Using Hoare Logic in a Process Algebra Setting

Bergstra, J.A.; Middelburg, C.A.

DOI
10.3233/FI-2021-2026
Publication date
2021
Document Version
Submitted manuscript
Published in
Fundamenta Informaticae

Using Hoare Logic in a Process Algebra Setting

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. This paper concerns the relation between process algebra and Hoare logic. We investigate the question whether and how a Hoare logic can be used for reasoning about how data change in the course of a process when reasoning equationally about that process. We introduce an extension of ACP (Algebra of Communicating Processes) with features that are relevant to processes in which data are involved, present a Hoare logic for the processes considered in this process algebra, and discuss the use of this Hoare logic as a complement to pure equational reasoning with the equational axioms of the process algebra.

Keywords: process algebra, data parameterized action, assignment action, guarded command, asserted process, Hoare logic.

1998 ACM Computing Classification: D.1.3, D.1.4, D.2.4, F.1.2, F.3.1.

1 Introduction

ACP (Algebra of Communicating Processes) and its extensions provide a setting for equational reasoning about processes of some kind. The processes about which reasoning is in demand are often processes in which data are involved. It is quite common for such a process that the data that are involved change in the course of the process and that the process proceeds at certain stages in a way that depends on the changing data. This means that reasoning about a process often involves reasoning about how data change in the course of that process. The question arises whether and how a Hoare logic can be used for the second kind of reasoning when reasoning equationally about a process. After all, processes of the kind described above are reminiscent of the processes that arise from the execution of imperative programs.

This paper is concerned with the above-mentioned question. We investigate it using an extension of ACP [10] with features that are relevant to processes in which data are involved and a Hoare logic of asserted processes based on this extension of ACP. The extension concerned is called ACP_ϵ^*-D. Its additional features include assignment actions to deal with data that change in the course of a process and guarded commands to deal with processes that proceed at certain stages in a way that depends on certain data. In the Hoare logic concerned, an asserted process is a formula of the form $\{\phi\}\,p\,\{\psi\}$, where p is a term of ACP_ϵ^*-D that denotes a process and ϕ and ψ are terms of ACP_ϵ^*-D that denote conditions.

We define what it means that an asserted process is true in such a way that $\{\phi\}\, p\, \{\psi\}$ is true iff a set of equations that represents this judgment is derivable from the axioms of ACP^*_ϵ-D. Such a definition is a prerequisite for an affirmative answer to the question whether and how a Hoare logic can be used for reasoning about how data change in the course of a process when reasoning equationally about that process. The set of equations that represents the judgment expresses that a certain equivalence relation holds between processes determined by the asserted process. The equivalence relation concerned may be a useful equivalence relation when reasoning about processes in which data are involved. However, it is not a congruence relation, i.e. it is not preserved by all contexts. This complicates pure equational reasoning considerably. The presented Hoare logic can be considered to be a means to get partially round the complications concerned.

This paper is organized as follows. We begin with presenting ACP^*_ϵ, an extension of ACP with the empty process constant ϵ and the binary iteration operator *, and ACP^*_ϵ-D, an extension of ACP^*_ϵ with features that are relevant to processes in which data are involved (Sections 2 and 3). We also present a structural operational semantics of ACP^*_ϵ-D, define a notion of bisimulation equivalence based on this semantics, and show that the axioms of ACP^*_ϵ-D are sound with respect to this bisimulation equivalence (Section 4). After that, we present a Hoare logic of asserted processes based on ACP^*_ϵ-D, define what it means that an asserted process is true, and show that the axioms and rules of this Hoare logic are sound with respect to this meaning (Section 5). Following this, we go further into the connection of the presented Hoare logic with ACP^*_ϵ-D by way of the equivalence relation referred to in the previous paragraph (Section 6). We also go into the use of the presented Hoare logic as a complement to pure equational reasoning with the axioms of ACP^*_ϵ-D by means of examples (Section 7). Finally, we discuss related work and make some concluding remarks (Sections 8 and 9).

2 ACP with the Empty Process and Iteration

In this section, we present ACP^*_ϵ, ACP [10] extended with the empty process constant ϵ as in [7, Section 4.4] and the binary iteration operator * as in [8]. In ACP^*_ϵ, it is assumed that a fixed but arbitrary finite set A of *basic actions*, with $\delta, \epsilon \notin \mathsf{A}$, and a fixed but arbitrary commutative and associative *communication* function $\gamma : (\mathsf{A} \cup \{\delta\}) \times (\mathsf{A} \cup \{\delta\}) \to (\mathsf{A} \cup \{\delta\})$, such that $\gamma(\delta, a) = \delta$ for all $a \in \mathsf{A} \cup \{\delta\}$, have been given. Basic actions are taken as atomic processes. The function γ is regarded to give the result of synchronously performing any two basic actions for which this is possible, and to be δ otherwise. Henceforth, we write A_δ for $\mathsf{A} \cup \{\delta\}$.

The algebraic theory ACP^*_ϵ has one sort: the sort $\mathbf{P}$ of *processes*. We make this sort explicit to anticipate the need for many-sortedness later on. The algebraic theory ACP^*_ϵ has the following constants and operators to build terms of sort $\mathbf{P}$:

- the *inaction* constant $\delta : \to \mathbf{P}$;
- the *empty process* constant $\epsilon : \to \mathbf{P}$;

– for each $a \in \mathsf{A}$, the *basic action* constant $a : \to \mathbf{P}$;
– 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 *iteration* operator $^* : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– the binary *parallel composition* operator $\parallel : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– the binary *left merge* operator $\lfloor\!\lfloor : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– the binary *communication merge* operator $\mid : \mathbf{P} \times \mathbf{P} \to \mathbf{P}$;
– for each $H \subseteq \mathsf{A}$, the unary *encapsulation* operator $\partial_H : \mathbf{P} \to \mathbf{P}$.

We assume that there is a countably infinite set of variables of sort $\mathbf{P}$, which contains x, y and z. Terms are built as usual. We use infix notation for the binary operators. The following precedence conventions are used to reduce the need for parentheses: the operator $\cdot$ binds stronger than all other binary operators and the operator $+$ binds weaker than all other binary operators.

The constants and operators of ACP^*_ϵ are the constants and operators of ACP_ϵ [7, Section 4.4] and additionally the iteration operator *. Let p and q be closed ACP^*_ϵ terms, $a \in \mathsf{A}$, and $H \subseteq \mathsf{A}$.[1] Then the constants and operators of ACP^*_ϵ can be explained as follows:

– the constant δ denotes the process that is not capable of doing anything, not even terminating successfully;
– the constant ϵ denotes the process that is only capable of terminating successfully;
– the constant a denotes the process that is only capable of first performing action a and next terminating successfully;
– a closed term of the form $p + q$ denotes the process that behaves either as the process denoted by p or as the process denoted by q, but not both;
– a closed term of the form $p \cdot q$ denotes the process that first behaves as the process denoted by p and on successful termination of that process next behaves as the process denoted by q;
– a closed term of the form $p \,^*\, q$ denotes the process that behaves either as the process denoted by q or as the process that first behaves as the process denoted by p and on successful termination of that process next behaves as $p \,^*\, q$ again;
– a closed term of the form $p \parallel q$ denotes the process that behaves as the processes denoted by p and q taking place in parallel, by which we understand that, each time an action is performed, either a next action of one of the two processes is performed or a next action of the former process and a next action of the latter process are performed synchronously;
– a closed term of the form $p \lfloor\!\lfloor q$ denotes the process that behaves the same as the process denoted by $p \parallel q$, except that it starts with performing an action of the process denoted by p;
– a closed term of the form $p \mid q$ denotes the process that behaves the same as the process denoted by $p \parallel q$, except that it starts with performing an action of the process denoted by p and an action of the process denoted by q synchronously;

[1] As usual, a term in which no variables occur is called a closed term.

Table 1. Axioms of ACP^*_ϵ

$x + y = y + x$	A1	$x \parallel y = x \between y + y \between x + x \mid y + \partial_{\mathsf{A}}(x) \cdot \partial_{\mathsf{A}}(y)$		CM1T
$(x + y) + z = x + (y + z)$	A2	$\epsilon \between x = \delta$		CM2T
$x + x = x$	A3	$a \cdot x \between y = a \cdot (x \parallel y)$		CM3
$(x + y) \cdot z = x \cdot z + y \cdot z$	A4	$(x + y) \between z = x \between z + y \between z$		CM4
$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	A5	$\epsilon \mid x = \delta$		CM5T
$x + \delta = x$	A6	$x \mid \epsilon = \delta$		CM6T
$\delta \cdot x = \delta$	A7	$a \cdot x \mid b \cdot y = \gamma(a, b) \cdot (x \parallel y)$		CM7
$x \cdot \epsilon = x$	A8	$(x + y) \mid z = x \mid z + y \mid z$		CM8
$\epsilon \cdot x = x$	A9	$x \mid (y + z) = x \mid y + x \mid z$		CM9
		$\partial_H(\epsilon) = \epsilon$		D0
$x \mathbin{^*} y = x \cdot (x \mathbin{^*} y) + y$	BKS1	$\partial_H(a) = a$	if $a \notin H$	D1
$z = x \cdot z + y \;\rightarrow\; z = x \mathbin{^*} y$	RSP$*$	$\partial_H(a) = \delta$	if $a \in H$	D2
		$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		D3
		$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$		D4

Table 2. Derivable equations for iteration

$x \mathbin{^*} (y \cdot z) = (x \mathbin{^*} y) \cdot z$	BKS2
$x \mathbin{^*} (y \cdot ((x + y) \mathbin{^*} z) + z) = (x + y) \mathbin{^*} z$	BKS3
$\partial_H(x \mathbin{^*} y) = \partial_H(x) \mathbin{^*} \partial_H(y)$	BKS4
$\epsilon \mathbin{^*} x = x$	BKS5

– a closed term of the form $\partial_H(p)$ denotes the process that behaves the same as the process denoted by p, except that actions from H are blocked.

The axioms of ACP^*_ϵ are the equations given in Table 1. In these equations, a and b stand for arbitrary constants of ACP^*_ϵ that differ from ϵ and H stands for an arbitrary subset of A. So, CM3, CM7, and D0–D4 are actually axiom schemas. Axioms A1–A9, CM1T, CM2T, CM3, CM4, CM5T, CM6T, CM7–CM9, and D0–D4 are the axioms of ACP_ϵ (cf. [7, Section 4.4]). Axioms BKS1 and RSP* have been taken from [9].

The iteration operator originates from [8], where it is called the binary Kleene star operator. The unary counterpart of this operator can be defined by the equation $x^* = x \mathbin{^*} \epsilon$. From this defining equation, it follows, using RSP*, that $x^* = x \cdot x^* + \epsilon$ and also that $x \mathbin{^*} y = x^* \cdot y$.

