there is a process closed term that is provably equal to it that does not contain the operators $\parallel$, $\lfloor\!\lfloor$, $\mid$, $\partial_{\{d\}}$, $\circ_N$, ^, [/], ;.

3.9 HOARE'S INPUT ACTION.

We consider another kind of communication which is useful to model the various features of original CSP [HOA78] and theoretical CSP [BHR84].

We start from the set of atomic actions introduced above, so we have actions δ, $r_m(i)$, $s_m(i)$, $c_m(i)$, r_m for each $i \in N$, $m \in \mathcal{M}$. As in 3.3, we have a set of variables V ranging over N. We do not want to mix the variable prefixes of 3.4 with the variable constructs here, so we assume we have a different set of variables here. Let us denote the variables here by $x,y,z,...$ instead of $v,w,...$. We add the following atomic actions:

$m?x$	$\in A^c$	Hoare's input action, for $x \in V$, $m \in \mathcal{M}$
$(x := i)$	$\in A^c$	assign a value to a variable, for $x \in V$, $i \in N$
$ass(i)$	$\in A^c$	the value has been assigned, for $i \in N$

Furthermore, we add the functions

$\lambda_p^x: P \to P$ state operator (see [BB88], [VER92])

$\lambda_p^x(\delta) = \delta$	
$\lambda_p^x(m?x) = r_m(1) + ... + r_m(n)$	
$\lambda_p^x(x{:=}i) = ass(i)$	
$\lambda_p^x(d) = d$	for d not of the form $m?x$ or $x{:=}i$
$\lambda_p^x(m?x \cdot X) = r_m(1) \cdot \lambda_1^x(X) + ... + r_m(n) \cdot \lambda_n^x(X)$	
$\lambda_p^x(x{:=}i \cdot X) = ass(i) \cdot \lambda_i^x(X)$	
$\lambda_p^x(d \cdot X) = d \cdot \lambda_p^x(X)$	for d not of the form $m?x$ or $x{:=}i$
$\lambda_p^x(X + Y) = \lambda_p^x(X) + \lambda_p^x(Y)$	
$\lambda_p^x(X \lhd \beta \rhd Y) = \lambda_p^x(X) \lhd \beta[p/x] \rhd \lambda_p^x(Y)$	

TABLE 10. Hoare's input action.

We have the equations in table 10. In the expression $\lambda_p^x(X)$, the state operator puts process X in a context where the variable x is locally known and where it has initial value p.

The CSP output action $m!i$ can simply be defined as $s_m(i)$, and the notation $m!x$ can be used as an abbreviation for

$$s_m(1) \lhd x{=}\underline{1} \rhd \delta + ... + s_m(n) \lhd x{=}\underline{n} \rhd \delta \qquad \text{(cf. 3.5).}$$

4. PROCESS PREFIX.

In section 3 we added a number of prefix operators. We can generalize this to a setting where the ; operator becomes a binary operator on processes. The early input and late input become new atomic actions, satisfying the same axioms as the ones in table 7. Further on, we also define CCS restriction and CSP synchronisation merge.

4.1 DEFINITION.

Constants:

$er_m(v) \in A^c$	early input action (for each $v \in V$, $m \in \mathcal{M}$)
$lr_m(v) \in A^c$	late input action (for each $v \in V$, $m \in \mathcal{M}$)

Functions:

$; : P \times P \to P$	process prefix

In table 9, $a \in A$, $X,Y,X_i \in P$.

