Notre Dame Journal of Formal Logic
Volume 39, Number 4, Fall 1998

Bochvar-McCarthy Logic and Process Algebra

JAN A. BERGSTRA and ALBAN PONSE

Abstract We propose a combination of Bochvar's strict three-valued logic, McCarthy's sequential three-valued logic, and process algebra via the *conditional guard construct*. This combination entails the introduction of a new constant *meaningless* in process algebra. We present an operational semantics in SOS-style, and a completeness result for ACP with conditional guard construct and the proposed logic.

1 Introduction An (immediate) error in an algorithm or program, such as reference to a nonexisting instruction, or a type clash, is often easily detectable. In order to model this feature in a concurrent setting, we consider process algebra with conditional guard construct and a variant of *three-valued logic* as a means to represent concurrent algorithms and programs. (Some motivation is given in Section 3.) In general, errors can be classified in at least two categories: divergencies which can be hard to detect and more simple ones, such as described above. In this paper we propose how to deal with the occurrence of the latter sort, which we further call *meaningless*, notation M. In particular, evaluation of a proposition φ may now lead to M, in which case the evaluation of $\neg\varphi$ should of course also result in M. Thus the first logical identity we adopt is $\neg\mathsf{M} = \mathsf{M}$.

In process algebra we introduce μ as a process representing the effect of the logical (error-)value M. The new constant μ is axiomatized[1] by

$$\begin{aligned} x + \mu &= \mu, \\ \mu \cdot x &= \mu. \end{aligned}$$

Here $+$ is the commutative operation denoting choice, and $\cdot$ represents sequential composition. So the process μ ruins each (future) alternative.

We recall the *conditional guard construct* $\varphi :\rightarrow \_$ from Dijkstra [13] (roughly: *if* φ *holds, then* $\_$), which was introduced in process algebra with two-valued logic in Baeten and Bergstra [2] with the following typical laws where T denotes the value

Received January 12, 1998; revised December 2, 1999

true and F stands for *false*:

$$\begin{aligned} \mathsf{T} :\to x &= x, \\ \mathsf{F} :\to x &= \delta, \\ \varphi :\to x + \psi :\to x &= \varphi \vee \psi :\to x. \end{aligned}$$

The constant δ (inaction/deadlock) is well known in ACP-based approaches (see, e.g., Bergstra and Klop [6, 7] and Baeten and Wijland [3]) and is axiomatized by $x + \delta = x$ and $\delta \cdot x = \delta$. Another basic construct in the combination of two-valued propositional logic and process algebra is *conditional composition* $x \lhd \varphi \rhd y$, introduced in [2]. Here x, y are processes, and φ is a proposition. This operation satisfies (among others) the following axioms:

$$\begin{aligned} x \lhd \mathsf{T} \rhd y &= x, \\ x \lhd \mathsf{F} \rhd y &= y, \\ x \lhd \varphi \rhd y &= y \lhd \neg\varphi \rhd x. \end{aligned}$$

The notation $\_ \lhd \_ \rhd \_$ stems from Hoare et. al [17], and in that paper it is argued that $x \lhd \varphi \rhd y$ expresses **if** φ **then** x **else** y **fi**. Conditional composition $\_ \lhd \_ \rhd \_$ can be regarded as more basic than the conditional guard construct, as it does not presuppose the existence of the special constant δ. Of course,

$$x \lhd \varphi \rhd y = \varphi :\to x + \neg\varphi :\to y.$$

Finally, a characteristic identity for process algebra with *two-valued* logic is $x \lhd \varphi \rhd x = x$.

We fix the relation between M and μ with axiom

$$\mathsf{M} :\to x = \mu.$$

If it is *known* that a condition in a process term equals M, there is no point in considering any (future) alternative. Furthermore, preservation of the three laws on the conditional guard construct mentioned above and those for δ and μ implies symmetry of $\vee$. This symmetry together with $\neg\mathsf{M} = \mathsf{M}$ and the derivations

$$\begin{aligned} \mathsf{M} :\to x &= \mu \\ &= x + \mu \\ &= \mathsf{T} :\to x + \mathsf{M} :\to x \\ &= \mathsf{T} \vee \mathsf{M} :\to x, \end{aligned}$$

and

$$\begin{aligned} \mathsf{M} :\to x &= \mu \\ &= \mu + \delta \\ &= \mathsf{M} :\to x + \mathsf{F} :\to x \\ &= \mathsf{M} \vee \mathsf{F} :\to x, \end{aligned}$$

imply the following truth tables:

x	$\neg x$
M	M
T	F
F	T

$\vee$	M	T	F
M	M	M	M
T	M	T	T
F	M	T	F

This three-valued logic, in which M is totally persistent, was defined earlier by Bochvar in [11].