Among the equations derivable from the axioms of ACP^*_ϵ are the equations concerning the iteration operator given in Table 2. In the axiom system of ACP^* given in [8], the axioms for the iteration operator are BKS1–BKS4 instead of BKS1 and RSP*. There exist equations derivable from the axioms of ACP^*_ϵ that are not derivable from the axioms of ACP^*_ϵ with BKS1 and RSP* replaced by

BKS1–BKS5. For example, the equation $a^* \delta = (a \cdot a)^* \delta$ is derivable with BKS1 and RSP*, but not with BKS1–BKS5 (cf. [28]). Moreover, we do not see how Theorem 5 of this paper can be proved if RSP* is replaced by BKS2–BKS5 (see the remark following the proof of the theorem).

3 Data Enriched ACP$^*_\epsilon$

In this section, we present ACP$^*_\epsilon$-D, data enriched ACP$^*_\epsilon$. This extension of ACP$^*_\epsilon$ has been inspired by [12]. It extends ACP$^*_\epsilon$ with features that are relevant to processes in which data are involved, such as guarded commands (to deal with processes that only take place if some data-dependent condition holds), data parameterized actions (to deal with process interactions with data transfer), and assignment actions (to deal with data that change in the course of a process).

In ACP$^*_\epsilon$-D, it is assumed that the following has been given with respect to data:

- a (single- or many-sorted) signature $\Sigma_{\mathfrak{D}}$ that includes a sort $\mathbf{D}$ of *data* and constants and/or operators with result sort $\mathbf{D}$;
- a minimal algebra $\mathfrak{D}$ of the signature $\Sigma_{\mathfrak{D}}$.

Moreover, it is assumed that a countably infinite set $\mathcal{V}$ of *flexible variables* has been given. A flexible variable is a variable whose value may change in the course of a process.[2] Flexible variables are found under the name program variables in imperative programming. We write $\mathbb{D}$ for the set of all closed terms over the signature $\Sigma_{\mathfrak{D}}$ that are of sort $\mathbf{D}$. An *evaluation map* is a function σ from $\mathcal{V}$ to $\mathbb{D} \cup \mathcal{V}$ where, for all $v \in \mathcal{V}$, $\sigma(v) = v$ if $\sigma(v) \in \mathcal{V}$. Let σ be an evaluation map and let V be a finite subset of $\mathcal{V}$. Then σ is a *V-evaluation map* if, for all $v \in \mathcal{V}$, $\sigma(v) \in \mathbb{D}$ iff $v \in V$.

Evaluation maps are intended to provide the data values assigned to flexible variables of sort $\mathbf{D}$ when a term of sort $\mathbf{D}$ is evaluated. However, in order to fit better in an algebraic setting, they provide closed terms over the signature $\Sigma_{\mathfrak{D}}$ that denote those data values instead. The requirement that $\mathfrak{D}$ is a minimal algebra guarantees that each data value can be represented by a closed term. The possibility to map flexible variables to themselves may be used for partial evaluation, i.e. evaluation where some flexible variables are not evaluated.

The algebraic theory ACP$^*_\epsilon$-D has three sorts: the sort $\mathbf{P}$ of *processes*, the sort $\mathbf{C}$ of *conditions*, and the sort $\mathbf{D}$ of *data*. ACP$^*_\epsilon$-D has the constants and operators from $\Sigma_{\mathfrak{D}}$ and in addition the following constants to build terms of sort $\mathbf{D}$:

- for each $v \in \mathcal{V}$, the *flexible variable* constant $v : \to \mathbf{D}$.

ACP$^*_\epsilon$-D has the following constants and operators to build terms of sort $\mathbf{C}$:

- the binary *equality* operator $= : \mathbf{D} \times \mathbf{D} \to \mathbf{C}$;

[2] The term flexible variable is used for this kind of variables in e.g. [27,20].

– the *truth* constant $\mathsf{t} : \to \mathbf{C}$;
– the *falsity* constant $\mathsf{f} : \to \mathbf{C}$;
– the unary *negation* operator $\neg : \mathbf{C} \to \mathbf{C}$;
– the binary *conjunction* operator $\wedge : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$;
– the binary *disjunction* operator $\vee : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$;
– the binary *implication* operator $\to : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$;
– the unary variable-binding *universal quantification* operator $\forall : \mathbf{C} \to \mathbf{C}$ that binds a variable of sort $\mathbf{D}$;
– the unary variable-binding *existential quantification* operator $\exists : \mathbf{C} \to \mathbf{C}$ that binds a variable of sort $\mathbf{D}$.

ACP^*_ϵ-D has the constants and operators of ACP^*_ϵ and in addition the following operators to build terms of sort $\mathbf{P}$:

– the binary *guarded command* operator $:\to : \mathbf{C} \times \mathbf{P} \to \mathbf{P}$;
– for each $n \in \mathbb{N}$, for each $a \in \mathsf{A}$, the n-ary *data parameterized action* operator $a : \underbrace{\mathbf{D} \times \cdots \times \mathbf{D}}_{n \text{ times}} \to \mathbf{P}$;
– for each $v \in \mathcal{V}$, a unary *assignment action* operator $v{:=} : \mathbf{D} \to \mathbf{P}$;
– for each evaluation map σ, a unary *evaluation* operator $\mathsf{V}_\sigma : \mathbf{P} \to \mathbf{P}$.

We assume that there are countably infinite sets of variables of sort $\mathbf{C}$ and $\mathbf{D}$ and that the sets of variables of sort $\mathbf{P}$, $\mathbf{C}$, and $\mathbf{D}$ are mutually disjoint and disjoint from $\mathcal{V}$. The formation rules for terms are the usual ones for the many-sorted case (see e.g. [26,29]) and in addition the following rule:

– if O is a variable-binding operator $O : S_1 \times \ldots \times S_n \to S$ that binds a variable of sort S', $t_1, \ldots, t_n$ are terms of sorts $S_1, \ldots, S_n$, respectively, and X is a variable of sort S', then $OX(t_1, \ldots, t_n)$ is a term of sort S (cf. [25]).

We use the same notational conventions as before. We also use infix notation for the additional binary operators. Moreover, we use the notation $[v := e]$, where $v \in \mathcal{V}$ and e is a term of sort $\mathbf{D}$, for the term $v := (e)$.

We use the notation $\phi \leftrightarrow \psi$, where ϕ and ψ are terms of sort $\mathbf{C}$, for the term $(\phi \to \psi) \wedge (\psi \to \phi)$. Moreover, we use the notation $\bigvee \Phi$, where $\Phi = \{\phi_1, \ldots, \phi_n\}$ and $\phi_1, \ldots, \phi_n$ are terms of sort $\mathbf{C}$, for the term $\phi_1 \vee \ldots \vee \phi_n$.

We write $\mathcal{P}$ for the set of all closed terms of sort $\mathbf{P}$, $\mathcal{C}$ for the set of all closed terms of sort $\mathbf{C}$, and $\mathcal{D}$ for the set of all closed terms of sort $\mathbf{D}$.

Each term from $\mathcal{C}$ can be taken as a formula of a first-order language with equality of $\mathfrak{D}$ by taking the flexible variable constants as additional variables of sort $\mathbf{D}$. We implicitly take the flexible variable constants as additional variables of sort $\mathbf{D}$ wherever the context asks for a formula. In this way, each term from $\mathcal{C}$ can be interpreted as a formula in $\mathfrak{D}$. The axioms of ACP^*_ϵ-D (given below) include an equation $\phi = \psi$ for each two terms ϕ and ψ from $\mathcal{C}$ for which the formula $\phi \leftrightarrow \psi$ holds in $\mathfrak{D}$.

Let p be a term from $\mathcal{P}$, ϕ be a term from $\mathcal{C}$, and $e_1, \ldots, e_n$ and e be terms from $\mathcal{D}$. Then the additional operators can be explained as follows:

– the term $\phi :\rightarrow p$ denotes the process that behaves as the process denoted by p under condition ϕ;
– the term $a(e_1, \ldots, e_n)$ denotes the process that is only capable of first performing action $a(e_1, \ldots, e_n)$ and next terminating successfully;
– the term $[v := e]$ denotes the process that is only capable of first performing action $[v := e]$, whose intended effect is the assignment of the result of evaluating e to flexible variable v, and next terminating successfully;
– the term $\mathsf{V}_\sigma(p)$ denotes the process that behaves the same as the process denoted by p except that each subterm of p that belongs to $\mathcal{D}$ is evaluated using the evaluation map σ updated according to the assignment actions that have taken place at the point where the subterm is encountered.

Evaluation operators are a variant of state operators (see e.g. [3]).

The guarded command operator is often used to construct ACP^*_ϵ-D terms that are reminiscent of control flow statements of imperative programming languages. For example, terms of the form $\phi :\rightarrow t + (\neg\, \phi) :\rightarrow t'$ are reminiscent of if-then-else statements and terms of the form $(\phi :\rightarrow t)^* ((\neg\, \phi) :\rightarrow \epsilon)$ are reminiscent of while-do statements. The following ACP^*_ϵ-D term contains a subterm of the latter form $(i, j, q, r \in \mathcal{V})$:

$$[q := 0] \cdot [r := i] \cdot (((r \geq j) :\rightarrow [q := q + 1] \cdot [r := r - j])^* ((\neg\, r \geq j) :\rightarrow \epsilon)) \,.$$

This term is reminiscent of a program that computes the quotient and remainder of dividing two integers by repeated subtraction. That is, the final values of q and r are the quotient and remainder of dividing the initial value of i by the initial value of j. An evaluation operator can be used to show that this is the case for given initial values of i and j. For example, consider the case where the initial values of i and j are 11 and 3, respectively. Let σ be an evaluation map such that $\sigma(i) = 11$ and $\sigma(j) = 3$. Then the following equation can be derived from the axioms of ACP^*_ϵ-D given below:

$$\begin{aligned}&\mathsf{V}_\sigma([q := 0] \cdot [r := i] \cdot (((r \geq j) :\rightarrow [q := q + 1] \cdot [r := r - j])^* ((\neg\, r \geq j) :\rightarrow \epsilon)))\\ &\quad = [q := 0] \cdot [r := 11] \cdot [q := 1] \cdot [r := 8] \cdot [q := 2] \cdot [r := 5] \cdot [q := 3] \cdot [r := 2] \,.\end{aligned}$$

This equation shows that in the case where the initial values of i and j are 11 and 3 the final values of q and r are 3 and 2 (which are the quotient and remainder of dividing 11 by 3).

An evaluation map σ can be extended homomorphically from flexible variables to terms of sort **D** and terms of sort **C**. These extensions are denoted by σ as well. We write $\sigma\{e/v\}$ for the evaluation map σ' defined by $\sigma'(v') = \sigma(v')$ if $v' \not\equiv v$ and $\sigma'(v) = e$.