We now have two types of actions, *basic actions* that satisfy $a;X = a \cdot X$, and other actions, called non-basic. Here, we have basic actions δ, $r_m(i)$, $s_m(i)$, $c_m(i)$, r_m and non-basic actions $er_m(v)$, $lr_m(v)$ and we can write $|A^c| = |A^c_b| \cup |A^c_{nb}|$, with $|A^c_b| \cap |A^c_{nb}| = \varnothing$.

$$
\begin{aligned}
& er_m(v) \mid a = \delta \\
& lr_m(v) \mid a = \delta \\
& er_m(v)\hat{\ }p = \delta \\
& lr_m(v)\hat{\ }p = \delta \\
& \\
& \delta;X = \delta \\
& r_m(i);X = r_m(i) \cdot X \\
& s_m(i);X = s_m(i) \cdot X \\
& c_m(i);X = c_m(i) \cdot X \\
& r_m;X = r_m \cdot X \\
& er_m(v);X = r_m \circ_N \lambda v.X \\
& lr_m(v);X = r_m \bullet_N \lambda v.X \\
& (X + Y);Z = X;Z + Y;Z \\
& (X \cdot Y);Z = X;(Y;Z) \\
& (a \bullet_N \langle X_1, ..., X_n \rangle);Y = a \bullet_N \langle X_1;Y, ..., X_n;Y \rangle
\end{aligned}
$$

TABLE 11. Process prefix.

4.2 EXAMPLE.

As an example of the use of the process prefix, we can define parallel input:

$$(lr_1(v) \parallel lr_2(w)) ; P(v,w)$$

describes a process that receives two inputs independently along ports 1 and 2, and then continues dependent upon these inputs, under late bisimulation semantics, whereas

$$(er_1(v) \parallel er_2(w)) ; P(v,w)$$

describes the same process under early bisimulation semantics. We can even mix early and late as in

$$(lr_1(v) \parallel lr_2(w) \parallel er_3(x)) ; P(v,w,x).$$

Further, we allow the following:

$$(lr_1(v) \parallel lr_2(w) \parallel er_3(v)) ; P(v,w).$$

4.3 EXAMPLE.

In this example, we use the abbreviation $s_m(v)$ for the term

$$s_m(1) \lhd v=\underline{1} \rhd \delta + \ldots + s_m(n) \lhd v=\underline{n} \rhd \delta \qquad \text{(cf. 3.9).}$$

With this abbreviation, we can specify a one-item buffer with input port 1 and output port 2 as follows:

$$B = (er_1(v) \,;\, s_2(v)) \cdot B.$$

A two-item buffer containing one item (initially given by v) can be specified by means of a set of two equations:

$$C = (er_1(w) \parallel s_2(v)) \,;\, D$$
$$D = (er_1(v) \parallel s_2(w)) \,;\, C.$$

4.4 ELIMINATION.

For the theory including process prefix, we still have an elimination theorem as in 3.8: also the extra operator process prefix can be eliminated from process closed terms.

4.5 ITERATION, EXITS.

We obtain another application of the process prefix if we look at the prefix iteration operator, in combination with special non-basic actions. A^c_b is the set of basic actions, actions satisfying $a;X = a \cdot X$.

Functions:

$\_^{\omega;}: P \to P$	prefix iteration
$\tilde{\ }: A^c_b \cup A^c_{nb} \to A^c_{nb}$	exit operator

$\tilde{a};X = a$
$\tilde{a} \cdot X = \tilde{a}$
$\tilde{a} \mid b = \delta$
$\tilde{a} = \tilde{b} \Rightarrow a = b$
$X^{\omega;} = X \,;\, X^{\omega;}$

TABLE 12. Iteration and exits.

4.6 EXAMPLES.

i. Put $P = (a + \tilde{b})^{\omega;}$ for basic a,b. Then $P = (a + \tilde{b});P = a;P + \tilde{b};P = a \cdot P + b$, so P solves a well-known equation.

ii. All linear systems of equations over A^c_b can be solved using this mechanism. As an example, we consider a system with three equations, all $a_{ij} \in A^c_b$. Let

$$X_1 = a_{11} \cdot X_1 + a_{12} \cdot X_2 + a_{13} \cdot X_3 + a_{14}$$
$$X_2 = a_{21} \cdot X_1 + a_{22} \cdot X_2 + a_{23} \cdot X_3 + a_{24}$$
$$X_3 = a_{31} \cdot X_1 + a_{32} \cdot X_2 + a_{33} \cdot X_3 + a_{34}.$$

