Formal Aspects of Computing (1994) 6: 250–268

Formal Aspects of Computing

On Sequential Composition, Action Prefixes and Process Prefix

J. C. M. Baeten[1] and J. A. Bergstra[2]

[1] Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands; and [2] Programming Research Group, University of Amsterdam, Amsterdam, The Netherlands, and Department of Philosophy, Utrecht University, Utrecht, The Netherlands

Keywords: Process algebra; Process calculus; Sequential composition; Action prefix; Process prefix; CCS, ACP

Abstract. 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.

1. Introduction

The purpose of this paper is to present a family of axiom systems that illustrate 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

Correspondence and offprint requests to: J. C. M. Baeten, Department of Mathematics and Computer Science, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands.

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 $er_m(v);\_$ resp. $lr_m(v);\_$ 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 $er_m(v) \cdot P \cdot Q$ cannot bind v in $P \cdot Q$ because of the scoping ambiguity ($(er_m(v) \cdot P) \cdot Q$ vs. $er_m(v) \cdot (P \cdot Q)$). We conclude that binding actions must be prefixes.

We concentrate on finite processes, restricting attention to 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 placed 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 $\cdot$ 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 modelled 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.

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

$$\forall a \in A \forall b \in A(a|b = b|a).$$

The signature of ACP is as follows:

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].
$\| : P \times P \to P$	Parallel composition, merge. Laws CM1–9 from [BK84].
$\lfloor\!\lfloor : P \times P \to P$	left-merge. From [BK82].
$\| : P \times P \to P$	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].

Table 1. ACP

$X+Y=Y+X$	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
$X\parallel Y=X\between Y+Y\between X+X\mid Y$	CM1
$a\between X=a\cdot X$	CM2
$a\cdot X\between Y=a\cdot(X\parallel Y)$	CM3
$(X+Y)\between Z=X\between Z+Y\between Z$	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

Variables:

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

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 $\partial_{\{d_1,\ldots,d_k\}}(X)$ for $\partial_{\{d_1\}}\circ\ldots\circ\partial_{\{d_k\}}(X)$, if $d_1,\ldots,d_k\in A^c$.

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

Table 2. Renaming

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

add an operator $\rho_f:\mathsf{P}\to\mathsf{P}$ to ACP, that renames each $\mathsf{d}\in\mathsf{A}^c$ into $\mathsf{f}(\mathsf{d})$. We have the axioms in Table 2. The theory ACP plus renaming is called ACP+RN.

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

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 $|\mathsf{A}^c|$ of atomic constants, so for each $\mathsf{d}\in|\mathsf{A}^c|$ we have the declaration

$$\mathsf{d}\in\mathsf{A}^c$$

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

$$\mathsf{d}\,|\,\mathsf{e} = \gamma(\mathsf{d},\mathsf{e})$$

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

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

Thirdly, the axioms CM2, 3, 5, 6, 7, D2, 3, 4 are interpreted as axiom schemas, for all $\mathsf{d},\mathsf{e}\in|\mathsf{A}^c|$. These three steps lead to the finite equational specification ACP($|\mathsf{A}^c|,\gamma$) 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($|\mathsf{A}^c|,\gamma_\delta$):

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

Let I be the initial algebra of ACP($|\mathsf{A}^c|,\gamma_\delta$). 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]:

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

Table 3. Axioms for PA, PA_δ

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

2.4. Functional Prefixes

Fix a natural number $n \geqslant 1$. 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 $1, \ldots, n$ (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 $\{1, \ldots, n\}$
P^N	functions from N to processes, represented by a sequence of length n.