The axioms of ACP^*_ϵ-D are the axioms of ACP^*_ϵ and in addition the equations given in Table 3. In these equations, ϕ and ψ stand for arbitrary terms from $\mathcal{C}$, e, $e_1, e_2, \ldots$, and e', $e'_1, e'_2, \ldots$ stand for arbitrary terms from $\mathcal{D}$, v stands for an arbitrary flexible variable from $\mathcal{V}$, σ stands for an arbitrary evaluation map, a and b stand for arbitrary constants of ACP^*_ϵ-D that differ from ϵ, c stands for

Table 3. Axioms of ACP^*_ϵ-D

$e = e'$	if $\mathfrak{D} \models e = e'$	IMP1
$\phi = \psi$	if $\mathfrak{D} \models \phi \leftrightarrow \psi$	IMP2
$\mathsf{t} :\to x = x$		GC1
$\mathsf{f} :\to x = \delta$		GC2
$\phi :\to \delta = \delta$		GC3
$\phi :\to (x + y) = \phi :\to x + \phi :\to y$		GC4
$\phi :\to x \cdot y = (\phi :\to x) \cdot y$		GC5
$\phi :\to (\psi :\to x) = (\phi \wedge \psi) :\to x$		GC6
$(\phi \vee \psi) :\to x = \phi :\to x + \psi :\to x$		GC7
$(\phi :\to x) \between y = \phi :\to (x \between y)$		GC8
$(\phi :\to x) \mid y = \phi :\to (x \mid y)$		GC9
$x \mid (\phi :\to y) = \phi :\to (x \mid y)$		GC10
$\partial_H(\phi :\to x) = \phi :\to \partial_H(x)$		GC11
$\mathsf{V}_\sigma(\epsilon) = \epsilon$		V0
$\mathsf{V}_\sigma(a \cdot x) = a \cdot \mathsf{V}_\sigma(x)$		V1
$\mathsf{V}_\sigma(a(e_1, \ldots, e_n) \cdot x) = a(\sigma(e_1), \ldots, \sigma(e_n)) \cdot \mathsf{V}_\sigma(x)$		V2
$\mathsf{V}_\sigma([v := e] \cdot x) = [v := \sigma(e)] \cdot \mathsf{V}_{\sigma\{\sigma(e)/v\}}(x)$		V3
$\mathsf{V}_\sigma(x + y) = \mathsf{V}_\sigma(x) + \mathsf{V}_\sigma(y)$		V4
$\mathsf{V}_\sigma(\phi :\to y) = \sigma(\phi) :\to \mathsf{V}_\sigma(x)$		V5
$a(e_1, \ldots, e_n) \cdot x \between y = a(e_1, \ldots, e_n) \cdot (x \parallel y)$		CM3D
$a(e_1, \ldots, e_n) \cdot x \mid b(e'_1, \ldots, e'_n) \cdot y =$		
$\quad (e_1 = e'_1 \wedge \ldots \wedge e_n = e'_n) :\to c(e_1, \ldots, e_n) \cdot (x \parallel y)$	if $\gamma(a, b) = c$	CM7Da
$a(e_1, \ldots, e_n) \cdot x \mid b(e'_1, \ldots, e'_m) \cdot y = \delta$	if $\gamma(a, b) = \delta$ or $n \neq m$	CM7Db
$a(e_1, \ldots, e_n) \cdot x \mid b \cdot y = \delta$		CM7Dc
$a \cdot x \mid b(e_1, \ldots, e_n) \cdot y = \delta$		CM7Dd
$\partial_H(a(e_1, \ldots, e_n)) = a(e_1, \ldots, e_n)$	if $a \notin H$	D1D
$\partial_H(a(e_1, \ldots, e_n)) = \delta$	if $a \in H$	D2D
$[v := e] \cdot x \between y = [v := e] \cdot (x \parallel y)$		CM3A
$[v := e] \cdot x \mid y = \delta$		CM5A
$x \mid [v := e] \cdot y = \delta$		CM6A
$\partial_H([v := e]) = [v := e]$		D1A

an arbitrary constant of ACP^*_ϵ-D that differ from ϵ and δ, and H stands for an arbitrary subset of A. Axioms GC1–GC11 have been taken from [4] (using a different numbering), but with the axioms with occurrences of conditional expressions of the form $p \lhd \phi \rhd q$ replaced by simpler axioms. Axioms CM3D, CM7Da, CM7Db, D1D, and D2D have been inspired by [12].

The set $\mathcal{A}$ of *actions* of ACP^*_ϵ-D is inductively defined by the following rules:

- if $a \in \mathsf{A}$, then $a \in \mathcal{A}$;
- if $a \in \mathsf{A}$ and $e_1, \ldots, e_n \in \mathcal{D}$, then $a(e_1, \ldots, e_n) \in \mathcal{A}$;
- if $v \in \mathcal{V}$ and $e \in \mathcal{D}$, then $[v := e] \in \mathcal{A}$.

The elements of $\mathcal{A}$ are the processes that are considered to be atomic.

The set $\mathcal{H}$ of *head normal forms* of ACP^*_ϵ-D is inductively defined by the following rules:

- $\delta \in \mathcal{H}$;
- if $\phi \in \mathcal{C}$, then $\phi :\to \epsilon \in \mathcal{H}$;
- if $\phi \in \mathcal{C}$, $\alpha \in \mathcal{A}$, and $p \in \mathcal{P}$, then $\phi :\to a \cdot p \in \mathcal{H}$;
- if $p, p' \in \mathcal{H}$, then $p + p' \in \mathcal{H}$.

The following lemma about head normal forms is used in later sections.

Lemma 1. *For all terms $p \in \mathcal{P}$, there exists a term $q \in \mathcal{H}$ such that $p = q$ is derivable from the axioms of* ACP^*_ϵ-D.

Proof. This is straightforwardly proved by induction on the structure of p. The cases where p is of the form δ, ϵ or α ($\alpha \in \mathcal{A}$) are trivial. The case where p is of the form $p_1 + p_2$ follows immediately from the induction hypothesis. The case where p is of the form $p_1 \parallel p_2$ follows immediately from the case that p is of the form $p_1 \between p_2$ and the case that p is of the form $p_1 \mid p_2$. Each of the other cases follow immediately from the induction hypothesis and a claim that is easily proved by structural induction. In the case where p is of the form $p_1 \mid p_2$, each of the cases to be considered in the inductive proof demands an additional proof by structural induction. □

Some earlier extensions of ACP include Hoare's ternary counterpart of the binary guarded command operator (see e.g. [4]). This operator can be defined by the equation $x \lhd u \rhd y = u :\to x + (\neg\, u) :\to y$. From this defining equation, it follows that $u :\to x = x \lhd u \rhd \delta$. In [15], a unary counterpart of the binary guarded command operator is used. This operator can be defined by the equation $\{u\} = u :\to \epsilon$. From this defining equation, it follows that $u :\to x = \{u\} \cdot x$ and also that $\{\mathsf{t}\} = \epsilon$ and $\{\mathsf{f}\} = \delta$. In [15], the processes denoted by closed terms of the form $\{\phi\}$ are called guards.

4 Structural Operational Semantics and Bisimulation Equivalence

In this section, we present a structural operational semantics of ACP^*_ϵ-D, define a notion of bisimulation equivalence based on this semantics, and show that the axioms of ACP^*_ϵ-D are sound with respect to this bisimulation equivalence.

We write $\mathcal{C}^{sat}$ for the set of all terms $\phi \in \mathcal{C}$ for which $\mathfrak{D} \not\models \phi \leftrightarrow \mathsf{f}$. As formulas of a first-order language with equality of $\mathfrak{D}$, the terms from $\mathcal{C}^{sat}$ are the formulas that are satisfiable in $\mathfrak{D}$.

We start with the presentation of the structural operational semantics of ACP_ϵ^*-D. The following transition relations on $\mathcal{P}$ are used:

- for each $\phi \in \mathcal{C}^{sat}$, a unary relation ${}^{\{\phi\}}\!\!\downarrow$;
- for each $\ell \in \mathcal{C}^{sat} \times \mathcal{A}$, a binary relation $\xrightarrow{\ell}$.

We write $p\ {}^{\{\phi\}}\!\!\downarrow$ instead of $p \in {}^{\{\phi\}}\!\!\downarrow$ and $p \xrightarrow{\{\phi\}\,\alpha} q$ instead of $(p,q) \in \xrightarrow{(\phi,\alpha)}$. The relations ${}^{\{\phi\}}\!\!\downarrow$ and $\xrightarrow{\ell}$ can be explained as follows:

- $p\ {}^{\{\phi\}}\!\!\downarrow$: p is capable of terminating successfully under condition ϕ;
- $p \xrightarrow{\{\phi\}\,\alpha} q$: p is capable of performing action α under condition ϕ and then proceeding as q.

The structural operational semantics of ACP_ϵ^*-D is described by the transition rules given in Table 4. In this table, a, b, and c stand for arbitrary basic actions from A, v stands for an arbitrary flexible variable from $\mathcal{V}$, e and $e_1, e_2, \ldots$ stand for arbitrary terms from $\mathcal{D}$, ϕ and ψ stand for arbitrary terms from $\mathcal{C}^{sat}$, α stands for an arbitrary term from $\mathcal{A}$, H stands for arbitrary subset of A, and σ stands for an arbitrary evaluation map.

Two process are considered equal if they can simulate each other. In order to make this precise, we will define the notion of bisimulation equivalence on the set $\mathcal{P}$ below. In the definition concerned, we need an equivalence relation on the set $\mathcal{A}$.

Two actions $\alpha, \alpha' \in \mathcal{A}$ are *data equivalent*, written $\alpha \simeq \alpha'$, iff one of the following holds:

- there exists an $a \in \mathsf{A}$ such that $\alpha = a$ and $\alpha' = a$;
- there exist an $a \in \mathsf{A}$ and $e_1, \ldots, e_n, e'_1, \ldots, e'_n \in \mathcal{D}$ such that $\mathfrak{D} \models e_1 = e'_1 \wedge \ldots \wedge e_n = e'_n$, $\alpha = a(e_1, \ldots, e_n)$, and $\alpha' = a(e'_1, \ldots, e'_n)$;
- there exist a $v \in \mathcal{V}$ and $e, e' \in \mathcal{D}$ such that $\mathfrak{D} \models e = e'$, $\alpha = [v := e]$, and $\alpha' = [v := e']$.

We write $[\alpha]$, where $\alpha \in \mathcal{A}$, for the equivalence class of α with respect to $\simeq$.

