A Process Calculus with Finitary Comprehended Terms

J.A. Bergstra and C.A. Middelburg

Informatics Institute, Faculty of Science, University of Amsterdam,
Science Park 904, 1098 XH Amsterdam, the Netherlands
J.A.Bergstra@uva.nl, C.A.Middelburg@uva.nl

Abstract. We introduce the notion of an ACP process algebra and the notion of a meadow enriched ACP process algebra. The former notion originates from the models of the axiom system ACP. The latter notion is a simple generalization of the former notion to processes in which data are involved, the mathematical structure of data being a meadow. Moreover, for all associative operators from the signature of meadow enriched ACP process algebras that are not of an auxiliary nature, we introduce variable-binding operators as generalizations. These variable-binding operators, which give rise to comprehended terms, have the property that they can always be eliminated. Thus, we obtain a process calculus whose terms can be interpreted in all meadow enriched ACP process algebras. Use of the variable-binding operators can have a major impact on the size of terms.

Keywords: ACP process algebra, meadow enriched ACP process algebra, variable-binding operator, comprehended term, process calculus.

1998 ACM Computing Classification: D.1.3, F.1.2, F.4.1.

1 Introduction

In many formalisms proposed for the description and analysis of processes in which data are involved, algebraic specifications of the data types concerned have to be given over and over again. This is also the case with the principal ACP-based formalisms proposed for the description and analysis of processes in which data are involved, to wit μCRL [16,17] and PSF [23]. There is a mismatch between the process specification part and the data specification part of these formalisms. Firstly, there is a choice of one built-in type of processes, whereas there is a choice of all types of data that can be specified algebraically. Secondly, the semantics of the data specification part is its initial algebra in the case of PSF and its class of minimal Boolean preserving algebras in the case of μCRL, whereas the semantics of the process specification part is a model based on transition systems and bisimulation equivalence. Sticking to this mismatch, no lasting axiomatizations in the style of ACP has emerged for process algebras that have to do with processes in which data are involved.

Our first main objective is to obtain a lasting axiomatization in the style of ACP for process algebras that have to do with processes in which data are involved. To achieve this objective, we first introduce the notion of an ACP process algebra and then the notion of a meadow enriched ACP process algebra.

ACP process algebras are essentially models of the axiom system ACP. Meadow enriched ACP process algebras are data enriched ACP process algebras in which the mathematical structure for data is a meadow. Meadows were defined for the first time in [13]. The prime example of a meadow is the rational number field with the multiplicative inverse operation made total by imposing that the multiplicative inverse of zero is zero. Although the notion of a meadow enriched ACP process algebra is a simple generalization of the notion of an ACP process algebra, it is an interesting one: there is a multitude of finite and infinite meadows and meadows obviate the need for Boolean values and operations on data that yield Boolean values to deal with conditions on data.

In the work on ACP, the emphasis has always been on axiom systems. In this paper, we put the emphasis on algebras. That is, ACP process algebras are looked upon in the same way as groups, rings, fields, etc. are looked upon in universal algebra (see e.g. [14]). The set of equations that are taken to characterize ACP process algebras is a revision of the axiom system ACP. The revision is primarily a matter of streamlining. However, it also involves a minor generalization that allows for the generalization to meadow enriched ACP process algebras to proceed smoothly.

In μCRL and PSF, we find variable-binding operators generalizing associative operators of ACP. Our second main objective is to determine to what extent such variable-binding operators fit in with meadow enriched ACP process algebras. To achieve this objective, we introduce, for all associative operators from the signature of meadow enriched ACP process algebras that are not of an auxiliary nature, variable-binding operators as generalizations.

These variable-binding operators, which give rise to comprehended terms, have the property that they can always be eliminated. That is, for each comprehended term, we can derive from axioms concerning the variable-binding operators that the comprehended term is equal to a term over the signature of meadow enriched ACP process algebras. Those axioms are axioms of a calculus because the distinction between free and bound variables is essential in derivations. The terms of this process calculus are interpreted in meadow enriched ACP process algebras.

Full elimination of all variable-binding operators occurring in a comprehended term can lead to a combinatorial explosion. We show that a combinatorial explosion can be prevented if variable-binding operators that bind variables with a two-valued range are still permitted in the resulting term. We also show that in the latter case the size of the resulting term can be further reduced if we add an identity element for sequential composition to meadow enriched ACP process algebras. Moreover, we demonstrate that there is an alternative to introducing variable-binding operators for several associative operators on processes if we add a sort of process sequences and suitable operators on process sequences to meadow enriched ACP process algebras.

For readability, it is imprecisely said above that the mathematical structure for data in meadow enriched ACP process algebras is a meadow. It is actually a signed meadow, i.e. a meadow expanded with a signum operation. In the presence of a signum operation, the ordering on the elements of a meadow that corresponds with the usual ordering on the elements of a field becomes definable.

This paper is organized as follows. First, we give a brief summary of signed meadows (Section 2). Next, we introduce the notion of an ACP process algebra (Section 3) and the notion of a meadow enriched ACP process algebra (Section 4). After that, we associate a calculus with meadow enriched ACP process algebras (Section 5) and define the interpretation of terms of this calculus in meadow enriched ACP process algebras (Section 6). Following this, we investigate the consequences of elimination of variable-binding operators from comprehended terms on the size of the resulting terms (Section 7). Then, we investigate the effects of adding an identity element for sequential composition to ACP process algebras (Section 8) and the effects of adding process sequences to ACP process algebras (Section 9). Finally, we make some concluding remarks (Section 10).

This paper consolidates material from [9,10].

2 Signed Meadows

In this paper, the mathematical structure for data is a signed meadow. In this section, we give a brief summary of signed meadows.

A meadow is a field with the multiplicative inverse operation made total by imposing that the multiplicative inverse of zero is zero. A signed meadow is a meadow expanded with a signum operation. Meadows were defined for the first time in [13] and were investigated in e.g. [5,6,11]. The expansion of meadows with a signum operation originates from [5].

The signature of meadows is the same as the signature of fields. It is a one-sorted signature. We make the single sort explicit because we will extend this signature to a two-sorted signature in Section 4. The signature of meadows consists of the sort $\mathbf{Q}$ of *quantities* and the following constants and operators:

- the constants $0 : \to \mathbf{Q}$ and $1 : \to \mathbf{Q}$;
- the binary *addition* operator $+ : \mathbf{Q} \times \mathbf{Q} \to \mathbf{Q}$;
- the binary *multiplication* operator $\cdot : \mathbf{Q} \times \mathbf{Q} \to \mathbf{Q}$;
- the unary *additive inverse* operator $- : \mathbf{Q} \to \mathbf{Q}$;
- the unary *multiplicative inverse* operator ${}^{-1} : \mathbf{Q} \to \mathbf{Q}$.

We assume that there is a countably infinite set $\mathcal{U}$ of variables of sort $\mathbf{Q}$, which contains u, v and w, with and without subscripts. Terms are built as usual. We use infix notation for the binary operators $+$ and $\cdot$, prefix notation for the unary operator $-$, and postfix notation for the unary operator ${}^{-1}$. We use the usual precedence convention to reduce the need for parentheses. We introduce subtraction and division as abbreviations: $p - q$ abbreviates $p + (-q)$ and p/q abbreviates $p \cdot q^{-1}$. For each non-negative natural number n, we write $\underline{n}$

Table 1. Axioms for meadows

$(u + v) + w = u + (v + w)$	$(u \cdot v) \cdot w = u \cdot (v \cdot w)$	$(u^{-1})^{-1} = u$
$u + v = v + u$	$u \cdot v = v \cdot u$	$u \cdot (u \cdot u^{-1}) = u$
$u + 0 = u$	$u \cdot 1 = u$	
$u + (-u) = 0$	$u \cdot (v + w) = u \cdot v + u \cdot w$	

for the numeral for n. That is, the term $\underline{n}$ is defined by induction on n as follows: $\underline{0} = 0$ and $\underline{n+1} = \underline{n} + 1$. We also use the notation p^n for exponentiation with a natural number as exponent. For each term p over the signature of meadows, the term p^n is defined by induction on n as follows: $p^0 = 1$ and $p^{n+1} = p^n \cdot p$.

The constants and operators from the signature of meadows are adopted from rational arithmetic, which gives an appropriate intuition about these constants and operators.

A *meadow* is an algebra with the signature of meadows that satisfies the equations given in Table 1. Thus, a meadow is a commutative ring with identity equipped with a multiplicative inverse operation ${}^{-1}$ satisfying the reflexivity equation $(u^{-1})^{-1} = u$ and the restricted inverse equation $u \cdot (u \cdot u^{-1}) = u$. From the equations given in Table 1, the equation $0^{-1} = 0$ is derivable (see [13]).

A *non-trivial meadow* is a meadow that satisfies the *separation axiom*

$$0 \neq 1 \;;$$

and a *cancellation meadow* is a meadow that satisfies the *cancellation axiom*

$$u \neq 0 \land u \cdot v = u \cdot w \Rightarrow v = w \;,$$

or equivalently, the *general inverse law*

$$u \neq 0 \Rightarrow u \cdot u^{-1} = 1 \;.$$

Important properties of non-trivial cancellation meadows are $u/u = 0 \Leftrightarrow u = 0$ and $u/u = 1 \Leftrightarrow u \neq 0$. Henceforth, we will write $p \triangleleft r \triangleright q$ for $(1 - r/r) \cdot p + (r/r) \cdot q$. For non-trivial cancellation meadows, $p \triangleleft r \triangleright q$ can be read as follows: if r equals 0 then p else q.