Put $U_3 = \tilde{\tilde{a}}_{31} + \tilde{a}_{32} + a_{33} + \tilde{\tilde{\tilde{a}}}_{34}$

$$U_2 = \tilde{a}_{21} + a_{22} + a_{23} \cdot U_3^{\omega;} + \tilde{\tilde{a}}_{24}$$

$$U_1 = a_{11} + a_{12}\cdot U_2^{\omega;} + a_{13}\cdot U_3^{\omega;}\cdot U_2^{\omega;} + \tilde{a}_{14},$$

then it is an easy calculation to show that

$$X_1 = U_1^{\omega;} \qquad X_2 = U_2^{\omega;};U_1^{\omega;} \qquad X_3 = U_3^{\omega;};U_2^{\omega;};U_1^{\omega;}.$$

4.7 CCS RESTRICTION.

For use in the following section, we define the CCS restriction operator (with a subscript δ to indicate that we are renaming to δ, not to 0 as in CCS):

$\backslash_\delta: P \times \mathcal{M} \to P$ restriction

$X \backslash_\delta m = \partial_{\{r_m(i),\, s_m(i) \mid i = 1,\ldots,n\}}(X)$

TABLE 13. Restriction.

4.8 SYNCHRONISATION MERGE.

In order to define the CSP synchronisation merge, we start from a finite set of atomic synchronisation actions, A^c_{sy}. For each $a \in A^c_{sy}$, we add a new atomic action a^2. This gives us the set

$$A^{c2}_{sy} = \{a^2 \mid a \in A^c_{sy}\}.$$

On these atoms, we define a special communication function γ_{sy} by means of the first axioms of table 14. Next, we have the renaming operator $\rho_{2\to1}$ that removes squares again; it is based on the function $(2\to1)$ on atoms given in the next two axioms of table 14.

Finally, let H be a subset of A^c_{sy}. Then we define the following operators:

$\|_H: P \times P \to P$ synchronisation merge ($H \subseteq A^c_{sy}$)

$⫫_H: P \times P \to P$ synchronisation left merge ($H \subseteq A^c_{sy}$)

$|_H: P \times P \to P$ synchronisation communication merge ($H \subseteq A^c_{sy}$)

In the definition, we use the set $K = \{a^2 \mid a \in A^c_{sy} - H\}$.

$a \mid a = a^2$	for $a \in A^c_{sy}$
$a \mid b = \delta$	otherwise
$(2\to1)(a^2) = a$	for $a \in A^c_{sy}$
$(2\to1)(a) = a$	for $a \in A^c_{sy}$ or $a=\delta$
$X \|_H Y = \rho_{2\to1} \circ \partial_K \circ \partial_H(\rho_{2\to1}(X) \,\|\, \rho_{2\to1}(Y))$	
$X ⫫_H Y = \rho_{2\to1} \circ \partial_K \circ \partial_H(\rho_{2\to1}(X) ⫫ \rho_{2\to1}(Y))$	
$X \mid_H Y = \rho_{2\to1} \circ \partial_K \circ \partial_H(\rho_{2\to1}(X) \mid \rho_{2\to1}(Y))$	

TABLE 14. Synchronisation merge.

5. SRM SPECIFICATIONS.

Of the theory developed in section 3, we can specify several subalgebras of reduced models. As a starting point we take the theory $FPA_{ECA}(N, rsc)$. This is $FPA_{ECA}(N)$ together with the additional syntax and axioms of section 3.1 through 3.5.