A *bisimulation* is a binary relation R on $\mathcal{P}$ such that, for all terms $p, q \in \mathcal{P}$ with $(p,q) \in R$, the following conditions hold:

- if $p \xrightarrow{\{\phi\}\,\alpha} p'$, then there exists a finite set $\Psi \subseteq \mathcal{C}^{sat}$ such that $\mathfrak{D} \models \phi \to \bigvee \Psi$ and, for all $\psi \in \Psi$, there exist an $\alpha' \in [\alpha]$ and a $q' \in \mathcal{P}$ such that $q \xrightarrow{\{\psi\}\,\alpha'} q'$ and $(p', q') \in R$;
- if $q \xrightarrow{\{\phi\}\,\alpha} q'$, then there exists a finite set $\Psi \subseteq \mathcal{C}^{sat}$ such that $\mathfrak{D} \models \phi \to \bigvee \Psi$ and, for all $\psi \in \Psi$, there exist an $\alpha' \in [\alpha]$ and a $p' \in \mathcal{P}$ such that $p \xrightarrow{\{\psi\}\,\alpha'} p'$ and $(p', q') \in R$;
- if $p\ {}^{\{\phi\}}\!\!\downarrow$, then there exists a finite set $\Psi \subseteq \mathcal{C}^{sat}$ such that $\mathfrak{D} \models \phi \to \bigvee \Psi$ and, for all $\psi \in \Psi$, $q\ {}^{\{\psi\}}\!\!\downarrow$;

Table 4. Transition rules for ACP^*_ϵ-D

$$\frac{}{\epsilon \overset{\{\mathsf{t}\}}{\downarrow}}$$

$$\frac{}{a \xrightarrow{\{\mathsf{t}\}\, a} \epsilon} \qquad \frac{}{a(e_1,\ldots,e_n) \xrightarrow{\{\mathsf{t}\}\, a(e_1,\ldots,e_n)} \epsilon} \qquad \frac{}{[v := e] \xrightarrow{\{\mathsf{t}\}\, [v:=e]} \epsilon}$$

$$\frac{x \overset{\{\phi\}}{\downarrow}}{x + y \overset{\{\phi\}}{\downarrow}} \qquad \frac{y \overset{\{\phi\}}{\downarrow}}{x + y \overset{\{\phi\}}{\downarrow}} \qquad \frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{x + y \xrightarrow{\{\phi\}\, \alpha} x'} \qquad \frac{y \xrightarrow{\{\phi\}\, \alpha} y'}{x + y \xrightarrow{\{\phi\}\, \alpha} y'}$$

$$\frac{x \overset{\{\phi\}}{\downarrow},\ y \overset{\{\psi\}}{\downarrow}}{x \cdot y \overset{\{\phi\wedge\psi\}}{\downarrow}}\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f} \qquad \frac{x \overset{\{\phi\}}{\downarrow},\ y \xrightarrow{\{\psi\}\, \alpha} y'}{x \cdot y \xrightarrow{\{\phi\wedge\psi\}\, \alpha} y'}\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f} \qquad \frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{x \cdot y \xrightarrow{\{\phi\}\, \alpha} x' \cdot y}$$

$$\frac{y \overset{\{\phi\}}{\downarrow}}{x * y \overset{\{\phi\}}{\downarrow}} \qquad \frac{y \xrightarrow{\{\phi\}\, \alpha} y'}{x * y \xrightarrow{\{\phi\}\, \alpha} y'} \qquad \frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{x * y \xrightarrow{\{\phi\}\, \alpha} x' \cdot (x * y)}$$

$$\frac{x \overset{\{\phi\}}{\downarrow}}{\psi :\rightarrow x \overset{\{\phi\wedge\psi\}}{\downarrow}}\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f} \qquad \frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{\psi :\rightarrow x \xrightarrow{\{\phi\wedge\psi\}\, \alpha} x'}\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f}$$

$$\frac{x \overset{\{\phi\}}{\downarrow},\ y \overset{\{\psi\}}{\downarrow}}{x \parallel y \overset{\{\phi\wedge\psi\}}{\downarrow}}\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f} \qquad \frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{x \parallel y \xrightarrow{\{\phi\}\, \alpha} x' \parallel y} \qquad \frac{y \xrightarrow{\{\phi\}\, \alpha} y'}{x \parallel y \xrightarrow{\{\phi\}\, \alpha} x \parallel y'}$$

$$\frac{x \xrightarrow{\{\phi\}\, a} x',\ y \xrightarrow{\{\psi\}\, b} y'}{x \parallel y \xrightarrow{\{\phi\wedge\psi\}\, c} x' \parallel y'}\ \gamma(a,b) = c,\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f}$$

$$\frac{x \xrightarrow{\{\phi\}\, a(e_1,\ldots,e_n)} x',\ y \xrightarrow{\{\psi\}\, b(e'_1,\ldots,e'_n)} y'}{x \parallel y \xrightarrow{\{\phi\wedge\psi\wedge e_1=e'_1\wedge\ldots\wedge e_n=e'_n\}\, c(e_1,\ldots,e_n)} x' \parallel y'}\ \begin{array}{c}\gamma(a,b) = c,\\ \mathfrak{D} \not\models \phi \wedge \psi \wedge e_1 = e'_1 \wedge \ldots \wedge e_n = e'_n \leftrightarrow \mathsf{f}\end{array}$$

$$\frac{x \xrightarrow{\{\phi\}\, \alpha} x'}{x \between y \xrightarrow{\{\phi\}\, \alpha} x' \parallel y}$$

$$\frac{x \xrightarrow{\{\phi\}\, a} x',\ y \xrightarrow{\{\psi\}\, b} y'}{x \mid y \xrightarrow{\{\phi\wedge\psi\}\, c} x' \parallel y'}\ \gamma(a,b) = c,\ \mathfrak{D} \not\models \phi \wedge \psi \leftrightarrow \mathsf{f}$$

$$\frac{x \xrightarrow{\{\phi\}\, a(e_1,\ldots,e_n)} x',\ y \xrightarrow{\{\psi\}\, b(e'_1,\ldots,e'_n)} y'}{x \mid y \xrightarrow{\{\phi\wedge\psi\wedge e_1=e'_1\wedge\ldots\wedge e_n=e'_n\}\, c(e_1,\ldots,e_n)} x' \parallel y'}\ \begin{array}{c}\gamma(a,b) = c,\\ \mathfrak{D} \not\models \phi \wedge \psi \wedge e_1 = e'_1 \wedge \ldots \wedge e_n = e'_n \leftrightarrow \mathsf{f}\end{array}$$

$$\frac{x \overset{\{\phi\}}{\downarrow}}{\partial_H(x) \overset{\{\phi\}}{\downarrow}} \qquad \frac{x \xrightarrow{\{\phi\}\, a} x'}{\partial_H(x) \xrightarrow{\{\phi\}\, a} \partial_H(x')}\ a \notin H \qquad \frac{x \xrightarrow{\{\phi\}\, a(e_1,\ldots,e_n)} x'}{\partial_H(x) \xrightarrow{\{\phi\}\, a(e_1,\ldots,e_n)} \partial_H(x')}\ a \notin H$$

$$\frac{x \xrightarrow{\{\phi\}\, [v:=e]} x'}{\partial_H(x) \xrightarrow{\{\phi\}\, [v:=e]} \partial_H(x')}$$

$$\frac{x \overset{\{\phi\}}{\downarrow}}{\mathsf{V}_\sigma(x) \overset{\{\sigma(\phi)\}}{\downarrow}} \qquad \frac{x \xrightarrow{\{\phi\}\, a} x'}{\mathsf{V}_\sigma(x) \xrightarrow{\{\sigma(\phi)\}\, a} \mathsf{V}_\sigma(x')} \qquad \frac{x \xrightarrow{\{\phi\}\, a(e_1,\ldots,e_n)} x'}{\mathsf{V}_\sigma(x) \xrightarrow{\{\sigma(\phi)\}\, a(\sigma(e_1),\ldots,\sigma(e_n))} \mathsf{V}_\sigma(x')}$$

$$\frac{x \xrightarrow{\{\phi\}\, [v:=e]} x'}{\mathsf{V}_\sigma(x) \xrightarrow{\{\sigma(\phi)\}\, [v:=\sigma(e)]} \mathsf{V}_{\sigma\{\sigma(e)/v\}}(x')}\ \sigma(v) \in \mathbb{D} \qquad \frac{x \xrightarrow{\{\phi\}\, [v:=e]} x'}{\mathsf{V}_\sigma(x) \xrightarrow{\{\sigma(\phi)\}\, [v:=\sigma(e)]} \mathsf{V}_\sigma(x')}\ \sigma(v) \notin \mathbb{D}$$

– if $q \stackrel{\{\phi\}}{\downarrow}$, then there exists a finite set $\Psi \subseteq \mathcal{C}^{sat}$ such that $\mathfrak{D} \models \phi \to \bigvee \Psi$ and, for all $\psi \in \Psi$, $p \stackrel{\{\psi\}}{\downarrow}$.

Two terms $p, q \in \mathcal{P}$ are *bisimulation equivalent*, written $p \leftrightarrows q$, if there exists a bisimulation R such that $(p, q) \in R$. Let R be a bisimulation such that $(p, q) \in R$. Then we say that R is a bisimulation *witnessing* $p \leftrightarrows q$.

The above definition of a bisimulation deviates from the standard definition because a transition on one side may be simulated by a set of transitions on the other side. For example, the transition $(\phi_1 \vee \phi_2) :\to a \cdot b \xrightarrow{\{\phi_1 \vee \phi_2\}\, a} b$ is simulated by the set of transitions consisting of $\phi_1 :\to a \cdot b \xrightarrow{\{\phi_1\}\, a} b$ and $\phi_2 :\to a \cdot b \xrightarrow{\{\phi_2\}\, a} b$. A bisimulation as defined above is called a *splitting* bisimulation in [11].

Bisimulation equivalence is a congruence with respect to the operators of ACP^*_ϵ-D of which the result sort and at least one argument sort is $\mathbf{P}$.

Theorem 1 (Congruence). *For all terms $p, q, p', q' \in \mathcal{P}$ and all terms $\phi \in \mathcal{C}$, $p \leftrightarrows p'$ and $q \leftrightarrows q'$ only if $p + q \leftrightarrows p' + q'$, $p \cdot q \leftrightarrows p' \cdot q'$, $p \mathbin{^*} q \leftrightarrows p' \mathbin{^*} q'$, $\phi :\to p \leftrightarrows \phi :\to p'$, $p \parallel q \leftrightarrows p' \parallel q'$, $p \lfloor\!\lfloor q \leftrightarrows p' \lfloor\!\lfloor q'$, $p \mid q \leftrightarrows p' \mid q'$, $\partial_H(p) \leftrightarrows \partial_H(p')$, and $\mathsf{V}_\sigma(p) \leftrightarrows \mathsf{V}_\sigma(p')$.*

Proof. We can reformulate the transition rules such that:

– bisimulation equivalence based on the reformulated transition rules according to the standard definition of bisimulation equivalence coincides with bisimulation equivalence based on the original transition rules according to the definition of bisimulation equivalence given above;
– the reformulated transition rules make up a transition system specification in path format.