Each field with the multiplicative inverse operation made total by imposing that the multiplicative inverse of zero is zero is a non-trivial meadow. The prime example of a non-trivial cancellation meadow is the rational number field with the multiplicative inverse operation made total by imposing that the multiplicative inverse of zero is zero.

A *signed meadow* is a meadow expanded with a unary *signum* operation s satisfying the equations given in Table 2. In combination with the cancellation axiom, the last equation in this table is equivalent to the conditional equation

Table 2. Additional axioms for signum operation

$\mathsf{s}(u/u) = u/u$	$\mathsf{s}(u^{-1}) = \mathsf{s}(u)$
$\mathsf{s}(1 - u/u) = 1 - u/u$	$\mathsf{s}(u \cdot v) = \mathsf{s}(u) \cdot \mathsf{s}(v)$
$\mathsf{s}(-1) = -1$	$(1 - \frac{\mathsf{s}(u)-\mathsf{s}(v)}{\mathsf{s}(u)-\mathsf{s}(v)}) \cdot (\mathsf{s}(u+v) - \mathsf{s}(u)) = 0$

$\mathsf{s}(u) = \mathsf{s}(v) \Rightarrow \mathsf{s}(u+v) = \mathsf{s}(u)$. In signed meadows, the predicates $<$ and $>$ are defined as follows:

$$u < v \Leftrightarrow 1 + \mathsf{s}(u - v) = 0 \;,$$
$$u > v \Leftrightarrow 1 - \mathsf{s}(u - v) = 0 \;.$$

In [5], it is shown that the equational theories of signed meadows and signed cancellation meadows are identical.

3 ACP Process Algebras

In this section, we introduce the notion of an ACP process algebra. This notion originates from the models of ACP, an axiom system that was first presented in [7]. A comprehensive introduction to ACP can be found in [3,15].

It is assumed that a fixed but arbitrary set A of *atomic action names*, with $\delta \notin \mathsf{A}$, has been given.

The signature of ACP process algebras is a one-sorted signature. We make the single sort explicit because we will extend this signature to a two-sorted signature in Section 4. The signature of ACP process algebras consists of the sort $\mathbf{P}$ of *processes* and the following constants, operators, and predicate symbols:

- the *deadlock* constant $\delta : \to \mathbf{P}$;
- for each $e \in \mathsf{A}$, the *atomic action* constant $e : \to \mathbf{P}$;
- the binary *alternative composition* operator $+ : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *sequential composition* operator $\cdot : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *parallel composition* operator $\parallel : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *left merge* operator $⌊\!⌊ : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- the binary *communication merge* operator $\mid : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
- for each $H \subseteq \mathsf{A}$, the unary *encapsulation* operator $\partial_H : \mathbf{P} \to \mathbf{P}$;
- the unary *atomic action* predicate symbol $\mathcal{A} : \mathbf{P}$.

We assume that there is a countably infinite set $\mathcal{X}$ of variables of sort $\mathbf{P}$, which contains x, y and z, with and without subscripts. Terms are built as usual. We use infix notation for the binary operators. We use the following precedence conventions to reduce the need for parentheses: the operator $+$ binds weaker than all other binary operators and the operator $\cdot$ binds stronger than all other binary operators.

Let P and Q be closed terms of sort $\mathbf{P}$. Intuitively, the constants, operators and predicate symbols introduced above can be explained as follows:

- δ is not capable of doing anything;
- e is only capable of performing atomic action e and next terminating successfully;
- $P + Q$ behaves either as P or as Q, but not both;
- $P \cdot Q$ first behaves as P and on successful termination of P it next behaves as Q;
- $P \parallel Q$ behaves as the process that proceeds with P and Q in parallel;
- $P \mathbin{\lfloor\!\lfloor} Q$ behaves the same as $P \parallel Q$, except that it starts with performing an atomic action of P;
- $P \mid Q$ behaves the same as $P \parallel Q$, except that it starts with performing an atomic action of P and an atomic action of Q synchronously;
- $\partial_H(P)$ behaves the same as P, except that atomic actions from H are blocked;
- $\mathcal{A}(P)$ holds if P is an atomic action.

The operators $\mathbin{\lfloor\!\lfloor}$ and $\mid$ are of an auxiliary nature. They are needed for the axiomatization of ACP process algebras.

The predicate symbol $\mathcal{A}$ is used to distinguish atomic actions from other processes. This predicate symbol, which does not occur in the axiom system ACP, obviates the need to have a constant for each atomic action. An alternative way to distinguish atomic actions from other processes is to have a subsort $\mathbf{A}$ of the sort $\mathbf{P}$. We have not chosen this alternative way because it complicates matters considerably. Moreover, we prefer to keep close to elementary algebraic specification (see e.g. [12]). By the notational convention introduced below, we seldom have to use the predicate symbol $\mathcal{A}$ explicitly.

In equations between terms of sort $\mathbf{P}$, we will use a notational convention which requires the following assumption: there is a countably infinite set $\mathcal{X}' \subseteq \mathcal{X}$ that contains a, b and c, with and without subscripts, but does not contain x, y and z, with and without subscripts. Let ϕ be an equation between terms of sort $\mathbf{P}$, and let $\{a_1, \ldots, a_n\}$ be the set of all variables from $\mathcal{X}'$ that occur in ϕ. Then we write ϕ for $\mathcal{A}(x_1) \wedge \ldots \wedge \mathcal{A}(x_n) \Rightarrow \phi'$, where ϕ' is ϕ with, for all $i \in [1, n]$, all occurrences of a_i replaced by x_i, and $x_1, \ldots, x_n$ are variables from $\mathcal{X}$ that do not occur in ϕ.

An *ACP process algebra* is an algebra with the signature of ACP process algebras that satisfies the formulas given in Table 3. Three formulas in this table are actually schemas of formulas: e is a syntactic variable which stands for an arbitrary constant of sort $\mathbf{P}$ (i.e. an atomic action constant or the deadlock constant). A side condition is added to two schemas to restrict the constants for which the syntactic variable stands.

Because the notational convention introduced above is used, the four equations in Table 3 that are actually conditional equations look the same as their counterpart in the axiom system ACP. It happens that these conditional equations allow for the generalization to meadow enriched ACP process algebras to proceed smoothly. Apart from this, the set of formulas given in Table 3 differs from the axiom system ACP on three points. Firstly, the equations $x \mid y = y \mid x$, $(x \mid y) \mid z = x \mid (y \mid z)$, and $\delta \mid x = \delta$ have been added. In the axiom system ACP,

Table 3. Axioms for ACP process algebras

$x + y = y + x$		$x \parallel y = (x \mathbin{\lfloor\!\lfloor} y + y \mathbin{\lfloor\!\lfloor} x) + x \mid y$
$(x + y) + z = x + (y + z)$		$a \mathbin{\lfloor\!\lfloor} x = a \cdot x$
$x + x = x$		$a \cdot x \mathbin{\lfloor\!\lfloor} y = a \cdot (x \parallel y)$
$(x + y) \cdot z = x \cdot z + y \cdot z$		$(x + y) \mathbin{\lfloor\!\lfloor} z = x \mathbin{\lfloor\!\lfloor} z + y \mathbin{\lfloor\!\lfloor} z$
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$		$a \mid b \cdot x = (a \mid b) \cdot x$
$x + \delta = x$		$a \cdot x \mid b \cdot y = (a \mid b) \cdot (x \parallel y)$
$\delta \cdot x = \delta$		$(x + y) \mid z = x \mid z + y \mid z$
		$x \mid y = y \mid x$
		$(x \mid y) \mid z = x \mid (y \mid z)$
$\partial_H(e) = e$	if $e \notin H$	$\delta \mid x = \delta$
$\partial_H(e) = \delta$	if $e \in H$	
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		$\mathcal{A}(e)$
$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$		$\mathcal{A}(x) \land \mathcal{A}(y) \Rightarrow \mathcal{A}(x \mid y)$

all closed substitution instances of these equations are derivable. Secondly, the equations $a \cdot x \mid b = (a \mid b) \cdot x$ and $x \mid (y + z) = x \mid y + x \mid z$ have been removed. These equations can be derived using the added equation $x|y = y|x$. Thirdly, the formulas $\mathcal{A}(e)$ and $\mathcal{A}(x) \land \mathcal{A}(y) \Rightarrow \mathcal{A}(x \mid y)$ have been added. They express that the processes denoted by constants of sort $\mathbf{P}$ are atomic actions and that the processes that result from the communication merge of two atomic actions are atomic actions. This does not exclude that there are additional atomic actions, which is impossible in the case of ACP.

For each model of the axiom system ACP given in [3], its expansion with the appropriate interpretation of the atomic action predicate symbol $\mathcal{A}$ is an ACP process algebra.

Not all processes in an ACP process algebra have to be interpretations of closed terms, even if all atomic actions are interpretations of closed terms. The processes concerned may be solutions of sets of recursion equations. It is recommendable to restrict the attention to ACP process algebras satisfying additional axioms by which sets of recursion equations that fulfil a guardedness condition have unique solutions. For a comprehensive treatment of this issue, the reader is referred to [3].

4 Meadow Enriched ACP Process Algebras

In this section, we introduce the notion of a meadow enriched ACP process algebra. This notion is a simple generalization of the notion of an ACP process algebra introduced in Section 3 to processes in which data are involved. The elements of a signed meadow are taken as data.

The signature of meadow enriched ACP process algebras is a two-sorted signature. It consists of the sorts, constants and operators from the signatures of ACP process algebras and signed meadows and in addition the following operators:

- for each $n \in \mathbb{N}$ and $e \in \mathsf{A}$, the n-ary *data handling atomic action* operator $e : \underbrace{\mathbf{Q} \times \cdots \times \mathbf{Q}}_{n \text{ times}} \to \mathbf{P}$;
- the binary *guarded command* operator $:\to \, : \mathbf{Q} \times \mathbf{P} \to \mathbf{P}$.