5.1 BASIC VALUE MATCHING ALGEBRA.

Let I be the initial algebra of $FPA_{ECA}(N, rsc)$. The signature Σ(BVMA) is obtained by deleting functional prefixing, all prefix operators and the input actions r_m. Then $\langle I\rangle_{\Sigma(BVMA)}$ is axiomatised by ACP($|A^c|$, γ) plus Booleans of 3.2 and substitution of 3.3, where $|A^c| = \{r_m(i), s_m(i), c_m(i) \mid i \in N, m \in \mathcal{M}\}$ and γ is given by the axioms in table 4. This is the theory BVMA (Basic Value Matching Algebra), also called ACP with read-send communication, that was used in [BK86], [BAE90].

5.2 VALUE MATCHING CALCULUS.

Let I be the initial algebra of $FPA_{ECA}(N, rsc)$. The signature Σ(VMC) is listed below. Then $\langle I\rangle_{\Sigma(VMC)}$ is axiomatised by the axioms A1,2,3,6,7, CM1,4,8,9 of ACP plus the axioms in table 5 (Booleans), the axioms in table 7 (substitution) except for the sixth and seventh, and the axioms in tables 15 and 16 below. This is the theory VMC (Value Matching Calculus), very similar to finitary CCS under strong early bisimulation semantics.

By axiomatisation of an algebra by a calculus, we understand that all valid closed identities are provable from the axioms and rules of the calculus.

$er_m(v);X + er_m(v);Y =$	
$= er_m(v);X + er_m(v);Y + er_m(v);(X \lhd v{=}p \rhd Y)$	EIA
$er_m(v);X = er_m(w);X[w/v]$	if $w \notin FV(X)$
$\delta \mathbin{⫡} X = \delta$	
$er_m(v);X \mathbin{⫡} Y = er_m(v) ; (X \parallel Y)$	if $v \notin FV(Y)$
$s_m(i);X \mathbin{⫡} Y = s_m(i) ; (X \parallel Y)$	
$c_m(i);X \mathbin{⫡} Y = c_m(i) ; (X \parallel Y)$	
$\delta \mid X = \delta$	
$X \mid \delta = \delta$	
$er_m(v);X \mid s_m(i);Y = c_m(i) ; (X[\underline{i}/v] \parallel Y)$	
$er_m(v);X \mid s_k(i);Y = \delta$	if $m \neq k$
$s_m(i);X \mid er_m(v);Y = c_m(i) ; (X \parallel Y[\underline{i}/v])$	
$s_m(i);X \mid er_k(v);Y = \delta$	if $m \neq k$
$er_m(v);X \mid er_k(w);Y = \delta$	
$s_m(i);X \mid s_k(j);Y = \delta$	
$c_m(i);X \mid Y = \delta$	
$X \mid c_m(i);Y = \delta$	
$\delta \setminus_\delta m = \delta$	
$er_m(v);X \setminus_\delta m = \delta$	
$er_m(v);X \setminus_\delta k = er_m(v);(X \setminus_\delta k)$	if $m \neq k$
$s_m(i);X \setminus_\delta m = \delta$	
$s_m(i);X \setminus_\delta k = s_m(i);(X \setminus_\delta k)$	if $m \neq k$
$c_m(i);X \setminus_\delta k = c_m(i);(X \setminus_\delta k)$	
$(X + Y) \setminus_\delta m = X \setminus_\delta m + Y \setminus_\delta m$	

TABLE 15. Value matching calculus.

The signature of VMC:

Sorts:

P	processes
B	booleans

Constants:

$\delta \in P$	nil
$T, F \in B$	true, false

Functions:

$er_m(v);\_: P \to P$	early input prefix, for $m \in \mathcal{M}$, $v \in Var(N)$
$s_m(i);\_: P \to P$	output prefix, for $m \in \mathcal{M}$, $i \in N$