The reformulation is similar to the one for the transition rules for BPAps outlined in [5]. The proposition follows now immediately from the well-known result that bisimulation equivalence according to the standard definition of bisimulation equivalence is a congruence if the transition rules concerned make up a transition system specification in path format (see e.g. [6]). □

The underlying idea of the reformulation referred to above is that we replace each transition $p \xrightarrow{\{\phi\}\, \alpha} p'$ by a transition $p \xrightarrow{\{\nu\}\, [\alpha]} p'$ for each valuation of variables ν such that $\mathfrak{D} \models \phi\,[\nu]$, and likewise $p \stackrel{\{\phi\}}{\downarrow}$. Thus, in a bisimulation, a transition on one side must be simulated by a single transition on the other side. We did not present the reformulated structural operational semantics in this paper because it is, in our opinion, intuitively less appealing.

The axioms of ACP^*_ϵ-D are sound with respect to $\leftrightarrows$ for equations between terms from $\mathcal{P}$.

Theorem 2 (Soundness). *For all terms $p, q \in \mathcal{P}$, $p = q$ is derivable from the axioms of* ACP^*_ϵ-D *only if $p \leftrightarrows q$.*

Proof. Because $\leftrightarrows$ is a congruence, it is sufficient to prove the theorem for all substitution instances of each axiom of ACP^*_ϵ-D. We will loosely say that a

relation contains all closed substitution instances of an equation if it contains all pairs (p, q) such that $p = q$ is a closed substitution instance of the equation.

For each axiom, we can construct a bisimulation R witnessing $p \leftrightarrows q$ for all closed substitution instances $p = q$ of the axiom as follows:

- in the case of A1–A6, A8, A9, BKS1, CM3, CM4, CM7–CM9, D1, D3, D4, GC1, GC4–GC11, V1–V5, CM3D, CM7Da, D1D, CM3A, and D1A, we take the relation R that consists of all closed substitution instances of the axiom concerned and the equation $x = x$;
- in the case of A7, CM2T, CM5T, CM6T, D0, D2, GC2, GC3, V0, CM7Db–CM7Dd, D2D, CM5A, and CM6A, we take the relation R that consists of all closed substitution instances of the axiom concerned;
- in the case of CM1T, we take the relation R that consists of all closed substitution instances of CM1T, the equation $x \parallel y = y \parallel x$, and the equation $x = x$;
- in the case of RSP*, we take the relation R that consists of all closed substitution instances $r = p^* q$ of the consequent of RSP* for which $r \leftrightarrows p \cdot r + q$ and all closed substitution instances of the equation $x = x$. □

We have not been able to prove the completeness of the axioms of ACP^*_ϵ-D with respect to $\leftrightarrows$ for equations between terms from $\mathcal{P}$. Such a proof would give an affirmative answer to an open question about the axiomatization of the iteration operator already posed in 1984 by Milner [23, page 465]. Until now, all attempts to answer this question have failed (see [14]).

5 A Hoare Logic of Asserted Processes

In this section, we present $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$, a Hoare logic of asserted processes based on ACP^*_ϵ-D, define what it means that an asserted process is true, and show that the axioms and rules of this logic are sound with respect to this meaning.

We write $\mathcal{P}^{hl}$ for the set of all closed terms of sort $\mathbf{P}$ in which the evaluation operators V_σ and the auxiliary operators $\between$ and $\mid$ do not occur and we write $\mathcal{C}^{hl}$ for the set of all terms of sort $\mathbf{C}$ in which variables of sort $\mathbf{C}$ do not occur. Clearly, $\mathcal{P}^{hl} \subset \mathcal{P}$ and $\mathcal{C} \subset \mathcal{C}^{hl}$.

An *asserted process* is a formula of the form $\{\phi\}\, p\, \{\psi\}$, where $p \in \mathcal{P}^{hl}$ and $\phi, \psi \in \mathcal{C}^{hl}$. Here, ϕ is called the *pre-condition* of the asserted process and ψ is called the *post-condition* of the asserted process.

The intuitive meaning of an asserted process $\{\phi\}\, p\, \{\psi\}$ is as follows: if ϕ holds at the start of p and p eventually terminates successfully, then ψ holds at the successful termination of p. The conditions ϕ and ψ concern the data values assigned to flexible variables at the start and at successful termination, respectively. Therefore, in general, one or more flexible variables occur in ϕ and ψ. Unlike in p, (logical) variables of sort $\mathbf{D}$ may also occur in ϕ and ψ. This allows of referring in ψ to the data values assigned to flexible variables at the start, like in $\{v = u\}\, [v := v + 1]\, \{v = u + 1\}$.

Below, we use the notion of equivalence under V-evaluation to make the intuitive meaning of asserted processes more precise.

We write $FV(p)$, where $p \in \mathcal{P}$, for the set of all $v \in \mathcal{V}$ that occur in p and likewise $FV(\phi)$, where $\phi \in \mathcal{C}^{hl}$, for the set of all $v \in \mathcal{V}$ that occur in ϕ. We write $AFV(p)$, where $p \in \mathcal{P}$, for the set of all $v \in FV(p)$ that occur in subterms of p that are of the form $[v := e]$. Moreover, we write $\mathcal{P}_V$, where V is a finite subset of $\mathcal{V}$, for the set $\{p \in \mathcal{P} \mid FV(p) \subseteq V\}$.

Let V be a finite subset of $\mathcal{V}$ and let $p, q \in \mathcal{P}_V$. Then p and q are *equivalent under V-evaluation*, written $p \overset{V}{\sim} q$, if, for all V-evaluation maps σ, $\mathsf{V}_\sigma(p) = \mathsf{V}_\sigma(q)$ is derivable from the axioms of ACP^*_ϵ-D.

Notice that $\overset{V}{\sim}$, where V be a finite subset of $\mathcal{V}$, is an equivalence relation indeed. Notice further that, for all $p, q \in \mathcal{P}_W$, $W \subset V$ and $p \overset{W}{\sim} q$ only if $p \overset{V}{\sim} q$.

Let $\{\phi\}\, p\, \{\psi\}$ be an asserted process and let $V = FV(\phi) \cup FV(p) \cup FV(\psi)$. Then $\{\phi\}\, p\, \{\psi\}$ is *true* if, for all closed substitution instances $\{\phi'\}\, p\, \{\psi'\}$ of $\{\phi\}\, p\, \{\psi\}$, $\phi' :\rightarrow p \overset{V}{\sim} (\phi' :\rightarrow p) \cdot (\psi' :\rightarrow \epsilon)$.

To justify the claim that the definition given above reflects the intuitive meaning given earlier, we mention that $\phi' :\rightarrow p \overset{V}{\sim} (\phi' :\rightarrow p) \cdot (\psi' :\rightarrow \epsilon)$ only if, for all V-evaluation maps σ, there exists a V-evaluation map σ' such that $\mathsf{V}_\sigma(\phi' :\rightarrow p) \leftrightarrows \mathsf{V}_\sigma(\phi' :\rightarrow p) \cdot \mathsf{V}_{\sigma'}(\psi' :\rightarrow \epsilon)$.

Notice that, using the unary guard operator mentioned in Section 3, we can write $\{\phi'\} \cdot p \overset{V}{\sim} \{\phi'\} \cdot p \cdot \{\psi'\}$ instead of $\phi' :\rightarrow p \overset{V}{\sim} (\phi' :\rightarrow p) \cdot (\psi' :\rightarrow \epsilon)$.

Below, we will present the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. In addition to axioms and rules that concern a particular constant or operator of ACP^*_ϵ-D, there is a rule concerning auxiliary flexible variables and a rule for precondition strengthening and/or postcondition weakening.

We use some special terminology and notations with respect to auxiliary variables. Let $p \in \mathcal{P}^{hl}$, and let $A \subseteq FV(p)$. Then A is a *set of auxiliary variables of p* if each flexible variable in A occurs in p only in subterms of the form $[v := e]$ with $v \in A$. We write $AVS(p)$, where $p \in \mathcal{P}^{hl}$, for the set of all sets of auxiliary variables of p. Moreover, we write p_A, where $p \in \mathcal{P}^{hl}$ and $A \in AVS(p)$, for p with all occurrences of subterms of the form $[v := e]$ with $v \in A$ replaced by ϵ.

The axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ are given in Table 5. In this table, p and q stand for arbitrary terms from $\mathcal{P}^{hl}$, ϕ, ψ, χ, ϕ', and ψ' stand for arbitrary terms from $\mathcal{C}^{hl}$, a stands for an arbitrary basic action from A, v stands for an arbitrary flexible variable from $\mathcal{V}$, and e and $e_1, e_2, \ldots$ stand for arbitrary terms from $\mathcal{D}$. The parallel composition rule may only be applied if the premises are disjoint. Premises $\{\phi\}\, p\, \{\psi\}$ and $\{\phi'\}\, q\, \{\psi'\}$ are *disjoint* if

- $AFV(p) \cap FV(q) = \emptyset$, $AFV(p) \cap FV(\phi') = \emptyset$, and $AFV(p) \cap FV(\psi') = \emptyset$;
- $AFV(q) \cap FV(p) = \emptyset$, $AFV(q) \cap FV(\phi) = \emptyset$, and $AFV(q) \cap FV(\psi) = \emptyset$.

In the consequence rule, the first premise and the last premise are not asserted processes. They assert that $\phi \rightarrow \phi' = \mathsf{t}$ and $\psi' \rightarrow \psi = \mathsf{t}$ are derivable from the axioms of ACP^*_ϵ-D.

Before we move on to the soundness of the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$, we consider two congruence related properties of the equivalences $\overset{V}{\sim}$ that are relevant to the soundness proof.