We take the variables in $\mathcal{U}$ for the variables of sort $\mathbf{Q}$ and the variables in $\mathcal{X}$ for the variables of sort $\mathbf{P}$. We assume that the sets $\mathcal{U}$ and $\mathcal{X}$ are disjoint. Terms are built as usual for a many-sorted signature (see e.g. [28,31]). We use the same notational conventions as before. In addition, we use infix notation for the binary operator $:\to$.

Let $p_1, \ldots, p_n$ and p be closed terms of sort $\mathbf{Q}$ and P be a closed term of sort $\mathbf{P}$. Intuitively, the additional operators can be explained as follows:

- $e(p_1, \ldots, p_n)$ is only capable of performing data handling atomic action $e(p_1, \ldots, p_n)$ and next terminating successfully;
- $p :\to P$ behaves as the process P if p equals 0 and is not capable of doing anything otherwise.

The different guarded command operators that have been proposed before in the setting of ACP have one thing in common: their first operand is considered to stand for an element of the domain of a Boolean algebra (see e.g. [8]). In contrast with those guarded command operators, the first operand of the guarded command operator introduced here is considered to stand for an element of the domain of a signed meadow.

A *meadow enriched ACP process algebra* is an algebra with the signature of meadow enriched ACP process algebras that satisfies the formulas given in Tables 1–4. Like in Table 3, some formulas in Table 4 are actually schemas of formulas: e, e' and e'' are syntactic variables which stand for arbitrary constants of sort $\mathbf{P}$ different from δ and, in addition, n and m stand for arbitrary natural numbers.

For meadow enriched ACP process algebras that satisfy the separation axiom and the cancellation axiom, the five equations concerning the guarded command operator on the left-hand side in the upper half of Table 4 can easily be understood by taking the view that 0 and 1 represent the Boolean values T and F, respectively. In that case, we have that

- p/p models the test that yields T if $p = 0$ and F otherwise;
- if both p and q are equal to 0 or 1, then $1 - p$ models $\neg p$, $p \cdot q$ models $p \vee q$, and consequently $1 - (1 - p) \cdot (1 - q)$ models $p \wedge q$.

From this view, the equations given in the upper half of Table 4 differ from the axioms for the most general kind of guarded command operator that has been proposed in the setting of ACP (see e.g. [8]) on two points only. Firstly, the equation $u :\to x = u/u :\to x$ has been added. This equation formalizes the informal explanation of the guarded command given above. Secondly, the

Table 4. Additional axioms for meadow enriched ACP process algebras

$0 :\rightarrow x = x$	$u :\rightarrow \delta = \delta$
$1 :\rightarrow x = \delta$	$u :\rightarrow (x + y) = u :\rightarrow x + u :\rightarrow y$
$u :\rightarrow x = (u/u) :\rightarrow x$	$u :\rightarrow x \cdot y = (u :\rightarrow x) \cdot y$
$u :\rightarrow (v :\rightarrow x) = (1 - (1 - u/u) \cdot (1 - v/v)) :\rightarrow x$	$(u :\rightarrow x) \mathbin{\lfloor\!\lfloor} y = u :\rightarrow (x \mathbin{\lfloor\!\lfloor} y)$
$u :\rightarrow x + v :\rightarrow x = (u/u \cdot v/v) :\rightarrow x$	$(u :\rightarrow x) \mid y = u :\rightarrow (x \mid y)$
	$\partial_H(u :\rightarrow x) = u :\rightarrow \partial_H(x)$
$e \mid e' = e'' \Rightarrow$ $e(u_1, \ldots, u_n) \mid e'(v_1, \ldots, v_n) = (u_1 - v_1) :\rightarrow (\cdots :\rightarrow ((u_n - v_n) :\rightarrow e''(u_1, \ldots, u_n)) \cdots)$	
$e \mid e' = \delta \Rightarrow e(u_1, \ldots, u_n) \mid e'(v_1, \ldots, v_n) = \delta$	
$e(u_1, \ldots, u_n) \mid e'(v_1, \ldots, v_m) = \delta$	if $n \neq m$
$\partial_H(e(u_1, \ldots, u_n)) = e(u_1, \ldots, u_n)$	if $e \notin H$
$\partial_H(e(u_1, \ldots, u_n)) = \delta$	if $e \in H$
$\mathcal{A}(e(u_1, \ldots, u_n))$	

equation $x \mid (u :\rightarrow y) = u :\rightarrow (x \mid y)$ has been removed. This equation can be derived using the equation $x \mid y = y \mid x$ from Table 3.

The equations in Table 4 concerning the communication merge of data handling atomic actions formalize the intuition that two data handling atomic actions $e(p_1, \ldots, p_n)$ and $e'(q_1, \ldots, q_m)$ can be performed synchronously iff e and e' can be performed synchronously and $n = m$ and $p_1 = q_1$ and $\ldots$ and $p_n = q_n$. The equations concerning the encapsulation of data handling atomic actions agree with the way in which the encapsulation of data handling atomic actions is dealt with in μCRL and PSF. The formula concerning the atomic action predicate simply expresses that data handling atomic actions are also atomic actions.

Henceforth, we will write $P \lhd p \rhd Q$ for $(p/p) :\rightarrow P + (1 - p/p) :\rightarrow Q$. For meadow enriched ACP process algebras that satisfy the separation axiom and the cancellation axiom, $P \lhd p \rhd Q$ can be read as follows: if p equals 0 then P else Q.

For each ACP process algebra $\mathfrak{A}'$ and each signed non-trivial cancellation meadow $\mathfrak{A}''$, there exists an amalgamation of $\mathfrak{A}'$ and $\mathfrak{A}''$, i.e. a model of the axioms for both ACP process algebras and signed non-trivial cancellation meadows whose restriction to the signature of ACP process algebras is $\mathfrak{A}'$ and whose restriction to the signature of signed meadows is $\mathfrak{A}''$ (by the amalgamation result about expansions presented as Theorem 6.1.1 in [19], adapted to the many-sorted case). For each amalgamation of an ACP process algebra with a countably infinite set of atomic actions and a signed non-trivial cancellation meadow, its expansion with the appropriate interpretation of the data handling atomic action operators e and the guarded command operator $:\rightarrow$ is a meadow enriched ACP process algebra.

In subsequent sections, we write Σ_{mp} for the signature of meadow enriched ACP process algebras.

5 A Calculus for Meadow Enriched ACP Process Algebras

In this section, we associate a calculus with meadow enriched ACP process algebras. For that, we introduce, for all associative operators from the signature of meadow enriched ACP process algebras that are not of an auxiliary nature, variable-binding operators as generalizations. To build terms of the calculus, called binding terms, both the constants and operators from the signature of meadow enriched ACP process algebras and those variable-binding operators are available.

The sets of *binding terms* of sorts $\mathbf{Q}$ and $\mathbf{P}$, written $\mathcal{BT}_{\mathbf{Q}}$ and $\mathcal{BT}_{\mathbf{P}}$, respectively, are inductively defined by the following formation rules (where $S_1, \dots, S_n$ and S range over the sorts from Σ_{mp}):

- if $u \in \mathcal{U}$, then $u \in \mathcal{BT}_{\mathbf{Q}}$;
- if $x \in \mathcal{X}$, then $x \in \mathcal{BT}_{\mathbf{P}}$;
- if $c : \to S$ is a constant from Σ_{mp}, then $c \in \mathcal{BT}_S$;
- if $o{:}S_1 \times \dots \times S_n \to S$ is an operator from Σ_{mp} and $t_1 \in \mathcal{BT}_{S_1}, \dots, t_n \in \mathcal{BT}_{S_n}$, then $o(t_1, \dots, t_n) \in \mathcal{BT}_S$;
- if $u \in \mathcal{U}$ and $t \in \mathcal{BT}_{\mathbf{Q}}$, then, for each $n \in \mathbb{N}^+$, $\sum_u^n t \in \mathcal{BT}_{\mathbf{Q}}$ and $\prod_u^n t \in \mathcal{BT}_{\mathbf{Q}}$;[1]
- if $u \in \mathcal{U}$ and $t \in \mathcal{BT}_{\mathbf{P}}$, then, for each $n \in \mathbb{N}^+$, $\mathop{+}\nolimits_u^n t \in \mathcal{BT}_{\mathbf{P}}$, $\mathop{\bullet}\nolimits_u^n t \in \mathcal{BT}_{\mathbf{P}}$, and $\|_u^n t \in \mathcal{BT}_{\mathbf{P}}$.

$\sum^n$, $\prod^n$, $+^n$, $\bullet^n$, and $\|^n$ are the variable-binding operators mentioned above. They bind variables that range over all quantities that can be denoted by numerals $\underline{k}$ where $0 \leq k < n$ (in plain terms, quantities that correspond to natural numbers less than n). Intuitively, $\sum_u^n t$ stands for $t_1 + \dots + t_n$, where t_i $(1 \leq i \leq n)$ is t with all occurrences of u replaced by $\underline{u-1}$, and analogously in the case of $\prod^n$, $+^n$, $\bullet^n$, and $\|^n$.

A binding term t is a *comprehended term* if it is a binding term of the form $\Diamond_u^n t'$, where $\Diamond^n$ is a variable-binding operator.[2] Below, we will give the axioms of the calculus associated with meadow enriched ACP process algebras. We have to do with a calculus because the distinction between free and bound variables is essential in applying the axioms concerning comprehended terms.