Constants:

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

Functions:

$\langle \_, \ldots, \_ \rangle : P \times \ldots \times P \to P^N$	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 $\hat{}$ 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

Table 4. Early and late functional prefixing

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

$\delta \hat{} p = \delta$

$\delta \bullet_N F = \delta$

$(a \bullet_N \langle X_1, \ldots, X_n \rangle) \cdot Y = a \bullet_N \langle X_1 \cdot Y, \ldots, X_n \cdot Y \rangle$

$(a \bullet_N \langle X_1, \ldots, X_n \rangle) \mathbin{\lfloor\!\lfloor} Y = a \bullet_N \langle X_1 \| Y, \ldots, X_n \| Y \rangle$

$\partial_{\{a\}}(b \bullet_N \langle X_1, \ldots, X_n \rangle) = \partial_{\{a\}}(b) \bullet_N \langle \partial_{\{a\}}(X_1), \ldots, \partial_{\{a\}}(X_n) \rangle$

$(a \bullet_N F) \mid b = (a \circ_N F) \mid b$

$a \mid (b \bullet_N F) = a \mid (b \circ_N F)$

$(a \bullet_N F) \mid b \cdot X = (a \circ_N F) \mid b \cdot X$

$a \cdot X \mid (b \bullet_N F) = a \cdot X \mid (b \circ_N F)$

$(a \bullet_N F) \mid (b \bullet_N G) = \delta$ ECA

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]).

3. Forms of Communication

We will specialize 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].

Table 5. Read-send communication

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

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) $:\rightarrow$, 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:

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

Variables:

$\beta, \gamma \in B$.

Table 6. Booleans

$\neg T = F \qquad F \vee \beta = \beta$
$\neg F = T \qquad T \wedge \beta = \beta$
$T \vee \beta = t \qquad F \wedge \beta = F$

$(p = p) = T$
$(i = j) = F \quad \text{if } i \neq j$

$T :\to X = X$
$F :\to X = \delta$
$\beta :\to \delta = \delta$
$\beta :\to (X + Y) = (\beta :\to X) + (\beta :\to Y)$
$(\beta :\to X) \cdot Y = \beta :\to X \cdot Y$
$(\beta :\to X) \between Y = \beta :\to X \between Y$
$(\beta :\to X) \mid Y = \beta :\to X \mid Y$
$X \mid (\beta :\to Y) = \beta :\to X \mid Y$
$\partial_{\{d\}}(\beta :\to X) = \beta :\to \partial_{\{d\}}(X)$
$\rho_f(\beta :\to X) = \beta :\to \rho_f(X)$
$X \lhd \beta \rhd Y = (\beta :\to X) + (\neg\beta :\to Y)$

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.

Table 7. Substitution

$\delta[p/v] = \delta$
$a \in A^c \Rightarrow a[p/v] = a$
$(X + Y)\,[p/v] = X[p/v] + Y[p/v]$
$(X \cdot Y)\,[p/v] = X[p/v] \cdot Y[p/v]$

$(a \bullet_N \langle X_1, \ldots, X_n \rangle)\,[p/v] = a[p/v] \bullet_N \langle X_1[p/v], \ldots, X_n[p/v] \rangle$
$(X \lhd \beta \rhd Y)\,[p/v] = X[p/v] \lhd \beta[p/v] \rhd Y[p/v]$

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

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

Assume there is an infinite set of variables $V = \{v, w, v_1, \ldots\}$. 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 substitution 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:

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

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.

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.

Table 8. Early and late input

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

Notice that we can write $er_m(v);X = r_m(1).X[\underline{1}/v] + \ldots + r_m(n) \cdot X[\underline{n}/v]$, 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:

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

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:

$er_m^D(v);\_:P \to P$	restricted early input prefix
$lr_m^D(v);\_:P \to P$	restricted late input prefix
$r_m^D \in A^c$	restricted input action.

Table 9. Restricted input

$er_m^D(v);X = r_m^D \circ_N \lambda v \cdot X$	
$lr_m^D(v);X = r_m^D \bullet_N \lambda v \cdot X$	
$r_m^D \hat{} i = r_m(i)$	if $i \in D$
$r_m^D \hat{} i = \delta$	if $i \notin D$
$r_m^D \mid a = \delta$	

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 $|, \lfloor\!\lfloor, \lfloor, \partial_{\{d\}}, \circ_N, \hat{}, [/], ;$.