Another basic law in [2] that we want to accommodate relates to repeated application of the conditional guard construct and conjunction:

$$\varphi :\to (\psi :\to x) = \varphi \wedge \psi :\to x$$

(note the symmetry in $\varphi \wedge \psi$). However, this law is not preserved in the present setting:[2]

$$\begin{aligned} \mathsf{F} :\to (\mathsf{M} :\to x) &= \delta, \\ \mathsf{F} \wedge \mathsf{M} :\to x &= \mu. \end{aligned}$$

Therefore we replace it by a version in which also the right-hand side reflects the *order* of evaluation, and use *left-sequential* conjunction as introduced by McCarthy [20], with the asymmetric notation $\mathbin{\wedge\!\!\circ}$ taken from Bergstra, Bethke, and Rodenburg [5]:

$\mathbin{\wedge\!\!\circ}$	M	T	F
M	M	M	M
T	M	T	F
F	F	F	F

Here $\varphi \mathbin{\wedge\!\!\circ} \psi$ expresses that *first* φ is evaluated and *then* ψ. For recent work on McCarthy's logic see, for example, Konikowska [19]. A sequential version of the law mentioned above is

$$\varphi :\to (\psi :\to x) = \varphi \mathbin{\wedge\!\!\circ} \psi :\to x.$$

We further adopt this identity as the process algebra axiom that reduces repeated application of the conditional guard construct[3]. This design also allows us to extend the framework defined in this paper in a conservative way to a setting with four-valued logic as introduced in [5]. In that paper, complete axiomatizations for both Bochvar's and McCarthy's three-valued logic can be found.

In the next section we present the precise three-valued logic we consider and extend it with proposition symbols. In Section 3 we combine this extension with ACP. In Sections 4 and 5 we define an operational semantics and bisimulation equivalence and we prove a (relative) completeness result. In Section 6 we extend the setting with data-parametric actions and consider some examples. Finally, in Section 7 we provide some conclusions.

2 A three-valued propositional logic due to Bochvar with McCarthy's extension

We consider the following set of logical operations on the set $\mathbb{T}_3^{\mathsf{M}}$ of truth values:

$$\begin{aligned} \mathsf{M}, \mathsf{T}, \mathsf{F} : &\ \to \mathbb{T}_3^{\mathsf{M}} \\ \neg : &\ \mathbb{T}_3^{\mathsf{M}} \to \mathbb{T}_3^{\mathsf{M}} \\ \wedge, \vee, \mathbin{\wedge\!\!\circ}, \mathbin{\circ\!\!\vee}, \mathbin{\wedge\!\!\circ}, \mathbin{\vee\!\!\circ} : &\ \mathbb{T}_3^{\mathsf{M}} \times \mathbb{T}_3^{\mathsf{M}} \to \mathbb{T}_3^{\mathsf{M}} \end{aligned}$$

of which $\neg$, $\wedge$, and $\overset{\circ}{\wedge}$ are defined by the following truth tables:

x	$\neg x$
M	M
T	F
F	T

$\wedge$	M	T	F
M	M	M	M
T	M	T	F
F	M	F	F

$\overset{\circ}{\wedge}$	M	T	F
M	M	M	M
T	M	T	F
F	F	F	F

The remaining operations are all definable:

$$\begin{aligned}
&\textit{Disjunction}: && x \vee y \overset{\text{def}}{=} \neg(\neg x \wedge \neg y),\\
&\textit{Left-sequential disjunction}: && x \overset{\circ}{\vee} y \overset{\text{def}}{=} \neg(\neg x \overset{\circ}{\wedge} \neg y),\\
&\textit{Right-sequential conjunction}: && x \wedge_{\circ} y \overset{\text{def}}{=} y \overset{\circ}{\wedge} x,\\
&\textit{Right-sequential disjunction}: && x \vee^{\circ} y \overset{\text{def}}{=} y \overset{\circ}{\vee} x.
\end{aligned}$$