A variable $u \in \mathcal{U}$ occurs *free* in a binding term t if there is an occurrence of u in t that is not in a subterm of the form $\Diamond_u^n t'$, where $\Diamond^n$ is a variable-binding operator. A binding term t is *closed* if it is a binding term in which no variable occurs free.

Substitution of a binding term t' of sort $\mathbf{P}$ for a variable $x \in \mathcal{X}$ in a binding term t, written $t[t'/x]$, is defined by induction on the structure of t as usual:

[1] We write $\mathbb{N}^+$ for the set $\mathbb{N} \setminus \{0\}$.

[2] The name comprehended term originates from the name comprehended expression introduced in [27].

$$
\begin{aligned}
v[t'/x] &= v\ , \\
y[t'/x] &= \begin{cases} t' & \text{if } x \equiv y\ ,^{3} \\ y & \text{otherwise}\ , \end{cases} \\
c[t'/x] &= c\ , \\
o(t_1,\dots,t_n)[t'/x] &= o(t_1[t'/x],\dots,t_n[t'/x])\ , \\
(\Diamond_v^n\, t'')[t'/x] &= \begin{cases} \Diamond_w^n((t''[w/v])[t'/x]) & \text{if } v \text{ occurs free in } t' \\ & (w \text{ does not occur in } t', t'')\ , \\ \Diamond_v^n(t''[t'/x]) & \text{otherwise}\ . \end{cases}
\end{aligned}
$$

and substitution of a binding term t' of sort $\mathbf{Q}$ for a variable $u \in \mathcal{U}$ in a binding term t, written $t[t'/u]$, is defined by induction on the structure of t as follows:

$$
\begin{aligned}
v[t'/u] &= \begin{cases} t' & \text{if } u \equiv v\ , \\ v & \text{otherwise}\ , \end{cases} \\
x[t'/u] &= x\ , \\
c[t'/u] &= c\ , \\
o(t_1,\dots,t_n)[t'/u] &= o(t_1[t'/u],\dots,t_n[t'/u])\ , \\
(\Diamond_v^n\, t'')[t'/u] &= \begin{cases} \Diamond_v^n\, t'' & \text{if } u \equiv v\ , \\ \Diamond_w^n((t''[w/v])[t'/u]) & \text{if } u \not\equiv v, v \text{ occurs free in } t' \\ & (w \text{ does not occur in } t', t'')\ , \\ \Diamond_v^n(t''[t'/u]) & \text{otherwise}\ . \end{cases}
\end{aligned}
$$

The essentiality of the distinction between free and bound variables in applying the axioms concerning comprehended terms originates from the substitutions involved in applying those axioms.

The axioms of the calculus associated with meadow enriched ACP process algebras are the formulas given in Tables 1–5. Like some equations in Tables 3 and 4, the equations in Table 5 are actually schemas of equations: p and P are syntactic variables which stand for arbitrary binding terms of sort $\mathbf{Q}$ and sort $\mathbf{P}$, respectively, and n stands for an arbitrary positive natural number.

The axioms given in Table 5 are called the *axioms for comprehended terms*. They consist of three axioms, including an α-conversion axiom, for each of the variable-binding operators of the calculus. For each comprehended term, we can derive from these axioms that the comprehended term is equal to a term over the signature of meadow enriched ACP process algebras.

[3] We write $\equiv$ for syntactic identity.

Table 5. Axioms for comprehended terms

$\sum_u^n p = \sum_v^n (p[v/u])$	if v does not occur free in p
$\sum_u^1 p = p[0/u]$	
$\sum_u^{n+1} p = p[0/u] + \sum_u^n (p[u+1/u])$	
$\prod_u^n p = \prod_v^n (p[v/u])$	if v does not occur free in p
$\prod_u^1 p = p[0/u]$	
$\prod_u^{n+1} p = p[0/u] \cdot \prod_u^n (p[u+1/u])$	
$+_u^n P = +_v^n (P[v/u])$	if v does not occur free in P
$+_u^1 P = P[0/u]$	
$+_u^{n+1} P = P[0/u] + +_u^n (P[u+1/u])$	
$\bullet_u^n P = \bullet_v^n (P[v/u])$	if v does not occur free in P
$\bullet_u^1 P = P[0/u]$	
$\bullet_u^{n+1} P = P[0/u] \cdot \bullet_u^n (P[u+1/u])$	
$\|_u^n P = \|_v^n (P[v/u])$	if v does not occur free in P
$\|_u^1 P = P[0/u]$	
$\|_u^{n+1} P = P[0/u] \parallel \|_u^n (P[u+1/u])$	

Theorem 1 (Elimination). *For all comprehended terms t, there exists a term t' over the signature of meadow enriched* ACP *process algebras such that $t = t'$ is derivable from the axioms for comprehended terms.*

Proof. If t is of the form $\sum_u^n t''$, $\prod_u^n t''$, $+_u^n t''$, $\bullet_u^n t''$ or $\|_u^n t''$, where t'' is a term over the signature of meadow enriched ACP process algebras of the right sort, then it is easy to prove by induction on n that there exists a term t' over the signature of meadow enriched ACP process algebras such that $t = t'$ is derivable from the axioms for comprehended terms. Using this fact, the general case is easily proved by induction on the depth of t. □

The comprehended terms of the calculus associated with meadow enriched ACP process algebras are *finitary* comprehended terms because the variable-binding operators of the calculus bind variables with a finite range only. This is a prerequisite for elimination of variable-binding operators.

6 The Interpretation of Terms of the Calculus

In this section, we define the interpretation of terms of the calculus associated with meadow enriched ACP process algebras. We assume that a fixed but arbitrary meadow enriched ACP process algebra $\mathfrak{A}$ has been given.

We write $\sigma_{\mathfrak{A}}$, where σ in Σ_{mp}, for the interpretation of σ in $\mathfrak{A}$. Moreover, we write $f+1$, where $f : \mathbf{Q}_{\mathfrak{A}} \to \mathbf{Q}_{\mathfrak{A}}$ or $f : \mathbf{Q}_{\mathfrak{A}} \to \mathbf{P}_{\mathfrak{A}}$, for the function $f' : \mathbf{Q}_{\mathfrak{A}} \to \mathbf{Q}_{\mathfrak{A}}$ or $f' : \mathbf{Q}_{\mathfrak{A}} \to \mathbf{P}_{\mathfrak{A}}$, respectively, defined by $f'(q) = f(q +_{\mathfrak{A}} 1_{\mathfrak{A}})$.

The terms of the calculus introduced above can be directly interpreted in $\mathfrak{A}$. To achieve that, we associate with each variable-binding operator $\Diamond^n$ of the calculus a function $\Diamond^n_{\mathfrak{A}} : (\mathbf{Q}_{\mathfrak{A}} \to \mathbf{Q}_{\mathfrak{A}}) \to \mathbf{Q}_{\mathfrak{A}}$ or $\Diamond^n_{\mathfrak{A}} : (\mathbf{Q}_{\mathfrak{A}} \to \mathbf{P}_{\mathfrak{A}}) \to \mathbf{P}_{\mathfrak{A}}$ as follows:

$$
\begin{array}{llll}
\sum^1_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \;, & \textstyle+^1_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \;, \\
\sum^{n+1}_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) +_{\mathfrak{A}} \sum^n_{\mathfrak{A}}(f+1) \;, & \textstyle+^{n+1}_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) +_{\mathfrak{A}} \textstyle+^n_{\mathfrak{A}}(f+1) \;, \\
\prod^1_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \;, & \bullet^1_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \;, \\
\prod^{n+1}_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \cdot_{\mathfrak{A}} \prod^n_{\mathfrak{A}}(f+1) \;, & \bullet^{n+1}_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \cdot_{\mathfrak{A}} \bullet^n_{\mathfrak{A}}(f+1) \;, \\
 & & \|^1_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \;, \\
 & & \|^{n+1}_{\mathfrak{A}}(f) & = f(0_{\mathfrak{A}}) \parallel_{\mathfrak{A}} \|^n_{\mathfrak{A}}(f+1) \;.
\end{array}
$$

The interpretation of a term of the calculus in $\mathfrak{A}$ depends on the elements of $\mathbf{Q}_{\mathfrak{A}}$ and $\mathbf{P}_{\mathfrak{A}}$ that are associated with the variables that occur free in it. We model such associations by functions $\rho : (\mathcal{U} \cup \mathcal{X}) \to (\mathbf{Q}_{\mathfrak{A}} \cup \mathbf{P}_{\mathfrak{A}})$ such that $u \in \mathcal{U} \Rightarrow \rho(u) \in \mathbf{Q}_{\mathfrak{A}}$ and $x \in \mathcal{X} \Rightarrow \rho(x) \in \mathbf{P}_{\mathfrak{A}}$. These functions are called *assignments* in $\mathfrak{A}$. We write $\mathcal{A}ss_{\mathfrak{A}}$ for the set of all assignments in $\mathfrak{A}$. For each assignment $\rho \in \mathcal{A}ss_{\mathfrak{A}}$, $u \in \mathcal{U}$ and $q \in \mathbf{Q}_{\mathfrak{A}}$, we write $\rho(u \to q)$ for the unique assignment $\rho' \in \mathcal{A}ss_{\mathfrak{A}}$ such that $\rho'(v) = \rho(v)$ if $v \not\equiv u$ and $\rho'(u) = q$.