$c_m(i);\_ : P \to P$	communication prefix, for $m \in \mathcal{M}$, $i \in N$
$+: P \times P \to P$	alternative composition, sum
$\parallel : P \times P \to P$	parallel composition, merge
$\mathbin{⌊\!⌊} : P \times P \to P$	left-merge
$\mid : P \times P \to P$	communication merge
$\backslash_\delta: P \times \mathcal{M} \to P$	restriction
$\neg\_ \ : B \to B$	negation
$\_\vee\_ \ : B \times B \to B$	disjunction
$\_\wedge\_ \ : B \times B \to B$	conjunction
$(\_=\_): N \times N \to B$	data equality
$\_\lhd\_\rhd\_: P \times B \times P \to P$	conditional
$\_[\_/\_]: P \times N \times V \to P$	substitution on P
$\_[\_/\_]: B \times N \times V \to B$	substitution on B
$\_[\_/\_]: N \times N \times V \to N$	substitution on N.

In table 15, the crucial axiom is early input axiom EIA taken from [MPW91]. In fact, it constitutes the difference between early and late semantics. Essentially, it is the same axiom as axiom G4 of [GP91]. This specification is actually a calculus, since we make essential use of the notions of free and bound variables. We define these technicalities in the following table 16.

$FV(\delta) = \emptyset$
$FV(er_m(v);X) = FV(X) - \{v\}$
$FV(s_m(i);X) = FV(X)$
$FV(c_m(i);X) = FV(X)$
$FV(X + Y) = FV(X) \cup FV(Y)$
$FV(X \lhd \beta \rhd Y) = FV(X) \cup FV(\beta) \cup FV(Y)$

$FV(T) = \emptyset$
$FV(F) = \emptyset$
$FV(\neg \beta) = FV(\beta)$
$FV(\beta \wedge \gamma) = FV(\beta) \cup FV(\gamma)$
$FV(\beta \vee \gamma) = FV(\beta) \cup FV(\gamma)$
$FV(p = q) = FV(p) \cup FV(q)$
$FV(i) = \emptyset$
$FV(v) = \{v\}$

$(er_m(w);X)[p/v] = er_m(w) ; X[p/v]$ if $v \neq w$ and $w \notin FV(p)$
$(s_m(i);X)[p/v] = s_m(i);X$
$(c_m(i);X)[p/v] = c_m(i);X$

TABLE 16. Free variables and extra substitution axioms.

5.3 VALUE PASSING CALCULUS.

Let I be the initial algebra of $FPA_{ECA}(N, rsc)$. The signature $\Sigma(VPC)$ is the same as $\Sigma(VMC)$, only with the early input prefix replaced by the late input prefix. Then $\langle I\rangle_{\Sigma(VPC)}$ is axiomatised by the axioms of VMC without the axioms involving early input but with the axioms of table 17. The crucial difference is the absence of a counterpart of the Early Input Axiom. This is the theory VPC (Value Passing Calculus), very similar to finitary CCS under strong late bisimulation semantics.

$lr_m(v);X = lr_m(w);X[w/v]$	if $w \notin FV(X)$
$lr_m(v);X \mathbin{\lfloor\!\lfloor} Y = lr_m(v) ; (X \parallel Y)$	if $v \notin FV(Y)$
$lr_m(v);X \mid s_m(i);Y = c_m(i) ; (X[i/v] \parallel Y)$	
$lr_m(v);X \mid s_k(i);Y = \delta$	if $m \neq k$
$s_m(i);X \mid lr_m(v);Y = c_m(i) ; (X \parallel Y[i/v])$	
$s_m(i);X \mid lr_m(v);Y = \delta$	if $m \neq k$
$lr_m(v);X \mid lr_k(w);Y = \delta$	
$lr_m(v);X \backslash_\delta m = \delta$	
$lr_m(v);X \backslash_\delta k = lr_m(v);(X \backslash_\delta k)$	if $m \neq k$
$FV(lr_m(v);X) = FV(X) - \{v\}$	
$(lr_m(w);X)[p/v] = lr_m(w) ; X[p/v]$	if $v \neq w$ and $w \notin FV(p)$

TABLE 17. Value passing calculus.