Table 5. Axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$

inaction axiom:	$\{\phi\}\,\delta\,\{\psi\}$
empty process axiom:	$\{\phi\}\,\epsilon\,\{\phi\}$
basic action axiom:	$\{\phi\}\,a\,\{\phi\}$
data parameterized action axiom:	$\{\phi\}\,a(e_1,\ldots,e_n)\,\{\phi\}$
assignment axiom:	$\{\phi[e/v]\}\,[v := e]\,\{\phi\}$
alternative composition rule:	$\dfrac{\{\phi\}\,p\,\{\psi\},\ \{\phi\}\,q\,\{\psi\}}{\{\phi\}\,p + q\,\{\psi\}}$
sequential composition rule:	$\dfrac{\{\phi\}\,p\,\{\psi\},\ \{\psi\}\,q\,\{\chi\}}{\{\phi\}\,p \cdot q\,\{\chi\}}$
iteration rule:	$\dfrac{\{\phi\}\,p\,\{\phi\},\ \{\phi\}\,q\,\{\psi\}}{\{\phi\}\,p\,^*\,q\,\{\psi\}}$
guarded command rule:	$\dfrac{\{\phi \wedge \psi\}\,p\,\{\chi\}}{\{\phi\}\,\psi :\to p\,\{\chi\}}$
parallel composition rule:	$\dfrac{\{\phi\}\,p\,\{\psi\},\ \{\phi'\}\,q\,\{\psi'\}}{\{\phi \wedge \phi'\}\,p \parallel q\,\{\psi \wedge \psi'\}}$ premises are disjoint
encapsulation rule:	$\dfrac{\{\phi\}\,p\,\{\psi\}}{\{\phi\}\,\partial_H(p)\,\{\psi\}}$
auxiliary variables rule:	$\dfrac{\{\phi\}\,p\,\{\psi\}}{\{\phi\}\,p_A\,\{\psi\}}\ A \in AVS(p),\ FV(\psi) \cap A = \emptyset$
consequence rule:	$\dfrac{\vdash\ \phi \to \phi' = \mathsf{t},\ \{\phi'\}\,p\,\{\psi'\},\ \vdash\ \psi' \to \psi = \mathsf{t}}{\{\phi\}\,p\,\{\psi\}}$

Theorem 3 (Congruence). *For all finite $V \subseteq \mathcal{V}$, for all terms $p, q, p', q' \in \mathcal{P}_V$, $p \overset{V}{\sim} p'$ and $q \overset{V}{\sim} q'$ only if $p+q \overset{V}{\sim} p'+q'$, $p \cdot q \overset{V}{\sim} p' \cdot q'$, and $p^* q \overset{V}{\sim} p'^* q'$. Moreover, for all finite $V \subseteq \mathcal{V}$, for all terms $p, p' \in \mathcal{P}_V$ and all terms $\phi \in \mathcal{C}^{hl}$ with $FV(\phi) \subseteq V$, $p \overset{V}{\sim} p'$ only if $\phi :\to p \overset{V}{\sim} \phi :\to p'$ and $\partial_H(p) \overset{V}{\sim} \partial_H(p')$.*

Proof. Assume $p \overset{V}{\sim} p'$ and $q \overset{V}{\sim} q'$. Then $p + q \overset{V}{\sim} p' + q'$ follows immediately and $p \cdot q \overset{V}{\sim} p' \cdot q'$ and $p\,^*\,q \overset{V}{\sim} p'\,^*\,q'$ follow easily by induction on the number of proper subprocesses of p, where use is made of Lemma 1. Assume $p \overset{V}{\sim} p'$. Then $\phi :\to p \overset{V}{\sim} \phi :\to p'$ follows immediately and $\partial_H(p) \overset{V}{\sim} \partial_H(p')$ follows easily by induction on the number of proper subprocesses of p, where use is made of Lemma 1. □

Theorem 4 (Limited Congruence). *For all finite $V \subseteq \mathcal{V}$, for all terms $p, q, p', q' \in \mathcal{P}_V$ with $AFV(p) \cap FV(q) = \emptyset$, $AFV(q) \cap FV(p) = \emptyset$, $AFV(p') \cap FV(q') = \emptyset$, and $AFV(q') \cap FV(p') = \emptyset$, $p \overset{V}{\sim} p'$ and $q \overset{V}{\sim} q'$ only if $p \parallel q \overset{V}{\sim} p' \parallel q'$.*

Proof. Assume $AFV(p) \cap FV(q) = \emptyset$ and $AFV(q) \cap FV(p) = \emptyset$, $AFV(p') \cap FV(q') = \emptyset$ and $AFV(q') \cap FV(p') = \emptyset$, $p \overset{V}{\sim} p'$ and $q \overset{V}{\sim} q'$. Then $p \parallel q \overset{V}{\sim} p' \parallel q'$ follows easily by induction on the number of proper subprocesses of p, where use is made of Lemma 1. □

Theorem 5 (Soundness). *For all terms $p \in \mathcal{P}^{hl}$, for all terms $\phi, \psi \in \mathcal{C}^{hl}$, the asserted process $\{\phi\}\, p\, \{\psi\}$ is derivable from the axioms and rules of* $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ *only if $\{\phi\}\, p\, \{\psi\}$ is true.*

Proof. We will assume that $\phi, \psi \in \mathcal{C}$. We can do so without loss of generality because, by the definition of the truth of asserted processes, it is sufficient to consider arbitrary closed substitution instances of ϕ and ψ if $\phi, \psi \notin \mathcal{C}$. We will prove the theorem by proving that each of the axioms is true and each of the rules is such that only true conclusions can be drawn from true premises. The theorem then follows by induction on the length of the proof.

The proofs for the axioms and the consequence rule are trivial. Theorems 3 and 4 facilitate the proofs for the other rules. By these theorems, the proofs for the alternative composition rule, the sequential composition rule, and the guarded command rule are also trivial and the proofs for the parallel composition rule, the encapsulation rule, and the auxiliary variables rule are straightforward proofs by induction on the number of proper subprocesses, in which use is made of Lemma 1. The parallel composition rule is proved simultaneously with similar rules for the left merge operator and the communication merge operator. The proof for the iteration rule goes in a less straightforward way.

In case of the iteration rule, we assume that

(1) for all V-evaluation maps σ, $\mathsf{V}_\sigma(\phi :\rightarrow p) = \mathsf{V}_\sigma((\phi :\rightarrow p) \cdot (\phi :\rightarrow \epsilon))$ is derivable;
(2) for all V-evaluation maps σ, $\mathsf{V}_\sigma(\phi :\rightarrow q) = \mathsf{V}_\sigma((\phi :\rightarrow q) \cdot (\psi :\rightarrow \epsilon))$ is derivable;

and we prove that

(3) for all V-evaluation maps σ, $\mathsf{V}_\sigma(\phi :\rightarrow (p \mathbin{^*} q)) = \mathsf{V}_\sigma((\phi :\rightarrow (p \mathbin{^*} q)) \cdot (\psi :\rightarrow \epsilon))$ is derivable;

where $V = FV(\phi) \cup FV(p \mathbin{^*} q) \cup FV(\psi)$. We do so by induction on the number of proper subprocesses of $\mathsf{V}_\sigma(\phi :\rightarrow (p \mathbin{^*} q))$.

The basis step is trivial. The inductive step is proved in the following way. It follows easily from assumption (1), making use of BKS1, that

(4) for all V-evaluation maps σ, for some evaluation map σ', $\mathsf{V}_\sigma(\phi :\rightarrow (p \mathbin{^*} q)) = \mathsf{V}_\sigma(\phi :\rightarrow p) \cdot \mathsf{V}_{\sigma'}(\phi :\rightarrow (p \mathbin{^*} q)) + \mathsf{V}_\sigma(\phi :\rightarrow q)$ is derivable.

We distinguish two cases: $\sigma \neq \sigma'$ and $\sigma = \sigma'$.

In the case where $\sigma \neq \sigma'$, (3) follows easily from (4), the induction hypothesis, and assumption (2), making use of BKS1.

In the case where $\sigma = \sigma'$, it follows immediately from (4), making use of RSP*, that

(5) for all V-evaluation maps σ, $\mathsf{V}_\sigma(\phi :\rightarrow (p \mathbin{^*} q)) = \mathsf{V}_\sigma((\phi :\rightarrow p) \mathbin{^*} (\phi :\rightarrow q))$ is derivable;

and (3) follows easily from (5) and assumption (2), making use of BKS1. □

In the proof of Theorem 5, RSP* is used in the part concerning the iteration rule. We do not see how that part of the proof can be done if RSP* is replaced by BKS2–BKS5.

The following is a corollary of the definition of the truth of asserted processes and Theorem 5.

Corollary 1. *For all terms $p, p' \in \mathcal{P}^{hl}$, for all terms $\phi, \psi \in \mathcal{C}^{hl}$, the asserted process $\{\phi\}\, p\, \{\psi\}$ is derivable from the axioms and rules of* $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ *and $p = p'$ is derivable from the axioms of* ACP^*_ϵ-D *only if $\{\phi\}\, p'\, \{\psi\}$ is true.*

If it is possible at all, equational reasoning with the axioms of a process algebra about how data change in the course of a process is often rather cumbersome. In many cases, but not all, reasoning with the axioms and rules of a Hoare logic is much more convenient. We have not strived for a Hoare logic that covers the cases where it does not simplify reasoning. Actually, the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ are not complete (in the sense of Cook [13]). The side condition of the parallel composition rule precludes completeness. We have, for example, that the asserted process $\{i = 0\}\, [i := i + 1] \cdot [i := i + 1] \parallel [i := 0]\, \{i = 0 \vee i = 1 \vee i = 2\}$ is true, but this cannot be derived by means of the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ alone because a premise of the form $\{\phi\}\, [i := i + 1] \cdot [i := i + 1]\, \{\psi\}$ and a premise of the form $\{\phi'\}\, [i := 0]\, \{\psi'\}$ are never disjoint.

We could have replaced the disjointness side condition by an interference-freedom side condition to cover cases such as the example given above and perhaps this would lead to completeness. However, unless the disjointness side condition would suffice, fulfillment of the interference-freedom side condition generally needs a sophisticated proof. These interference-freedom proofs partly outweigh the advantage of using a Hoare logic for reasoning about how data change in the course of a process. As will be shown by means of an example in Section 7, equational reasoning with the axioms of ACP^*_ϵ-D offers an alternative without interference-freedom proofs. That is why we have chosen for the parallel composition rule with the disjointness side condition.

6 On the Connection between the Hoare Logic and ACP^*_ϵ-D

In this section, we go into the connection of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ with ACP^*_ϵ-D by way of the equivalence relations $\overset{V}{\sim}$.

Let $\{\phi\}\, p\, \{\psi\}$ be an asserted process, and let $V = FV(\phi) \cup FV(p) \cup FV(\psi)$. Suppose that $\{\phi\}\, p\, \{\psi\}$ has been derived from the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. Then, by Theorem 5, $\{\phi\}\, p\, \{\psi\}$ is true. This means that, for all closed substitution instances $\{\phi'\}\, p\, \{\psi'\}$ of $\{\phi\}\, p\, \{\psi\}$, $\phi' :\rightarrow p \overset{V}{\sim} (\phi' :\rightarrow p) \cdot (\psi' :\rightarrow \epsilon)$. In other

words, for all closed substitution instances $\{\phi'\}\,p\,\{\psi'\}$ of $\{\phi\}\,p\,\{\psi\}$, for all V-evaluation maps σ, $\mathsf{V}_\sigma(\phi' :\rightarrow p) = \mathsf{V}_\sigma(\phi' :\rightarrow p) \cdot \mathsf{V}_{\sigma'}(\psi' :\rightarrow \epsilon)$ is derivable from the axioms of ACP^*_ϵ-D. Thus, the derivation of $\{\phi\}\,p\,\{\psi\}$ from the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ has made a collection of equations available that can be considered to be derived by equational reasoning from the axioms of ACP^*_ϵ-D.