The interpretation of terms of the calculus in a meadow enriched ACP process algebra $\mathfrak{A}$ is given by the function $[\![\_]\!]_{\mathfrak{A}} : (\mathcal{BT}_{\mathbf{Q}} \cup \mathcal{BT}_{\mathbf{P}}) \to (\mathcal{A}ss_{\mathfrak{A}} \to (\mathbf{Q}_{\mathfrak{A}} \cup \mathbf{P}_{\mathfrak{A}}))$ defined as follows:

$$
\begin{array}{ll}
[\![u]\!]_{\mathfrak{A}}(\rho) & = \rho(u) \;, \\
[\![x]\!]_{\mathfrak{A}}(\rho) & = \rho(x) \;, \\
[\![c]\!]_{\mathfrak{A}}(\rho) & = c_{\mathfrak{A}} \;, \\
[\![o(t_1, \ldots, t_n)]\!]_{\mathfrak{A}}(\rho) & = o_{\mathfrak{A}}([\![t_1]\!]_{\mathfrak{A}}(\rho), \ldots, [\![t_n]\!]_{\mathfrak{A}}(\rho)) \;, \\
[\![\Diamond^n_u t]\!]_{\mathfrak{A}}(\rho) & = \Diamond^n_{\mathfrak{A}}(f), \text{ where } f \text{ is defined by } f(q) = [\![t]\!]_{\mathfrak{A}}(\rho(u \to q)) \;.
\end{array}
$$

The axioms of the calculus associated with meadow enriched ACP process algebras are sound with respect to the interpretation of the terms of the calculus given above.

Theorem 2 (Soundness). *For all equations $t = t'$ that belong to the axioms of the calculus associated with meadow enriched* ACP *process algebras, we have that $[\![t]\!]_{\mathfrak{A}}(\rho) = [\![t']\!]_{\mathfrak{A}}(\rho)$ for all assignments $\rho \in \mathcal{A}ss_{\mathfrak{A}}$.*

Proof. For all equations $t = t'$ that belong to the axioms for meadow enriched ACP process algebras, the soundness follows immediately from the fact that $\mathfrak{A}$ is a meadow enriched ACP process algebra. For all equations $t = t'$ that belong to the axioms for comprehended terms, the soundness is easily proved by induction on the structure of t. $\square$

Because the terms of the calculus associated with meadow enriched ACP process algebras can be directly interpreted in meadow enriched ACP process algebras, we consider the variable-binding operators of the calculus to constitute a process algebraic feature. Fitting them in an algebraic framework does not involve any serious theoretical complication. It is much more difficult to fit the variable-binding operators from μCRL and PSF that generalize associative operators of ACP, but do not give rise to finitary comprehended terms, in an algebraic framework (see e.g. [22]).

7 The Binary Variable-Binding Operators

Full elimination of all variable-binding operators occurring in a comprehended term can lead to a combinatorial explosion. In this section, we show that no combinatorial explosion takes place if variable-binding operators that bind variables with a two-valued range are still permitted in the resulting term.

We begin by looking at an example. From the axioms for comprehended terms, we easily derive the equation

$$\sum_u^7 p = p[\underline{0}/u] + \cdots + p[\underline{6}/u] \; .$$

This suggests that, on full elimination of variable-binding operators, the size of the resulting term grows rapidly as the size of the original term increases (there are seven substitution instances of p and they have increasing sizes). Using the axioms for comprehended terms as well as other axioms of the calculus, we derive the following:

$$\begin{aligned}
& p[\underline{0}/u] + \cdots + p[\underline{6}/u] \\
&= p[\underline{0}/u] + \cdots + p[\underline{6}/u] + 0 \\
&= (0 \lhd 1 - \mathsf{s}(u - \underline{6}) \rhd p)[\underline{0}/u] + \cdots + (0 \lhd 1 - \mathsf{s}(u - \underline{6}) \rhd p)[\underline{7}/u] \\
&= \textstyle\sum_u^2 \big(\sum_v^2 \big(\sum_w^2 \big((0 \lhd 1 - \mathsf{s}(u - \underline{6}) \rhd p)[\underline{2}^2 \cdot w + \underline{2}^1 \cdot v + \underline{2}^0 \cdot u/u]\big)\big)\big) \\
&= \textstyle\sum_u^2 \big(\sum_v^2 \big(\sum_w^2 \big(((0 \lhd 1 - \mathsf{s}(u - \underline{6}) \rhd p)[\underline{2} \cdot v + u/u])[\underline{2} \cdot w + v/v]\big)\big)\big) \; .
\end{aligned}$$

This suggests that, if variable-binding operators that bind variables with a two-valued range are still permitted in the resulting term, its size grows far less rapidly as the size of the original term increases (there is only one substitution instance of p). However, a counterpart of the first step in the derivation above does not exist for comprehended terms of the forms $\bullet_u^n p$ and $\|_u^n p$ because identity elements for sequential and parallel composition are missing.

Henceforth, we will use the term *binary variable-binding operators* for the variable-binding operators that bind variables with a two-valued range and the term *non-binary variable-binding operators* for the other variable-binding operators.

The size of binding terms is given by the function $\mathit{size} : (\mathcal{BT}_{\mathbf{Q}} \cup \mathcal{BT}_{\mathbf{P}}) \to \mathbb{N}$ defined as follows:

$$
\begin{aligned}
&\mathit{size}(u) &&= 1\ , \\
&\mathit{size}(x) &&= 1\ , \\
&\mathit{size}(c) &&= 1\ , \\
&\mathit{size}(o(t_1, \ldots, t_n)) &&= \mathit{size}(t_1) + \cdots + \mathit{size}(t_n) + 1\ , \\
&\mathit{size}\big(\textstyle\bigdiamond_u^n(t)\big) &&= \mathit{size}(t) + \log_2(n) + 1\ .^4
\end{aligned}
$$

The summand $\log_2(n)$ occurs in the equation for the size of a term of the form $\bigdiamond_u^n(t)$ because having (the cardinality of) the range of u encoded in the variable-binding operator is an artifice that must be taken into account using the most efficient way in which $\underline{n}$ could be represented by a binding term. It follows from Proposition 1 formulated below that the size of this term is of order $\log_2(n)$.

The important insights relevant to elimination of non-binary variable-binding operators are brought together in the following proposition.

Proposition 1. *From the axioms of the calculus associated with meadow enriched* ACP *process algebras, we can derive the equations from Table 6 for each binding term p of sort $\mathbf{Q}$, binding term P of sort $\mathbf{P}$, and $n, m \in \mathbb{N}^+$.*

Proof. It follows immediately from the axioms for comprehended terms that the first two equations for $\sum^n$ are derivable. It is easy to prove by induction on n that

$$\textstyle\sum_u^{2 \cdot n} p = \sum_u^n (p[2 \cdot u/u]) + \sum_u^n (p[2 \cdot u + 1/u])$$

is derivable. From this it follows easily that the third equation for $\sum^n$ is derivable. It is easy to prove by case distinction between $n = 1$ and $n > 1$ that

$$\textstyle\sum_u^n (0 \lhd 1 - \mathsf{s}(u - \underline{0}) \rhd p) = p[0/u]$$

is derivable. Using this fact, it is easy to prove by induction on n that for all $m \geq n + 1$:

$$\textstyle\sum_u^{n+1} p = \sum_u^m (0 \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd p)$$

is derivable. From this it follows easily that the fourth equation for $\sum^n$ is derivable. The proofs for the equations for $\prod^n$, $+^n$, $\bullet^n$ and $\|^n$ go analogously, with the exception of the fourth equation for $\bullet^n$ and $\|^n$. It is easy to prove by induction on n that for all $m < n$:

$$\bullet_u^n P = \bullet_u^m P \cdot \bullet_u^{n-m} (P[m + u/u])$$

[4] We use the convention that, whenever we write $\log_2(n)$ in a context requiring a natural number, $\lceil \log_2(n) \rceil$ is implicitly meant.

Table 6. Derived equations for comprehended terms

$$
\begin{array}{ll}
\sum\nolimits_u^1 p = p[0/u] & \\
\sum\nolimits_u^2 p = p[0/u] + p[1/u] & \\
\sum\nolimits_u^{2^{n+1}} p = \sum\nolimits_u^2 \left(\sum\nolimits_v^{2^n} (p[\underline{2} \cdot v + u/u]) \right) & \\
\sum\nolimits_u^{n+1} p = \sum\nolimits_u^{2^m} (0 \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd p) & \text{if } n+1 \leq 2^m \\
 & \\
\prod\nolimits_u^1 p = p[0/u] & \\
\prod\nolimits_u^2 p = p[0/u] \cdot p[1/u] & \\
\prod\nolimits_u^{2^{n+1}} p = \prod\nolimits_u^2 \left(\prod\nolimits_v^{2^n} (p[\underline{2} \cdot v + u/u]) \right) & \\
\prod\nolimits_u^{n+1} p = \prod\nolimits_u^{2^m} (1 \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd p) & \text{if } n+1 \leq 2^m \\
 & \\
\mathop{+}\nolimits_u^1 P = P[0/u] & \\
\mathop{+}\nolimits_u^2 P = P[0/u] + P[1/u] & \\
\mathop{+}\nolimits_u^{2^{n+1}} P = \mathop{+}\nolimits_u^2 \left(\mathop{+}\nolimits_v^{2^n} (P[\underline{2} \cdot v + u/u]) \right) & \\
\mathop{+}\nolimits_u^{n+1} P = \mathop{+}\nolimits_u^{2^m} (\delta \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd P) & \text{if } n+1 \leq 2^m \\
 & \\
\mathop{\bullet}\nolimits_u^1 P = P[0/u] & \\
\mathop{\bullet}\nolimits_u^2 P = P[0/u] \cdot P[1/u] & \\
\mathop{\bullet}\nolimits_u^{2^{n+1}} P = \mathop{\bullet}\nolimits_u^2 \left(\mathop{\bullet}\nolimits_v^{2^n} (P[\underline{2} \cdot v + u/u]) \right) & \\
\mathop{\bullet}\nolimits_u^{n+1} P = \mathop{\bullet}\nolimits_u^{2^m} P \cdot \mathop{\bullet}\nolimits_u^{(n+1)-2^m} (P[\underline{2}^m + u/u]) & \text{if } 2^m < n+1 < 2^{m+1} \\
 & \\
\mathop{\|}\nolimits_u^1 P = P[0/u] & \\
\mathop{\|}\nolimits_u^2 P = P[0/u] \parallel P[1/u] & \\
\mathop{\|}\nolimits_u^{2^{n+1}} P = \mathop{\|}\nolimits_u^2 \left(\mathop{\|}\nolimits_v^{2^n} (P[\underline{2} \cdot v + u/u]) \right) & \\
\mathop{\|}\nolimits_u^{n+1} P = \mathop{\|}\nolimits_u^{2^m} P \parallel \mathop{\|}\nolimits_u^{(n+1)-2^m} (P[\underline{2}^m + u/u]) & \text{if } 2^m < n+1 < 2^{m+1}
\end{array}
$$

is derivable. From this it follows easily that the fourth equation for $\bullet^n$ is derivable. The proof for the fourth equation for $\|^n$ goes analogously. □

The axioms for comprehended terms give rise to a corollary about full elimination of all variable-binding operators.