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 $x, y, z, \ldots$ instead of $v, w, \ldots$. 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])

Table 10. Hoare's input action

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

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

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

4. Process Prefix

In section 3 we added a number of prefix operators. We can generalise 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 $v \in V$, $m \in \mathcal{M}$)

Table 11. Process prefix

$er_m(v) \mid a = \delta$
$lr_m(v) \mid a = \delta$
$er_m(v) \wedge p = \delta$
$lr_m(v) \wedge p = \delta$
$\delta;X = \delta$
$r_m(i);X = r_m(i) \cdot X$
$s_m(i);X = s_m(i) \cdot X$
$c_m(i);X = c_m(i) \cdot X$
$r_m;X = r_m \cdot X$
$er_m(v);X = r_m \circ_N \lambda v \cdot X$
$lr_m(v);X = r_m \bullet_N \lambda v \cdot X$
$(X + Y);Z = X;Z + Y;Z$
$(X \cdot Y);Z = X;(Y;Z)$
$(a \bullet_N \langle X_1, \ldots, X_n \rangle);Y = a \bullet_N \langle X_1;Y, \ldots, X_n;Y \rangle$

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 $er_m(v)$, $Ir_m(v)$ and we can write $|A^c| = |A^c_b| \cup |A^c_{nb}|$, with $|A^c_b| \cap |A^c_{nb}| = \varnothing$.

4.2. Example

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

$$(Ir_1(v) \parallel Ir_2(w));P(v, w)$$

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

$$(Ir_1(v) \parallel Ir_2(w) \parallel er_3(x));P(v, w, x).$$

Further, we allow the following:

$$(Ir_1(v) \parallel Ir_2(w) \parallel er_3(v));P(v, w).$$

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.

Table 12. Iteration and exits

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

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_b^c is the set of basic actions, actions satisfying $a;X = a\cdot X$.

Functions:

$\_^{\omega;}: P \to P$	prefix iteration
$\tilde{}: A_b^c \cup A_{nb}^c \to A_{nb}^c$	exit operator

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_b^c can be solved using this mechanism. As an example, we consider a system with three equations, all $a_{ij} \in A_b^c$. Let

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

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

Table 13. Restriction

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

4.8. Synchronisation Merge

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

$$A_{sy}^{c2} = \{a^2 \mid a \in A_{sy}^c\}.$$

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 $\rho_{2\to 1}$ that removes squares again; it is based on the function $(2 \to 1)$ on atoms given in the next two axioms of Table 14.

Finally, let H be a subset of A_{sy}^c. Then we define the following operators;

$\|_H: P \times P \to P$	synchronisation merge $(H \subseteq A_{sy}^c)$
$\lfloor\!\lfloor_H: P \times P \to P$	synchronisation left merge $(H \subseteq A_{sy}^c)$
$\|_H: P \times P \to P$	synchronisation communication merge $(H \subseteq A_{sy}^c)$

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

Table 14. Synchronisation merge

$a \mid a = a^2$	for $a \in A_{sy}^c$
$a \mid b = \delta$	otherwise
$(2 \to 1)(a^2) = a$	for $a \in A_{sy}^c$
$(2 \to 1)(a) = a$	for $a \in A_{sy}^c$ or $a = \delta$
$X \|_H Y = \rho_{2\to 1} \circ \partial_K \circ \partial_H(\rho_{2\to 1}(X) \| \rho_{2\to 1}(Y))$	
$X \lfloor\!\lfloor_H Y = \rho_{2\to 1} \circ \partial_K \circ \partial_H(\rho_{2\to 1}(X) \lfloor\!\lfloor \rho_{2\to 1}(Y))$	
$X \|_H Y = \rho_{2\to 1} \circ \partial_K \circ \partial_H(\rho_{2\to 1}(X) \mid \rho_{2\to 1}(Y))$	

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 $\mathrm{FPA}_{\mathrm{ECA}}(N, rsc)$. This is $\mathrm{FPA}_{\mathrm{ECA}}(N)$ 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 $\mathrm{FPA}_{\mathrm{ECA}}(N, rsc)$. The signature Σ(BVMA) is obtained by deleting functional prefixing, all prefix operators and the input actions r_m. Then $\langle I \rangle_{\Sigma(\mathrm{BVMA})}$ is axiomatised by $\mathrm{ACP}(|A^c|, \gamma)$ 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 $\mathrm{FPA}_{\mathrm{ECA}}(N, rsc)$. The signature Σ(VMC) is listed below. Then $\langle I \rangle_{\Sigma(\mathrm{VMC})}$ is axiomatised by the axioms A1, 2, 3, 6, 7, CM1, 4, 8, 9 of