Let us have a closer look at the equivalence relation $\overset{V}{\sim}$ on $\mathcal{P}_V$. Clearly, this equivalence relation is useful when reasoning about processes in which data are involved. However, it is plain from the proof of Theorem 4 that $\overset{V}{\sim}$ is not a congruence relation on $\mathcal{P}_V$. This complicates the use of equational reasoning to derive, among other things, the collection of equations referred to above considerably. The presented Hoare logic can be considered to be a means to get partially round the complications concerned.

Dissociated from its connection with $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$, $\overset{V}{\sim}$ remains an interesting equivalence relation on $\mathcal{P}_V$ when it comes to reasoning about processes in which data is involved. Therefore, we mention below a result on this equivalence relation which is a corollary of results from Section 5 used to prove the soundness of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. The fact that $\overset{V}{\sim}$ is not a congruence relation on $\mathcal{P}_V$, and consequently that $\overset{V}{\sim}$ is not preserved by all contexts, makes this corollary to the point. In order to formulate the corollary, we first define a set of contexts, using $\Box$ as a placeholder.

For each finite $V \subseteq \mathcal{V}$, the set $\mathbb{C}^{seq}_V$ of *sequential evaluation supporting contexts for* V is the set $\bigcup_{W \subseteq V} \mathbb{C}^{seq}_{V,W}$, where the sets $\mathbb{C}^{seq}_{V,W}$, for finite $V, W \subseteq \mathcal{V}$ with $W \subseteq V$, are defined by simultaneous induction as follows:

- $\Box \in \mathbb{C}^{seq}_{V,W}$;
- if $p \in \mathcal{P}^{seq}$, $C \in \mathbb{C}^{seq}_{V,W}$, $FV(p) \subseteq V$, and $AFV(p) \subseteq W$, then $p + C,\ C + p,\ p \cdot C,\ C \cdot p,\ p \mathbin{*} C,\ C \mathbin{*} p \in \mathbb{C}^{seq}_{V,W}$;
- if $\phi \in \mathcal{C}$ and $C \in \mathbb{C}^{seq}_{V,W}$, $FV(\phi) \subseteq V$, then $\phi :\rightarrow C \in \mathbb{C}^{seq}_{V,W}$;
- if $p \in \mathcal{P}^{seq}$, $C \in \mathbb{C}^{seq}_{V,W}$, $AFV(p) \cap V = \emptyset$, and $FV(p) \cap W = \emptyset$, then $p \parallel C,\ C \parallel p \in \mathbb{C}^{seq}_{V \cup FV(p), W \cup AFV(p)}$;
- if $H \subseteq \mathsf{A}$ and $C \in \mathbb{C}^{seq}_{V,W}$, then $\partial_H(C) \in \mathbb{C}^{seq}_{V,W}$.

We write $C[p]$, where $C \in \mathbb{C}^{seq}_V$ and $p \in \mathcal{P}$, for C with the occurrence of $\Box$ replaced by p.

The following is a corollary of Theorems 3 and 4.

Corollary 2. *Let V be a finite subset of $\mathcal{V}$. Then, for all $p, p' \in \mathcal{P}_V$, for all $C \in \mathbb{C}^{seq}_V$, $p \overset{V}{\sim} p'$ only if $C[p] \overset{V}{\sim} C[p']$.*

Of course, Corollary 2 can be applied to results from using $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. Let $\{\phi\}\,p\,\{\psi\}$ be an asserted process, let $V = FV(\phi) \cup FV(p) \cup FV(\psi)$, and let $C \in \mathbb{C}^{seq}_V$. Suppose that $\{\phi\}\,p\,\{\psi\}$ has been derived from the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. Then, for all closed substitution instances $\{\phi'\}\,p\,\{\psi'\}$ of $\{\phi\}\,p\,\{\psi\}$, we have that $C[\phi' :\rightarrow p] \overset{V}{\sim} C[(\phi' :\rightarrow p) \cdot (\psi' :\rightarrow \epsilon)]$.

7 On the Role of the Hoare Logic for ACP^*_ϵ-D

Process algebras focus on the main role of a reactive system, namely maintaining some ongoing interaction with its environment. Hoare logics focus on the main role of a transformational system, namely producing, without interruption by its environment, outputs from inputs.[3] However, actual systems are often reactive systems composed of reactive components and transformational components. ACP^*_ϵ-D provides a setting for equational reasoning about the behaviour of such systems, but it does not offer by itself the possibility to reason in Hoare-logic style about the behaviour of the transformational components.

Below, we will take the behaviour of a very simple transformational component and reason about how it changes data both in Hoare-logic style with the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ and equationally with the axioms of ACP^*_ϵ-D. We assume that $\mathfrak{D}$ is the group of integers. We also assume that i and j are flexible variables from $\mathcal{V}$ and n and n' are variables of sort $\mathbf{D}$. Moreover, we use $e - e'$ as an abbreviation of $e + (-e')$. The behaviour of the very simple transformational component concerned is described by the closed ACP^*_ϵ-D term $[i := i + j] \cdot [j := i - j] \cdot [i := i - j]$.

We begin with showing by means of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ that this behaviour swaps the values of i and j. We derive

$$\{i = n \wedge j = n'\}\,[i := i + j]\,\{i = n + n' \wedge j = n'\}$$

using the assignment axiom and the consequence rule. Similarly, we derive

$$\{i = n + n' \wedge j = n'\}\,[j := i - j]\,\{i = n + n' \wedge j = n\}$$

and

$$\{i = n + n' \wedge j = n\}\,[i := i - j]\,\{i = n' \wedge j = n\}\ .$$

From these three asserted processes, we derive

$$\{i = n \wedge j = n'\}\,[i := i + j] \cdot [j := i - j] \cdot [i := i - j]\,\{i = n' \wedge j = n\}$$

using the sequential composition rule twice.

We continue with showing the same by means of ACP^*_ϵ-D. This means that we have to derive from the axioms of ACP^*_ϵ-D, for all $e, e' \in \mathbb{D}$, for all $\{i, j\}$-evaluation maps σ:

$$\begin{aligned} & \mathsf{V}_\sigma((i = e \wedge j = e') :\to [i := i + j] \cdot [j := i - j] \cdot [i := i - j]) \\ (*) \quad & = \mathsf{V}_\sigma((i = e \wedge j = e') \\ & \qquad :\to [i := i + j] \cdot [j := i - j] \cdot [i := i - j] \cdot (i = e' \wedge j = e) :\to \epsilon)\ . \end{aligned}$$

We derive

$$\mathsf{V}_\sigma((i = e \wedge j = e') :\to [i := i + j]) = \sigma(i = e \wedge j = e') :\to [i := \sigma(i + j)]$$

[3] The terms reactive system and transformational system were coined in [16].

using axioms V3 and V5; and

$$\begin{aligned}&\mathsf{V}_\sigma((i = e \wedge j = e') :\to [i := i + j] \cdot (i = e + e' \wedge j = e') :\to \epsilon)\\ &\quad = \sigma(i = e \wedge j = e')\\ &\qquad :\to [i := \sigma(i + j)] \cdot \sigma\{\sigma(e + e')/i\}(i = e + e' \wedge j = e') :\to \epsilon\end{aligned}$$

using axioms V0, V3, and V5.
We can derive the following equation for all $\{i, j\}$-evaluation maps σ:

$$\begin{aligned}&\sigma(i = e \wedge j = e') :\to [i := \sigma(i + j)]\\ (**)\qquad &= \sigma(i = e \wedge j = e')\\ &\qquad :\to [i := \sigma(i + j)] \cdot \sigma\{\sigma(e + e')/i\}(i = e + e' \wedge j = e') :\to \epsilon\ .\end{aligned}$$

In the case $\sigma(i) = e$ and $\sigma(j) = e'$, we derive $\sigma\{\sigma(e+e')/i\}(i = e+e' \wedge j = e') = \mathsf{t}$ using IMP2. From this, we derive equation (**) using axioms GC1 and A8.
In the case $\sigma(i) \neq e$ or $\sigma(j) \neq e'$, we derive $\sigma\{\sigma(e+e')/i\}(i = e+e' \wedge j = e') = \mathsf{f}$ using IMP2. From this, we derive equation (**) using axiom GC2.
Hence, we have for all $\{i, j\}$-evaluation maps σ:

$$\begin{aligned}&\mathsf{V}_\sigma((i = e \wedge j = e') :\to [i := i + j])\\ &\quad = \mathsf{V}_\sigma((i = e \wedge j = e') :\to [i := i + j] \cdot (i = e + e' \wedge j = e') :\to \epsilon)\ .\end{aligned}$$

Similarly, we find for all $\{i, j\}$-evaluation maps σ:

$$\begin{aligned}&\mathsf{V}_\sigma((i = e + e' \wedge j = e') :\to [j := i - j])\\ &\quad = \mathsf{V}_\sigma((i = e + e' \wedge j = e') :\to [j := i - j] \cdot (i = e + e' \wedge j = e) :\to \epsilon)\end{aligned}$$

and

$$\begin{aligned}&\mathsf{V}_\sigma((i = e + e' \wedge j = e) :\to [i := i - j])\\ &\quad = \mathsf{V}_\sigma((i = e + e' \wedge j = e) :\to [i := i - j] \cdot (i = e' \wedge j = e) :\to \epsilon)\ .\end{aligned}$$

From the last three equations, we derive equation (*) using axioms A5, A9, GC5, V3, and V5. By this we have finally shown by means of ACP^*_ϵ-D that the values of i and j are swapped by the process described by $[i{:=}i{+}j]\cdot[j{:=}i{-}j]\cdot[i{:=}i{-}j]$.

In this case, it is clear that Hoare-logic style reasoning with the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ is much more convenient than equational reasoning with the axioms of ACP^*_ϵ-D. Because a single application of a rule of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ cannot be justified by a single application of an axiom of ACP^*_ϵ-D, we expect that this also holds for virtually all other cases of reasoning about how the behaviour of a transformational system changes data.