Corollary 1. *Let t be a comprehended term without comprehended terms as proper subterms, and let $k = size(t)$. Then there exists a term t' over the signature of meadow enriched* ACP *process algebras such that $t = t'$ is derivable from the axioms of the calculus associated with meadow enriched* ACP *process algebras and*

– *$size(t') = O(k^2 \cdot 2^k)$;*

- *$size(t') = \Omega(k \cdot 2^{k-2})$ if t is a term of the form $\sum_u^n t''$ or $\prod_u^n t''$ and the number of times that u occurs free in t'' is greater than zero;*
- *$size(t') = \Omega(k \cdot 2^{k-3})$ if t is a term of the form $\mathop{+}_u^n t''$, $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$ and the number of times that u occurs free in t'' is greater than zero.*

Proof. Term t is a binding term of the form $\mathop{\Diamond}_u^n t''$, where $\Diamond^n$ is a variable-binding operator. Let $k' = size(t'')$, let k'' be the number of times that u occurs free in t'', and let l_i $(0 \leq i < n)$ be the size of the smallest term p over the signature of meadow enriched ACP process algebras such that $p = \underline{i}$. Then $size(t') = n \cdot k' + \sum_{i=0}^{n-1}(k'' \cdot l_i) + n - 1$. Because $k = k' + \log_2(n) + 1$, we know that $k' < k$, $\log_2(n) < k$ and $n < 2^k$. Moreover, we know that $k'' < k'$ and $l_i = \Theta(\log_2(i+1))$. Hence $size(t') = O(k^2 \cdot 2^k)$. We also know that $k' \geq 1$ and, because $k = k' + \log_2(n) + 1$, $\log_2(n) \geq k - 2$ and $n \geq 2^{k-2}$ if t is of the form $\sum_u^n t''$ or $\prod_u^n t''$; and that $k' \geq 2$ and, because $k = k' + \log_2(n) + 1$, $\log_2(n) \geq k - 3$ and $n \geq 2^{k-3}$ if t is of the form $\mathop{+}_u^n t''$, $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$. Hence, in the case where $k'' \geq 1$, $size(t') = \Omega(k \cdot 2^{k-2})$ if t is of the form $\sum_u^n t''$ or $\prod_u^n t''$ and $size(t') = \Omega(k \cdot 2^{k-3})$ if t is of the form $\mathop{+}_u^n t''$, $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$. $\square$

Proposition 1 gives rise to a corollary about full elimination of all non-binary variable-binding operators.

Corollary 2. *Let t be a comprehended term without comprehended terms as proper subterms, and let $k = size(t)$. Then there exists a binding term t' without non-binary variable-binding operators such that $t = t'$ is derivable from the axioms of the calculus associated with meadow enriched* ACP *process algebras and*

- *$size(t') = O(k^3)$ if t is a term of the form $\sum_u^n t''$, $\prod_u^n t''$ or $\mathop{+}_u^n t''$;*
- *$size(t') = \Omega(k^2)$ if t is a term of the form $\sum_u^n t''$, $\prod_u^n t''$ or $\mathop{+}_u^n t''$;*
- *$size(t') = O(k^4)$ if t is a term of the form $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$;*
- *$size(t') = \Omega(k^3)$ if t is a term of the form $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$ and the number of times that u occurs free in t'' is greater than zero.*

Proof. Firstly, we consider the case where t is a term of the form $\sum_u^n t''$, $\prod_u^n t''$ or $\mathop{+}_u^n t''$. Let $k' = size(t'')$, let k'' be the number of times that u occurs free in t'', and let l'_n be the size of the smallest term p over the signature of meadow enriched ACP process algebras such that $p = 1 - \mathsf{s}(u - \underline{n})$. Then $size(t') = k' + \sum_{i=0}^{\log_2(n)}(k'' \cdot (6 \cdot i)) + \log_2(n) \cdot (\log_2(n)+1) + 4 \cdot l'_n + 6$. Because $k = k' + \log_2(n) + 1$, we know that $k' < k$ and $\log_2(n) < k$. Moreover, we know that $k'' < k'$ and $l'_n = \Theta(\log_2(n+1))$. Hence $size(t') = O(k^3)$. We also know that $k' \geq 1$ and, because $k = k' + \log_2(n) + 1$, $\log_2(n) \geq k - 2$ if t is of the form $\sum_u^n t''$ or $\prod_u^n t''$; and that $k' \geq 2$ and, because $k = k' + \log_2(n) + 1$, $\log_2(n) \geq k - 3$ if t is of the form $\mathop{+}_u^n t''$. Hence, $size(t') = \Omega(k^2)$.

Secondly, we consider the case where t is a term of the form $\mathop{\bullet}_u^n t''$ or $\mathop{\|}_u^n t''$. Let $k' = size(t'')$, and let k'' be the number of times that u occurs free in t''. Then $size(t') \leq \sum_{i=0}^{\log_2(n)}(k' + \sum_{j=0}^{\log_2(i)}(k'' \cdot (6 \cdot j)) + \log_2(i) \cdot (\log_2(i) + 1))$. Because $k = k' + \log_2(n) + 1$, we know that $k' < k$ and $\log_2(n) < k$. Moreover, we know that $k'' < k'$. Hence $\mathit{size}(t') = O(k^4)$. We also have that $\mathit{size}(t') \geq k' + \sum_{i=0}^{\log_2(n)} (k'' \cdot (6 \cdot i)) + \log_2(n) \cdot (\log_2(n) + 1)$. Because $k = k' + \log_2(n) + 1$ and $k' \geq 2$, we also know that $\log_2(n) \geq k - 3$. Hence, in the case where $k'' \geq 1$, $\mathit{size}(t') = \Omega(k^3)$. □

Corollaries 1 and 2 show that much of the compactness that can be achieved with the variable-binding operators of the calculus associated with meadow enriched ACP process algebras can already be achieved with the binary variable-binding operators.

In Corollary 2, $\mathit{size}(t')$ is $O(k^4)$ instead of $O(k^3)$ if t is of the form $\bullet_u^n \, t''$ or $\|_u^n \, t''$. The origin of this is that ACP process algebras do not have identity elements for sequential and parallel composition. In the setting of ACP, the identity element for sequential composition, as well as parallel composition, is known as the empty process.

8 Adding an Identity Element for Sequential Composition

In this section, we investigate the effect of adding an identity element for sequential composition to ACP process algebras on the result concerning elimination of non-binary variable-binding operators presented above.

The signature of these algebras is the signature of ACP process algebras extended with the following:

- the *empty process* constant $\epsilon : \to \mathbf{P}$;
- the unary *termination* operator $\surd : \mathbf{P} \to \mathbf{P}$.

Let P be a closed term of sort $\mathbf{P}$. Intuitively, the additional constant and operator can be explained as follows:

- ϵ is only capable of terminating successfully;
- $\surd(P)$ is only capable of terminating successfully if P is capable of terminating successfully and is not capable of doing anything otherwise.

In the setting of ACP, the addition of the empty process constant has been treated in several ways. The treatment in [21] yields a non-associative parallel composition operator. The first treatment that yields an associative parallel composition operator [30] is from 1986, but was not published until 1997. The treatment in this paper is based on [1].

An *ACP process algebra with an identity element for sequential composition* is an algebra with the signature of ACP process algebras with an identity element for sequential composition that satisfies the formulas given in Table 3 with the exception of $x \parallel y = (x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y$ and the formulas given in Table 7.

We could dispense with the equations $a \mathbin{⌊\!⌊} x = a \cdot x$ and $a \mid b \cdot x = (a \mid b) \cdot x$ from Table 3 because they have become derivable from the other equations. In spite of the replacement of the equation $x \parallel y = (x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y$ by the equation $x \parallel y = ((x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y) + \surd(x) \cdot \surd(y)$, the equations characterizing ACP process algebras with an identity element for sequential composition constitute

Table 7. Replacing and additional axioms for empty process constant

$x \cdot \epsilon = x$	$\surd(\epsilon) = \epsilon$
$\epsilon \cdot x = x$	$\surd(a) = \delta$
$x \parallel y = ((x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y) + \surd(x) \cdot \surd(y)$	$\surd(x + y) = \surd(x) + \surd(y)$
$x \mathbin{⌊\!⌊} \epsilon = x$	$\surd(x \cdot y) = \surd(x) \cdot \surd(y)$
$\epsilon \mathbin{⌊\!⌊} x = \delta$	$\surd(x) \cdot \surd(y) = \surd(y) \cdot \surd(x)$
$\epsilon \mid x = \delta$	$x + \surd(x) = x$
$\partial_H(\epsilon) = \epsilon$	

a conservative extension of the equations characterizing ACP process algebras. The equation $\surd(x) \cdot \surd(y) = \surd(y) \cdot \surd(x)$ is of importance because it makes the equation $(x \parallel y) \parallel z = x \parallel (y \parallel z)$ derivable. The equation $x + \surd(x) = x$ is of importance because it makes the equation $x \parallel \epsilon = x$ derivable.