Table 15. Value matching 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 $w \notin FV(X)$
$\delta \between X = \delta$	
$er_m(v);X \between Y = er_m(v);(X \parallel Y)$	if $v \notin FV(Y)$
$s_m(i);X \between Y = s_m(i);(X \parallel Y)$	
$c_m(i);X \between Y = c_m(i);(X \parallel Y)$	
$\delta \mid X = \delta$	
$X \mid \delta = \delta$	
$er_m(v);X \mid s_m(i);Y = c_m(i);(X[i/v] \parallel Y)$	
$er_m(v);X \mid s_k(i);Y = \delta$	if $m \neq k$
$s_m(i);X \mid er_m(v);Y = c_m(i);(X \parallel Y[i/v])$	
$s_m(i);X \mid er_k(v);Y = \delta$	if $m \neq k$
$er_m(v);X \mid er_k(w);Y = \delta$	
$s_m(i);X \mid s_k(j);Y = \delta$	
$c_m(i);X \mid Y = \delta$	
$X \mid c_m(i);Y = \delta$	
$\delta \backslash_\delta m = \delta$	
$er_m(v);X \backslash_\delta m = \delta$	
$er_m(v);X \backslash_\delta k = er_m(v);(X \backslash_\delta k)$	if $m \neq k$
$s_m(i);X \backslash_\delta m = \delta$	
$s_m(i);X \backslash_\delta k = s_m(i);(X \backslash_\delta k)$	if $m \neq k$
$c_m(i);X \backslash_\delta k = c_m(i);(X \backslash_\delta k)$	
$(X+Y) \backslash_\delta m = X \backslash_\delta m + Y \backslash_\delta m$	

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.
The signature of VMC:

Sorts:

P	process
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
$\between : P \times P \to P$	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.

Table 16. Free variables and extra substitution axioms

$FV(\delta) = \varnothing$	
$FV(er_m(v);X) = FV(X) - \{v\}$	
$FV(s_m(i);X) = FV(X)$	
$FV(c_m(i);X) = FV(X)$	
$FV(X + Y) = FV(X) \cup FV(Y)$	
$FV(X \lhd \beta \rhd Y) = FV(X) \cup FV(\beta) \cup FV(Y)$	
$FV(T) = \varnothing$	
$FV(F) = \varnothing$	
$FV(\neg \beta) = FV(\beta)$	
$FV(\beta \wedge \gamma) = FV(\beta) \cup FV(\gamma)$	
$FV(\beta \vee \gamma) = FV(\beta) \cup FV(\gamma)$	
$FV(p = q) = FV(p) \cup FV(q)$	
$FV(i) = \varnothing$	
$FV(v) = \{v\}$	
$(er_m(w);X)[p/v] = er_m(w);X[p/v]$	if $v \neq w$ and $w \notin FV(p)$
$(s_m(i);X)[p/v] = s_m(i);X$	
$(c_m(i);X)[p/v] = c_m(i);X$	

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

Table 17. Value passing calculus

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

$\langle I \rangle_{\Sigma(\mathrm{VPC})}$ 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.

5.4. Value Passing Algebra

Let I be the initial algebra of $\mathrm{FPA}_{\mathrm{ECA}}(\mathsf{N}, \mathrm{rsc})$. The signature $\Sigma(\mathrm{VPA})$ is listed below. Then $\langle I \rangle_{\Sigma(\mathrm{VPA})}$ 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.

Table 18. Value passing algebra

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

Sorts:

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

Constants:

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

Functions:

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

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 $(|A_{sy}^c| \cup |A_{sy}^{c2}|, \gamma_{sy}) + RN$ plus synchronisation merge of 4.8. The signature $\Sigma(SY)$ contains constants $|A_{sy}^c| \cup \{\delta\}$ and operators $+, \cdot, \|_H, \lfloor_H, |_H$ for $H \subseteq A_{sy}^c$. Then $\langle I\rangle_{\Sigma(SY)}$ is axiomatised by the axioms A1–7 of ACP plus the axioms in Table 19 below.