5.4 VALUE PASSING ALGEBRA.

Let I be the initial algebra of $FPA_{ECA}(N, rsc)$. The signature $\Sigma(VPA)$ is listed below. Then $\langle I\rangle_{\Sigma(VPA)}$ is axiomatised by the axioms of ACP plus the axioms of table 18 below. This is the theory VPA (Value Passing Algebra), an algebraic variant of VPC.

Sorts:

P	processes
$A \subseteq P$	subsort of atomic actions
N	set of numbers $\{1,...,n\}$
P^N	functions from N to processes

Constants:

$\delta \in A$	inaction
$r_m \in A$	late input action, for $m \in \mathcal{M}$
$s_m(i) \in A$	output action, for $m \in \mathcal{M}$, $i \in N$
$c_m(i) \in A$	communication action, for $m \in \mathcal{M}$, $i \in N$

Functions:

$+: P \times P \to P$	alternative composition, sum
$\cdot: P \times P \to P$	sequential composition, product
$\parallel: P \times P \to P$	parallel composition, merge

$\mathbin{\lfloor\!\lfloor} : P \times P \to P$	left-merge
$\mid : P \times P \to P$	communication merge
$\partial : A \times P \to P$	encapsulation operator
$\langle \_, \ldots, \_ \rangle : P \times \ldots \times P \to P^N$	sequence of processes of length n
$\bullet_N : A \times P^N \to P$	late functional prefix

$r_m \mid a = \delta$	
$s_m(i) \mid a = \delta$	
$c_m(i) \mid a = \delta$	
$(r_m \bullet_N \langle X_1, ..., X_n\rangle) \mid s_m(i) = c_m(i)\cdot X_i$	
$(r_m \bullet_N F) \mid s_k(i) = \delta$	if $k \neq m$
$s_m(i) \mid (r_m \bullet_N \langle X_1, ..., X_n\rangle) = c_m(i)\cdot X_i$	
$s_m(i) \mid (r_k \bullet_N F) = \delta$	if $k \neq m$
$(r_m \bullet_N \langle X_1, ..., X_n\rangle) \mid s_m(i)\cdot Y = c_m(i)\cdot(X_i \parallel Y)$	
$(r_m \bullet_N F) \mid s_k(i)\cdot X = \delta$	if $k \neq m$
$s_m(i)\cdot Y \mid (r_m \bullet_N \langle X_1, ..., X_n\rangle) = c_m(i)\cdot(Y \parallel X_i)$	
$s_m(i)\cdot X \mid (r_k \bullet_N F) = \delta$	if $k \neq m$
$(r_m \bullet_N F) \mid r_k = \delta$	
$r_m \mid (r_k \bullet_N F) = \delta$	
$(r_m \bullet_N F) \mid r_k\cdot X = \delta$	
$r_m\cdot X \mid (r_k \bullet_N F) = \delta$	
$(r_m \bullet_N F) \mid c_k(i) = \delta$	
$c_m(i) \mid (r_k \bullet_N F) = \delta$	
$(r_m \bullet_N F) \mid c_k(i)\cdot X = \delta$	
$c_m(i)\cdot X \mid (r_k \bullet_N F) = \delta$	
$\delta \mid X = \delta$	
$X \mid \delta = \delta$	
$\delta \bullet_N F = \delta$	
$s_m(i) \bullet_N F = \delta$	
$c_m(i) \bullet_N F = \delta$	
$(a \bullet_N \langle X_1, ..., X_n\rangle)\cdot Y = a \bullet_N \langle X_1\cdot Y, ..., X_n\cdot Y\rangle$	
$(a \bullet_N \langle X_1, ..., X_n\rangle) \mathbin{\lfloor\!\lfloor} Y = a \bullet_N \langle X_1 \parallel Y, ..., X_n \parallel Y\rangle$	
$\partial_{\{a\}}(b \bullet_N \langle X_1, ..., X_n\rangle) = \partial_{\{a\}}(b) \bullet_N \langle \partial_{\{a\}}(X_1), ..., \partial_{\{a\}}(X_n)\rangle$	
$(a \bullet_N F) \mid (b \bullet_N G) = \delta$	ECA

TABLE 18. Value passing algebra.