Meadow enriched ACP process algebras with an identity element for sequential composition are defined like meadow enriched ACP process algebras. We can associate a calculus with meadow enriched ACP process algebras with an identity element for sequential composition like we did before for meadow enriched ACP process algebras.

By the addition of an identity element for sequential composition, the properties of $\bullet^n$ and $\|^n$ with respect to elimination of non-binary variable-binding operators become comparable to the properties of $\sum^n$, $\prod^n$ and $+\!\!\!\!+^n$ with respect to elimination of non-binary variable-binding operators.

Proposition 2. *From the axioms of the above-mentioned calculus, we can derive the following equations for each binding term P of sort $\mathbf{P}$ and $n, m \in \mathbb{N}^+$:*

$$\bullet_u^{n+1} P = \bullet_u^{2^m} (\epsilon \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd P) \quad \text{if } n + 1 \leq 2^m ,$$
$$\|_u^{n+1} P = \|_u^{2^m} (\epsilon \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd P) \quad \text{if } n + 1 \leq 2^m .$$

Proof. The proofs for these equations go analogously to the proofs for the last equations for $\sum^n$, $\prod^n$ and $+\!\!\!\!+^n$ in the proof of Proposition 1. □

Proposition 2 gives rise to a corollary about full elimination of the non-binary variable-binding operators for sequential and parallel composition in the presence of an identity element for sequential composition.

Corollary 3. *Let t be a comprehended term of the form $\bullet_u^n t''$ or $\|_u^n t''$ without comprehended terms as proper subterms, and let $k = size(t)$. Then there exists a binding term t' without non-binary variable-binding operators such that $t = t'$ is derivable from the axioms of the above-mentioned calculus and $size(t') = O(k^3)$ and $size(t') = \Omega(k^2)$.*

Proof. The proof goes analogously to the case where t is of the form $\sum_u^n t''$, $\prod_u^n t''$ or $+\!\!\!\!+_u^n t''$ in the proof of Corollary 2. □

Corollaries 2 and 3 imply that, on full elimination of the non-binary variable-binding operators for sequential and parallel composition, the addition of an identity element for sequential composition to ACP process algebras gives rise to polynomially smaller terms.

9 Adding Process Sequences

In this section, we introduce process sequences to demonstrate that there is an alternative to introducing variable-binding operators for several associative operators on processes.

The signature of ACP process algebras with an identity element for sequential composition and process sequences is the signature of ACP process algebras with an identity element for sequential composition extended with the sort $\mathbf{PS}$ of *process sequences* and the following constants and operators:

- the *empty process sequence* constant $\langle\,\rangle : \to \mathbf{PS}$;
- the unary *singleton process sequence* operator $\langle \_ \rangle : \mathbf{P} \to \mathbf{PS}$;
- the binary *process sequence concatenation* operator $\frown : \mathbf{PS} \times \mathbf{PS} \to \mathbf{PS}$;
- the unary *generalized alternative composition* operator $\sum : \mathbf{PS} \to \mathbf{P}$;
- the unary *generalized sequential composition* operator $\bullet : \mathbf{PS} \to \mathbf{P}$;
- the unary *generalized parallel composition* operator $\| : \mathbf{PS} \to \mathbf{P}$.

We assume that there is a countably infinite set $\mathcal{V}$ of variables of sort $\mathbf{PS}$, which contains α, β and γ, with and without subscripts. We use the same notational conventions as before. In addition, we use infix notation for the binary operator $\frown$ and mixfix notation for the unary operator $\langle \_ \rangle$.

The constant and the first two operators introduced above are the usual ones for sequences, which gives an appropriate intuition about them. The remaining three operators introduced above generalize alternative, sequential and parallel composition to an arbitrary finite number of processes.

An *ACP process algebra with an identity element for sequential composition and process sequences* is an algebra with the signature of ACP process algebras with an identity element for sequential composition and process sequences that satisfies the formulas given in Table 3 with the exception of $x \parallel y = (x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y$ and the formulas given in Tables 7 and 8.

If we would introduce process sequences in the absence of an identity element for sequential composition, we should consider non-empty process sequences only.

Meadow enriched ACP process algebras with an identity element for sequential composition and process sequences are defined like meadow enriched ACP process algebras. We can associate a calculus with meadow enriched ACP process algebras with an identity element for sequential composition and process sequences like we did before for meadow enriched ACP process algebras. Moreover, we can extend the resulting calculus with variable-binding operators that generalize the process sequence concatenation operator. For the terms of the extended calculus, we need the following additional formation rule:

- if $u \in \mathcal{U}$ and $t \in \mathcal{BT}_{\mathbf{PS}}$, then, for each $n \in \mathbb{N}^+$, $\frown_u^n t \in \mathcal{BT}_{\mathbf{PS}}$.

Table 8. Additional axioms for process sequences

$\alpha \frown \langle\,\rangle = \alpha$	$\bullet(\langle\,\rangle) = \epsilon$
$\langle\,\rangle \frown \alpha = \alpha$	$\bullet(\langle x\rangle) = x$
$(\alpha \frown \beta) \frown \gamma = \alpha \frown (\beta \frown \gamma)$	$\bullet(\langle x\rangle \frown \alpha) = x \cdot \bullet(\alpha)$
$\mathop{\Large +}(\langle\,\rangle) = \delta$	$\mathop{\Large \|}(\langle\,\rangle) = \epsilon$
$\mathop{\Large +}(\langle x\rangle) = x$	$\mathop{\Large \|}(\langle x\rangle) = x$
$\mathop{\Large +}(\langle x\rangle \frown \alpha) = x + \mathop{\Large +}(\alpha)$	$\mathop{\Large \|}(\langle x\rangle \frown \alpha) = x \parallel \mathop{\Large \|}(\alpha)$

Table 9. Additional axioms for comprehended terms of sort **PS**

$\curvearrowright_u^n S = \curvearrowright_v^n (S[v/u])$
$\curvearrowright_u^1 S = S[0/u]$
$\curvearrowright_u^{n+1} S = S[0/u] \frown \curvearrowright_u^n (S[u+1/u])$

The axioms of the extended calculus are the formulas given in Tables 1–5 and 7–9. Like some equations in Tables 3–5, the equations in Table 9 are actually schemas of equations: S is a syntactic variable which stands for an arbitrary binding term of sort **PS**, and n stands for an arbitrary positive natural number.

The properties of $\curvearrowright^n$ with respect to elimination of non-binary variable-binding operators are comparable to the properties of $\mathop{\Large +}^n$, $\bullet^n$ and $\mathop{\Large \|}^n$ with respect to elimination of non-binary variable-binding operators.

Proposition 3. *From the axioms of the extended calculus, we can derive the following equations for each binding term S of sort* **PS** *and $n, m \in \mathbb{N}^+$:*

$$\curvearrowright_u^1 S = S[0/u]\ ,$$
$$\curvearrowright_u^2 S = S[0/u] \frown S[1/u]\ ,$$
$$\curvearrowright_u^{2^{n+1}} S = \curvearrowright_u^2 \left(\curvearrowright_v^{2^n} (S[\underline{2} \cdot v + u/u])\right)\ ,$$
$$\curvearrowright_u^{n+1} S = \curvearrowright_u^{2^m} (\langle\,\rangle \lhd 1 - \mathsf{s}(u - \underline{n}) \rhd S) \quad \text{if } n+1 \leq 2^m\ .$$

Proof. The proof goes analogously to the case of the equations for $\sum^n$ in the proof of Proposition 1. □

Proposition 3 gives rise to a corollary about full elimination of the non-binary variable-binding operators for process sequence concatenation.

Corollary 4. *Let t be a comprehended term of the form $\curvearrowright_u^n t''$ without comprehended terms as proper subterms, and let $k = size(t)$. Then there exists a binding term t' without non-binary variable-binding operators such that $t = t'$ is derivable from the axioms of the extended calculus and $size(t') = O(k^3)$ and $size(t') = \Omega(k^2)$.*

Proof. The proof goes analogously to the case where t is of the form $\sum_u^n t''$, $\prod_u^n t''$ or $\mathop{\Large +}_u^n t''$ in the proof of Corollary 2. □

In the presence of the operators $+$, $\cdot$ and $\|$ and the variable-binding operator $\curvearrowright^n$, the variable-binding operators $+^n$, $\cdot^n$, and $\|^n$ are superfluous.

Proposition 4. *From the axioms of the extended calculus, we can derive the following equations for each binding term P of sort $\mathbf{P}$ and $n \in \mathbb{N}^+$:*

$$+^n_u P = +(\curvearrowright^n_u \langle P \rangle)\ , \qquad \cdot^n_u P = \cdot(\curvearrowright^n_u \langle P \rangle)\ , \qquad \|^n_u P = \|(\curvearrowright^n_u \langle P \rangle)\ .$$

Proof. This is easy to prove by induction on n. □

If we would introduce quantity sequences as well, we could get a similar result for the variable-binding operators $\sum^n$ and $\prod^n$.