Table 19. Synchronisation merge

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

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.

Acknowledgement

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

References

[Bae90] Baeten, J.C.M.: *Applications of process algebra*, Cambridge Tracts in TCS 17, Cambridge University Press 1990.

[BB88] Baeten, J.C.M. and Bergstra, J.A.: *Global renamings in concrete process algebra*, Inf. and Comp. 78, 1988, pp. 205–245.

[BB90] Baeten, J.C.M. and Bergstra, J.A.: *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] Baeten, J.C.M. and Bergstra, J.A.: *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] Baeten, J.C.M. and Weijland, W.P.: *Process algebra*, Cambridge Tracts in TCS 18, Cambridge University Press, 1990.

[BZ82] de Bakker, J.W. and Zucker, J.I.: *Processes and the denotational semantics of concurrency*, Inf. & Control 54 (1/2), 1982, pp. 70–120.

[BK82] Bergstra, J.A. and Klop, J.W.: *Fixed point semantics in process algebra*, report IW 206, Mathematical Centre, Amsterdam 1982.

[BK84] Bergstra, J.A. and Klop, J.W.: *Process algebra for synchronous communication*, Inf. & Control 60, 1984, pp. 109–137.

[BK86] Bergstra, J.A. and Klop, J.W.: *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] Bergstra, J.A. and Tucker, J.V.: *Top down design and the algebra of communicating processes*, Sci. of Comp. Progr. 5, 1984, pp. 171–199.

[BT87] Bergstra, J.A. and Tucker, J.V.: *Algebraic specifications of computable and semicomputable datatypes*, TCS 50, 1987, pp. 137–181.

[Bri88] Brinksma, H.: *On the design of extended LOTOS – a specification language for open distributed systems*, Ph.D. Thesis, University of Twente 1988.

[BHR84] Brookes, S.D., Hoare, C.A.R. and Roscoe, A.W.: *A theory of communicating sequential processes*, JACM 31 (3), 1984, pp. 560–599.

[EN86] Engberg, U. and Nielsen, M.: *A calculus of communicating systems with label passing*, report DAIMA PB-208, Aarhus University 1986.

[GP91] Groote, J.F. and Ponse, A.: *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] Hennessy, M. and Ingolfsdottir, M.: *A theory of communicating processes with value-passing*, I&C 1991.

[HL93] Hennessy, M. and Lin, H.: *Proof systems for message-passing process algebras*, report 5/93, University of Sussex 1993.

[Hoa78] Hoare, C.A.R.: *Communicating sequential processes*, CACM 21, 1978, pp. 666–677.

[Hoa85] Hoare, C.A.R.: *Communicating sequential processes*, Prentice Hall 1985.

[HHJ+87] Hoare, C.A.R., Hayes, I.J., He Jifeng, Morgan, C.C., Roscoe, A.W., Sanders, J.W. et al.: *Laws of programming*, CACM 30, 1987, pp. 672–686.

[Kam79] Kamin, S.: *Some definitions for algebraic datatype specifications*, ACM Sigplan Notices 14, 1979, pp. 28–37.

[Mil80] Milner, R.: *A calculus for communicating systems*, Springer LNCS 92, 1980.

[Mil89] Milner, R.: *Communication and concurrency*, Prentice-Hall 1989.

[MPW91] Milner, R., Parrow, J. and Walker, D.: *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] Milner, R., Parrow, J. & Walker, D.: *A calculus of mobile processes*, I&C 100, 1992, pp. 1–77.

[Par81] Park, D.M.R.: *Concurrency and automata on infinite sequences*, in: Proc. 5th GI Conf. (P. Deussen, ed.), Springer LNCS 104, 1981, pp. 167–183.

[Vaa90] Vaandrager, F.W.: *Algebraic techniques for concurrency and their application*, Ph.D. Thesis, University of Amsterdam 1990.

[Ver92] Verhoef, C.: *Linear unary operators in process algebra*, Ph.D. Thesis, University of Amsterdam 1992.

Received September 1991
Accepted in revised form April 1993 *by J. V. Tucker*