5.5 SYNCHRONISATION MERGE.

As a last SRM specification, we consider a direct axiomatisation of the synchronisation merge. Let I be the initial algebra of ACP($|A^{c}_{sy}| \cup |A^{c2}_{sy}|$, γ_{sy}) + RN plus synchronisation merge of 4.8. The signature $\Sigma(SY)$ contains constants $|A^{c}_{sy}| \cup \{\delta\}$ and operators $+, \cdot$, $\|_H$, $\lfloor\!\lfloor_H$, $|_H$ for $H \subseteq A^{c}_{sy}$. Then $\langle I \rangle_{\Sigma(SY)}$ is axiomatised by the axioms A1-7 of ACP plus the axioms in table 19 below.

$X \|_H Y = X \lfloor\!\lfloor_H Y + Y \lfloor\!\lfloor_H X + X \mid_H Y$	
$a \lfloor\!\lfloor_H X = a \cdot X$	if $a \notin H$
$a \lfloor\!\lfloor_H X = \delta$	if $a \in H$
$a \cdot X \lfloor\!\lfloor_H Y = a \cdot (X \|_H Y)$	if $a \notin H$
$a \cdot X \lfloor\!\lfloor_H Y = \delta$	if $a \in H$
$(X + Y) \lfloor\!\lfloor_H Z = X \lfloor\!\lfloor_H Z + Y \lfloor\!\lfloor_H Z$	
$a \mid_H a = a$	if $a \in H$
$a \mid_H b = \delta$	if $a \notin H$ or $b \notin H$ or $a \neq b$
$a \cdot X \mid_H b = (a \mid_H b) \cdot X$	
$a \mid_H b \cdot X = (a \mid_H b) \cdot X$	
$a \cdot X \mid_H b \cdot Y = (a \mid_H b) \cdot (X \|_H Y)$	
$(X + Y) \mid_H Z = X \mid_H Z + Y \mid_H Z$	
$X \mid_H (Y + Z) = X \mid_H Y + X \mid_H Z$	

TABLE 19. Synchronisation merge.

6. CONCLUSION.

We conclude that this paper provides a satisfactory connection between the process calculus CCS and the process algebra axiomatisation ACP. By means of extra operators on top of ACP we can define on the one hand a new operator called process prefix, and on the other hand obtain versions of value passing CCS as subalgebras of reduced model specifications. In addition, some key features of CSP have been modeled in a similar fashion.

REFERENCES.

[BAE90] J.C.M. BAETEN, *Applications of process algebra,* Cambridge Tracts in TCS 17, Cambridge University Press 1990.

[BB88] J.C.M. BAETEN & J.A. BERGSTRA, *Global renamings in concrete process algebra,* Inf. and Comp. 78, 1988, pp. 205-245.

[BB90] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with a zero object,* in: Proc. CONCUR 90, Amsterdam (J.C.M. Baeten & J.W. Klop, eds.), Springer LNCS 458, 1990, pp. 83-98.

[BB92] J.C.M. BAETEN & J.A. BERGSTRA, *Process algebra with signals and conditions,* in: Programming and Mathematical Method, Marktoberdorf 1990 (M. Broy, ed.), NATO ASI Series F 88, Springer Verlag 1992, pp. 273-323.

[BW90] J.C.M. BAETEN & W.P. WEIJLAND, *Process algebra,* Cambridge Tracts in TCS 18, Cambridge University Press 1990.