Proposition 4 shows that there is an alternative to introducing variable-binding operators for alternative, sequential and parallel composition. However, this proposition also gives rise to a corollary about full elimination of the non-binary variable-binding operators for alternative, sequential and parallel composition.

Corollary 5. *Let t be a comprehended term of the form $+^n_u t''$, $\cdot^n_u t''$ or $\|^n_u t''$ without comprehended terms as proper subterms, and let $k = size(t)$. Then there exists a binding term t' without non-binary variable-binding operators such that $t = t'$ is derivable from the axioms of the extended calculus and $size(t') = O(k^3)$ and $size(t') = \Omega(k^2)$.*

Proof. This is a direct consequence of Corollary 4 and Proposition 4. □

Corollary 5 implies that in the presence of an identity element for sequential composition, on full elimination of the non-binary variable-binding operators for alternative, sequential and parallel composition, the addition of process sequences to ACP process algebras does not give rise to significantly smaller or larger terms.

10 Concluding Remarks

We have introduced the notion of an ACP process algebra. The set of equations that have been taken to characterize ACP process algebras is a revision of the axiom system ACP. We consider this revision worth mentioning of itself, if only because it removes the need to have a constant for each atomic action. We have also introduced the notion of a meadow enriched ACP process algebra. This notion is a simple generalization of the notion of an ACP process algebra to processes in which data are involved, the mathematical structure of data being a meadow. The primary mathematical structure for calculations is unquestionably a field, and a meadow differs from a field only in that the multiplicative inverse operation is made total by imposing that the multiplicative inverse of zero is zero. Therefore, we consider the combination of ACP process algebras and meadows made in this paper, a combination with potentially many applications.

For all associative operators from the signature of meadow enriched ACP process algebras that are not of an auxiliary nature, we have introduced variable-binding operators as generalizations. Thus, we have obtained a process calculus whose terms can be interpreted in all meadow enriched ACP process algebras. We have shown that the use of variable-binding operators that bind variables with a two-valued range can already have a major impact on the size of terms, and that the impact can be further increased if we add an identity element for sequential composition to meadow enriched ACP process algebras. In addition, we have demonstrated that there is an alternative to introducing variable-binding operators for several associative operators on processes if we add a sort of process sequences and suitable operators on process sequences to meadow enriched ACP process algebras.

All variable-binding operators of the calculus associated with meadow enriched ACP process algebras can be eliminated from all terms of the calculus by means of its axioms, and all terms of the calculus can be directly interpreted in meadow enriched ACP process algebras. Therefore, although they yield a calculus, we consider these variable-binding operators to constitute a process algebraic feature. Fitting them in an algebraic framework does not involve any serious theoretical complication.

Different from the variable-binding operators introduced in this paper, the variable-binding operators from μCRL and PSF that generalize associative operators of ACP do not give rise to finitary comprehended terms. It is much more difficult to fit the variable-binding operators from those formalisms in an algebraic framework, see e.g. [22]. This also holds for the integration operator, which is found in extensions of the axiom system ACP concerning timed processes to allow for the alternative composition of a continuum of differently timed processes to be expressed (see e.g. [2]). It is worth mentioning that in effective μCRL, a restriction of μCRL for which a simulator is feasible (see e.g. [17]), the variable bound by the variable binding operator that generalizes alternative composition must have a finite range.

We have also attempted to fit variable-binding operators that bind variables with an infinite range in an algebraic framework. We have looked at binding algebras [29], which are second-order algebras of a specific kind that covers variable-binding operators. The problem is that the theory of binding algebras is insufficiently elaborate for our purpose. For example, it is not known whether the important characterization results from the theory of first-order algebras, i.e. Birkhoff's variety result and Malcev's quasi-variety result (see e.g. [14,26]), have generalizations for binding algebras.

It is known that many important results from the theory of first-order algebras, including the above-mentioned ones, have generalizations for higher-order algebras as considered in the theory of general higher-order algebras developed in [24,20,25]. Therefore, we have also considered the replacement of variable-binding operators by higher-order operators that give rise to such higher-order algebras. However, owing to the absence of bound variables, additional higher-order operators are needed which serve the same purpose as the combinators of combinatory logic [18]. Thus, this leads to the line taken earlier with combinatory process algebra [4].

References

1. Baeten, J.C.M., van Glabbeek, R.J.: Merge and termination in process algebra. In: Nori, K.V. (ed.) Proceedings 7th Conference on Foundations of Software Technology and Theoretical Computer Science. Lecture Notes in Computer Science, vol. 287, pp. 153–172. Springer-Verlag (1987)
2. Baeten, J.C.M., Middelburg, C.A.: Process Algebra with Timing. Monographs in Theoretical Computer Science, An EATCS Series, Springer-Verlag, Berlin (2002)
3. Baeten, J.C.M., Weijland, W.P.: Process Algebra, Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, Cambridge (1990)
4. Bergstra, J.A., Bethke, I., Ponse, A.: Process algebra with combinators. In: Börger, E., Gurevich, Y., Meinke, K. (eds.) CSL '93. Lecture Notes in Computer Science, vol. 832, pp. 36–65. Springer-Verlag (1994)
5. Bergstra, J.A., Bethke, I., Ponse, A.: Cancellation meadows: A generic basis theorem and some applications. Computer Journal 56(1), 3–14 (2013)
6. Bergstra, J.A., Hirshfeld, Y., Tucker, J.V.: Meadows and the equational specification of division. Theoretical Computer Science 410(12–13), 1261–1271 (2009)
7. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Information and Control 60(1–3), 109–137 (1984)
8. Bergstra, J.A., Middelburg, C.A.: Splitting bisimulations and retrospective conditions. Information and Computation 204(7), 1083–1138 (2006)
9. Bergstra, J.A., Middelburg, C.A.: Meadow enriched ACP process algebras. `arXiv:0901.3012v2 [math.RA]` (2009)
10. Bergstra, J.A., Middelburg, C.A.: A process calculus with finitary comprehended terms. `arXiv:0903.2914v1 [cs.LO]` (2009)
11. Bergstra, J.A., Middelburg, C.A.: Inversive meadows and divisive meadows. Journal of Applied Logic 9(3), 203–220 (2011)
12. Bergstra, J.A., Tucker, J.V.: Elementary algebraic specifications of the rational complex numbers. In: Futatsugi, K., et al. (eds.) Goguen Festschrift. Lecture Notes in Computer Science, vol. 4060, pp. 459–475. Springer-Verlag (2006)
13. Bergstra, J.A., Tucker, J.V.: The rational numbers as an abstract data type. Journal of the ACM 54(2), Article 7 (2007)
14. Burris, S., Sankappanavar, H.P.: A Course in Universal Algebra, Graduate Texts in Mathematics, vol. 78. Springer-Verlag, Berlin (1981)
15. Fokkink, W.J.: Introduction to Process Algebra. Texts in Theoretical Computer Science, An EATCS Series, Springer-Verlag, Berlin (2000)
16. Groote, J.F., Ponse, A.: Proof theory for μCRL: A language for processes with data. In: Andrews, D.J., Groote, J.F., Middelburg, C.A. (eds.) Semantics of Specification Languages. pp. 232–251. Workshops in Computing Series, Springer-Verlag (1994)
17. Groote, J.F., Ponse, A.: The syntax and semantics of μCRL. In: Ponse, A., Verhoef, C., van Vlijmen, S.F.M. (eds.) Algebra of Communicating Processes 1994. pp. 26–62. Workshops in Computing Series, Springer-Verlag (1995)
18. Hindley, J.R., Seldin, J.P.: Introduction to Combinators and λ-calculus. Cambridge University Press, Cambridge (1986)
19. Hodges, W.A.: Model Theory, Encyclopedia of Mathematics and Its Applications, vol. 42. Cambridge University Press, Cambridge (1993)

20. Kosiuczenko, P., Meinke, K.: On the power of higher-order algebraic specification methods. Information and Computation 124(1), 85–101 (1996)
21. Koymans, C.P.J., Vrancken, J.L.M.: Extending process algebra with the empty process ϵ. Logic Group Preprint Series 1, Department of Philosophy, Utrecht University, Utrecht (1985)
22. Luttik, S.P.: Choice Quantification in Process Algebra. Ph.D. thesis, Programming Research Group, University of Amsterdam, Amsterdam (2002)
23. Mauw, S., Veltink, G.J.: A process specification formalism. Fundamenta Informaticae 13(2), 85–139 (1990)
24. Meinke, K.: Universal algebra in higher types. Theoretical Computer Science 100(2), 385–417 (1992)
25. Meinke, K.: Proof theory of higher-order equations: Conservativity, normal forms and term rewriting. Journal of Computer and System Sciences 67(1), 127–173 (2003)
26. Meinke, K., Tucker, J.V.: Universal algebra. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, vol. I, pp. 189–411. Oxford University Press, Oxford (1992)
27. RAISE Language Group: The RAISE Specification Language. Prentice-Hall, Englewood Cliffs (1992)
28. Sannella, D., Tarlecki, A.: Algebraic preliminaries. In: Astesiano, E., Kreowski, H.J., Krieg-Brückner, B. (eds.) Algebraic Foundations of Systems Specification, pp. 13–30. Springer-Verlag, Berlin (1999)
29. Sun Yong: An algebraic generalization of Frege structures – Binding algebras. Theoretical Computer Science 211(1–2), 189–232 (1999)
30. Vrancken, J.L.M.: The algebra of communicating processes with empty process. Theoretical Computer Science 177(2), 287–328 (1997)
31. Wirsing, M.: Algebraic specification. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 675–788. Elsevier, Amsterdam (1990)