Now, we turn our attention to the rather restrictive side condition of the parallel composition rule of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$. As mentioned before at the end of Section 5, we have that the asserted process

$$\{i = 0\}\,[i := i + 1] \cdot [i := i + 1] \parallel [i := 0]\,\{i = 0 \vee i = 1 \vee i = 2\}$$

is true, but this cannot be derived by means of the axioms and rules of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ alone because a premise of the form $\{\phi\}\,[i := i+1] \cdot [i := i+1]\,\{\psi\}$ and a premise of the form $\{\phi'\}\,[i := 0]\,\{\psi'\}$ are never disjoint. However, we can derive the following equation from the axioms of ACP^*_ϵ-D:

$$\begin{aligned}
&[i := i+1] \cdot [i := i+1] \parallel [i := 0] \\
&\quad = [i := i+1] \cdot ([i := i+1] \cdot [i := 0] + [i := 0] \cdot [i := i+1]) \\
&\qquad + [i := 0] \cdot [i := i+1] \cdot [i := i+1] \;.
\end{aligned}$$

By Corollary 1, it is sound to replace in the above asserted process the left-hand side of this equation by the right-hand side of this equation. This yields the asserted process

$$\begin{aligned}
&\{i = 0\} \\
&\{[i := i+1] \cdot ([i := i+1] \cdot [i := 0] + [i := 0] \cdot [i := i+1]) \\
&\quad + [i := 0] \cdot [i := i+1] \cdot [i := i+1]\} \\
&\{i = 0 \mathbin{\vee} i = 1 \mathbin{\vee} i = 2\} \;,
\end{aligned}$$

which can be derived using the assignment axiom, the alternative composition rule, the sequential composition rule, and the consequence rule of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ several times.

If the disjointness side condition of the parallel composition rule of $\mathrm{HL}_{\mathrm{ACP}^*_\epsilon\text{-D}}$ is replaced by an interference-freedom side condition, like in [24], then the original asserted process becomes derivable using the axioms and rules of the Hoare logic alone (see e.g. [1, page 278]). The interference-freedom proof involved needs proof outlines (see [24]) for $\{i = 0\}\,[i := i+1] \cdot [i := i+1]\,\{\mathsf{t}\}$ and $\{\mathsf{t}\}\,[i := 0]\,\{i = 0 \mathbin{\vee} i = 1 \mathbin{\vee} i = 2\}$. In this very simple case, the interference-freedom proof already amounts to seven interference-freedom checks. However, for two processes in which k and k' assignment actions occur, the number of interference-freedom checks is at least $2 \cdot k \cdot k' + k + k'$. Therefore, we expect that interference-freedom proofs partly outweigh the advantage of using a Hoare logic.

8 Related Work

The approach to the formal verification of programs that is now known as Hoare logic was proposed in [17]. The illustration of this approach was at the time confined to the very simple deterministic sequential programs that are mostly referred to as while programs (cf. [1]). The axioms, the sequential composition rule, the iteration rule, the guarded command rule, and the consequence rule from our Hoare logic savour strongly of the common rules for while programs. The alternative composition rule is the or rule due to [21], the parallel composition rule was proposed in [18], and the auxiliary variables rule was first introduced in [24]. The parallel composition rules proposed in [2,22,24] are more complicated than our parallel composition rule.

In the case of [2,22], the intention was to provide a Hoare logic for the first design of CSP [19]. In that design, one program may force another program to assign a data value sent by the former program to a program variable used by the latter program. This feature complicates the parallel composition rule considerably. Moreover, incorporating this feature in an ACP-like process algebra would lead to the situation that, in equational reasoning, certain axioms may not be applied in contexts of parallel processes (like in [15], see below). Because our concern is in the use of a Hoare logic as a complement to pure equational reasoning, we have not considered incorporating this feature.

In the case of [24], the rule is more complicated because, in the parallel programs covered, program variables may be shared variables, i.e. program variables that are assigned to in one program may be used in another program. Our process algebra also covers shared variables. However, covering shared variables in our Hoare logic as well would mean that the simple disjointness proof required by our parallel composition rule has to be replaced a sophisticated interference-freedom proof. We believe that this would diminish the usefulness of our Hoare logic as a complement to equational reasoning considerably. Therefore, we have not considered covering shared variables in the parallel composition rule.

In [15], an extension of ACP with the empty process constant and the unary counterpart of the binary guarded command operator is presented, the truth of an asserted sequential process is defined in terms of the transition relations from the given structural operational semantics of the presented extension of ACP, and it is shown that an asserted sequential process $\{\phi\}\,p\,\{\psi\}$ is true according to that definition iff $\{\phi\}\cdot p \leftrightarrows' \{\phi\}\cdot p\cdot\{\psi\}$, where $\leftrightarrows'$ is bisimulation equivalence as defined in [15] for sequential processes. Moreover, a Hoare logic of sequential asserted processes is presented and its soundness is shown. However, [15] does not go into the use of that Hoare logic as a complement to pure equational reasoning from the equational axioms.

Regarding the bisimulation equivalence $\leftrightarrows'$ defined in [15] for sequential processes, we can mention that, if the data-states are evaluation maps, $p \leftrightarrows' q$ iff $\mathsf{V}_\sigma(p) \leftrightarrows \mathsf{V}_\sigma(q)$ for all V-evaluation maps σ, where $V = FV(p) \cup FV(q)$. Due to the possibility of interference between parallel processes, a different bisimulation equivalence $\leftrightarrows''$, finer than $\leftrightarrows'$, is needed in [15] for parallel processes. As a consequence, in equational reasoning, certain axioms may not be applied in contexts of parallel processes. Moreover, $\leftrightarrows$ together with the operators V_σ allows of dealing with local data-states, whereas the combination of $\leftrightarrows'$ and $\leftrightarrows''$ does not allow of dealing with local data-states.

9 Concluding Remarks

We have taken an extension of ACP with features that are relevant to processes in which data are involved, devised a Hoare logic of asserted processes based on this extension of ACP, and gone into the use of this Hoare logic as a complement to pure equational reasoning from the axioms of the extension of ACP.

We have defined what it means that an asserted process is true in terms of an equivalence relation ($\overset{V}{\sim}$) that had been found to be central to relating the extension of ACP and the Hoare logic. That this equivalence relation is not a congruence relation with respect to parallel composition is related to the fact that in the extension of ACP presented in [15] certain axioms may not be applied in contexts of parallel processes.

In this paper, we build on earlier work on ACP. The axioms of ACP_ϵ have been taken from [7, Section 4.4], the axioms for the iteration operator have been taken from [9], and the axioms for the guarded command operator have been taken from [4]. The evaluation operators have been inspired by [11] and the data parameterized action operator has been inspired by [12].

References

1. Apt, K.R., de Boer, F.S., Olderog, E.R.: Verification of Sequential and Concurrent Programs. Texts in Computer Science, Springer-Verlag, Berlin, third edn. (2009)
2. Apt, K.R., Francez, N., de Roever, W.P.: A proof system for communicating sequential processes. ACM Transactions on Programming Languages and Systems 2(3), 359–385 (1980)
3. Baeten, J.C.M., Bergstra, J.A.: Global renaming operators in concrete process algebra. Information and Control 78(3), 205–245 (1988)
4. Baeten, J.C.M., Bergstra, J.A.: Process algebra with signals and conditions. In: Broy, M. (ed.) Programming and Mathematical Methods. NATO ASI Series, vol. F88, pp. 273–323. Springer-Verlag (1992)
5. Baeten, J.C.M., Bergstra, J.A.: Process algebra with propositional signals. Theoretical Computer Science 177, 381–405 (1997)
6. Baeten, J.C.M., Verhoef, C.: A congruence theorem for structured operational semantics with predicates. In: Best, E. (ed.) CONCUR'93. Lecture Notes in Computer Science, vol. 715, pp. 477–492. Springer-Verlag (1993)
7. Baeten, J.C.M., Weijland, W.P.: Process Algebra, Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, Cambridge (1990)
8. Bergstra, J.A., Bethke, I., Ponse, A.: Process algebra with iteration and nesting. Computer Journal 37, 243–258 (1994)
9. Bergstra, J.A., Fokkink, W.J., Ponse, A.: Process algebra with recursive operations. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, pp. 333–389. Elsevier, Amsterdam (2001)
10. Bergstra, J.A., Klop, J.W.: Process algebra for synchronous communication. Information and Control 60(1–3), 109–137 (1984)
11. Bergstra, J.A., Middelburg, C.A.: Splitting bisimulations and retrospective conditions. Information and Computation 204(7), 1083–1138 (2006)
12. Bergstra, J.A., Middelburg, C.A.: A process calculus with finitary comprehended terms. Theory of Computing Systems 53(4), 645–668 (2013)
13. Cook, S.A.: Soundness and completeness of an axiom system for program verification. SIAM Journal of Computing 7(1), 70–90 (1978)
14. Fokkink, W.J.: On the completeness of the equations for the kleene star in bisimulation. In: Wirsing, M., Nivat, M. (eds.) AMAST 96. Lecture Notes in Computer Science, vol. 1101, pp. 180–194. Springer-Verlag (1996)
15. Groote, J.F., Ponse, A.: Process algebra with guards: Combining Hoare logic with process algebra. Formal Aspects of Computing 6(2), 115–164 (1994)

16. Harel, D., Pnueli, A.: On the development of reactive systems. In: Apt, K. (ed.) Logics and Models of Concurrent Systems. NATO ASI Series, vol. F13, pp. 477–498. Springer-Verlag (1985)
17. Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM 12(10), 576–580, 583 (1969)
18. Hoare, C.A.R.: Towards a theory of parallel programming. In: Hoare, C.A.R., Perrott, R.H. (eds.) Operating Systems Techniques. pp. 61–71. Academic Press (1972)
19. Hoare, C.A.R.: Communicating sequential processes. Communications of the ACM 21(8), 666–677 (1978)
20. Lamport, L.: The temporal logic of actions. ACM Transactions on Programming Languages and Systems 16(3), 872–923 (1994)
21. Lauer, P.E.: Consistent formal theories of the semantics of programming languages. Technical Report 25.121, IBM Laboratory Vienna (1971)
22. Levin, G.M., Gries, D.: A proof technique for communicating sequential processes. Acta Informatica 15(3), 281–302 (1981)
23. Milner, R.: A complete inference system for a class of regular behaviours. Journal of Computer and System Sciences 28(3), 439–466 (1984)
24. Owicki, S., Gries, D.: An axiomatic proof technique for parallel programs I. Acta Informatica 6(4), 319–340 (1976)
25. Pigozzi, D., Salibra, A.: The abstract variable-binding calculus. Studia Logica 55(1), 129–179 (1995)
26. 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)
27. Schneider, F.B.: On Concurrent Programming. Graduate Texts in Computer Science, Springer-Verlag, Berlin (1997)
28. Sewell, P.: Bisimulation is not finitely (first order) equationally axiomatisable. In: LICS'94. pp. 62–70. IEEE Computer Society Press (1994)
29. Wirsing, M.: Algebraic specification. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 675–788. Elsevier, Amsterdam (1990)