[BZ82] J.W. DE BAKKER & J.I. ZUCKER, *Processes and the denotational semantics of concurrency,* Inf. & Control 54 (1/2), 1982, pp. 70-120.

[BK82] J.A. BERGSTRA & J.W. KLOP, *Fixed point semantics in process algebras,* report IW 206, Mathematical Centre, Amsterdam 1982.

[BK84] J.A. BERGSTRA & J.W. KLOP, *Process algebra for synchronous communication,* Inf. & Control 60, 1984, pp. 109-137.

[BK86] J.A. BERGSTRA & J.W. KLOP, *Verification of an alternating bit protocol by means of process algebra,* in: Math. Methods of Spec. and Synthesis of Software Systems '85 (W. Bibel & K.P. Jantke, eds.), Springer LNCS 215, 1986, pp. 9-23.

[BT84] J.A. BERGSTRA & J.V. TUCKER, *Top down design and the algebra of communicating processes,* Sci. of Comp. Progr. 5, 1984, pp. 171-199.

[BT87] J.A. BERGSTRA & J.V. TUCKER, *Algebraic specifications of computable and semicomputable datatypes,* TCS 50, 1987, pp. 137-181.

[BRI88] H. BRINKSMA, *On the design of extended LOTOS – a specification language for open distributed systems,* Ph.D. Thesis, University of Twente 1988.

[BHR84] S.D. BROOKES, C.A.R. HOARE & A.W. ROSCOE, *A theory of communicating sequential processes,* JACM 31 (3), 1984, pp. 560-599.

[EN86] U. ENGBERG & M. NIELSEN, *A calculus of communicating systems with label passing,* report DAIMI PB-208, Aarhus University 1986.

[GP91] J.F. GROOTE & A. PONSE, *Process algebra with guards (combining Hoare logic with process algebra),* in: Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 235-249.

[HI91] M. HENNESSY & M. INGOLFSDOTTIR, *A theory of communicating processes with value-passing,* I&C 1991.

[HL93] M. HENNESSY & H. LIN, *Proof systems for message-passing process algebras,* report 5/93, University of Sussex 1993.

[HOA78] C.A.R. HOARE, *Communicating sequential processes*, CACM 21, 1978, pp. 666-677.

[HOA85] C.A.R. HOARE, *Communicating sequential processes*, Prentice Hall 1985.

[HHJ+87] C.A.R. HOARE, I.J. HAYES, HE JIFENG, C.C. MORGAN, A.W. ROSCOE, J.W. SANDERS, I.H. SORENSEN, J.M. SPIVEY & B.A. SUFRIN, *Laws of programming,* CACM 30, 1987, pp. 672-686.

[KAM79] S. KAMIN, *Some definitions for algebraic datatype specifications,* ACM Sigplan Notices 14, 1979, pp. 28-37.

[MIL80] R. MILNER, *A calculus for communicating systems,* Springer LNCS 92, 1980.

[MIL89] R. MILNER, *Communication and concurrency,* Prentice-Hall 1989.

[MPW91] R. MILNER, J. PARROW & D. WALKER, *Modal logics for mobile processes,* in: Proc. CONCUR'91, Amsterdam (J.C.M. Baeten & J.F. Groote, eds.), Springer LNCS 527, 1991, pp. 45-60.

[MPW92] R. MILNER, J. PARROW & D. WALKER, *A calculus of mobile processes,* I&C 100, 1992, pp. 1-77.

[PAR81] D.M.R. PARK, *Concurrency and automata on infinite sequences,* in: Proc. 5th GI Conf. (P. Deussen, ed.), Springer LNCS 104, 1981, pp. 167-183.

[VAA90] F.W. VAANDRAGER, *Algebraic techniques for concurrency and their application,* Ph.D. Thesis, University of Amsterdam 1990.

[VER92] C. VERHOEF, *Linear unary operators in process algebra,* Ph.D. Thesis, University of Amsterdam 1992.