We call the resulting logic $\mathbb{BM}_3(\neg, \wedge, \overset{\circ}{\wedge})$, or shortly $\mathbb{BM}_3$. Notice that $\mathbb{BM}_3$ is not functionally complete: for example, one cannot define f with $f(\mathsf{M}) = \mathsf{F}$. We do not embark on complete equational specifications of $\mathbb{BM}_3$. All we need are the truth tables and equations above (so all in: $3+9+9=21$ equations, or 25 if we include the dual and right-sequential operations).

In case we use proposition symbols from set $\mathbb{P}$, we shall write $\mathbb{BM}_3(\mathbb{P})$, and for concise notation we shall identify $\mathbb{BM}_3$ and $\mathbb{BM}_3(\varnothing)$. In order to extend our evaluation system to propositions $\varphi, \psi, \ldots$ over $\mathbb{P}$, we use substitution on single proposition symbols: let $p, q \in \mathbb{P}$, then

$$\begin{aligned}
[\varphi/p]q &\overset{\text{def}}{=} q,\\
[\varphi/p]p &\overset{\text{def}}{=} \varphi,\\
[\varphi/p]c &\overset{\text{def}}{=} c \text{ for } c \in \{\mathsf{M}, \mathsf{T}, \mathsf{F}\},\\
[\varphi/p]\neg\psi &\overset{\text{def}}{=} \neg[\varphi/p]\psi,\\
[\varphi/p](\psi_1 \diamond \psi_2) &\overset{\text{def}}{=} [\varphi/p]\psi_1 \diamond [\varphi/p]\psi_2 \text{ for } \diamond \in \{\wedge, \vee, \overset{\circ}{\wedge}, \overset{\circ}{\vee}, \wedge_{\circ}, \vee^{\circ}\},
\end{aligned}$$

and as a proof rule the *excluded fourth rule*:

$$\frac{\sigma(\varphi) = \sigma(\psi) \quad \text{for all } \sigma \in \{[\mathsf{M}/p], [\mathsf{T}/p], [\mathsf{F}/p]\}}{\varphi = \psi}.$$

Together with the equations implied by the truth tables for $\neg$, $\wedge$, and $\overset{\circ}{\wedge}$, this yields a complete, inequational evaluation system for $\mathbb{BM}_3(\mathbb{P})$. Notice that the operations $\wedge$, $\overset{\circ}{\wedge}$, and their duals are associative and that $\wedge$ and $\vee$ are commutative as well. We write

$$\mathbb{BM}_3(\mathbb{P}) \models \varphi = \psi$$

if $\varphi = \psi$ can be proved by the system described above. We use the satisfaction symbol $\models$ to indicate that our system is just a syntactic version of the standard semantics for three-valued logic. If $\mathbb{P}$ is fixed, we often only write $\models \varphi = \psi$. The identities stated in the following lemma are used in the sequel.

Lemma 2.1 *The following identities hold in* $\mathbb{BM}_3(\mathbb{P})$*:*

1. $\models (\varphi \mathbin{\diamond\!\!\wedge} \mathsf{F}) \vee \mathsf{T} = \varphi \vee \mathsf{T}$,
2. $\models (\varphi \vee \mathsf{T}) \mathbin{\diamond\!\!\wedge} \varphi = \varphi$.

3 Process algebra with $\mathbb{BM}_3(\mathbb{P})$ In this section we consider the combination of process algebra and $\mathbb{BM}_3(\mathbb{P})$. This combination is based on ACP, the Algebra of Communicating Processes [6, 7, 3]. The signature of ACP is parameterized with a set A of constants $a, b, c, \ldots$ denoting atomic actions, that is, processes that are not subject to further division and that execute in finite time, and with a communication function γ that prescribes which actions can communicate. We consider a distinct action $t \notin A$, and set $A_t = A \cup \{t\}$. We further write $\mathrm{ACP}(A_t, \gamma)$ as to make these parameters explicit. It is assumed that γ is commutative, thus $\gamma(a, b) = \gamma(b, a)$, and associative: $\gamma(a, \gamma(b, c)) = \gamma(\gamma(a, b), c)$. In $\mathrm{ACP}(A_t, \gamma)$ there is a constant $\delta \notin A_t$, denoting the inactive process. The six operations of $\mathrm{ACP}(A_t, \gamma)$ are:

Alternative composition:	$x + y$ denotes the process that performs either x or y.
Sequential composition:	$x \cdot y$ denotes the process that performs x, and upon completion of x starts with y.
Merge or *parallel composition*:	$x \parallel y$ denotes the parallel execution of x and y (including the possibility of synchronization).
Left merge, an auxiliary operator:	$x \mathbin{\underline{\parallel}} y$ denotes $x \parallel y$ with the restriction that the first action stems for the left argument x.
Communication merge, an auxiliary operator:	$x \mid y$ denotes $x \parallel y$ with the restriction that the first action is a synchronization of both x and y.
Encapsulation:	$\partial_H(x)$ (where $H \subseteq A$) renames atoms in H to δ.

In Table 1 we present a slight modification of $\mathrm{ACP}(A_t, \gamma)$. We take γ *total* on $A_t \times A_t \to A_{t\delta}$, where $A_{t\delta} = A_t \cup \{\delta\}$, and we take the communication merge commutative (CMC) (by which (CM6) and (CM9), the symmetric variants of (CM5) and (CM8), see [3], become derivable). We note that left merge and communication merge are auxiliary operations used to axiomatize the merge (cf. [3]). By (A1) and (CMC), merge is a commutative operation. Although the merge is not axiomatized as an associative operation, it is associative for all process terms (i.e., closed terms, which can be proved with structural induction), and we will leave out brackets in repeated applications.

We shortly comment on the primitives of $\mathrm{ACP}(A_t, \gamma)$. Often, $+$ is used as an operation facilitating analysis rather than as a specification primitive: concurrency is analyzed in terms of sequential composition, choice, and communication. Verification of a concurrent system $\partial_H(C_1 \parallel \ldots \parallel C_n)$ generally boils down to representing the possible executions with $+$ and $\cdot$, having applied left-merge , communica-

Table 1: The axiom system ACP(A_t, γ), where $a, b \in A_{t\delta}$, $H \subseteq A_t$.

(A1)	$x+(y+z)$	$=$	$(x+y)+z$
(A2)	$x+y$	$=$	$y+x$
(A3)	$x+x$	$=$	x
(A4)	$(x+y)z$	$=$	$xz+yz$
(A5)	$(xy)z$	$=$	$x(yz)$
(A6)	$x+\delta$	$=$	x
(A7)	δx	$=$	δ
(CF1)	$a \mid b$	$=$	$\gamma(a,b)$ if $a, b \in A_t$
(CF2)	$a \mid \delta$	$=$	δ
(CM1)	$x \parallel y$	$=$	$(x \mathbin{\underline{\parallel}} y + y \mathbin{\underline{\parallel}} x) + x \mid y$
(CM2)	$a \mathbin{\underline{\parallel}} x$	$=$	ax
(CM3)	$ax \mathbin{\underline{\parallel}} y$	$=$	$a(x \parallel y)$
(CM4)	$(x+y) \mathbin{\underline{\parallel}} z$	$=$	$x \mathbin{\underline{\parallel}} z + y \mathbin{\underline{\parallel}} z$
(CMC)	$x \mid y$	$=$	$y \mid x$
(CM5)	$ax \mid b$	$=$	$(a \mid b)x$
(CM7)	$ax \mid by$	$=$	$(a \mid b)(x \parallel y)$
(CM8)	$(x+y) \mid z$	$=$	$x \mid z + y \mid z$
(D1)	$\partial_H(a)$	$=$	a if $a \notin H$
(D2)	$\partial_H(a)$	$=$	δ if $a \in H$
(D3)	$\partial_H(x+y)$	$=$	$\partial_H(x) + \partial_H(y)$
(D4)	$\partial_H(xy)$	$=$	$\partial_H(x)\partial_H(y)$

tion merge, and encapsulation (by which communication between components C_i can be enforced). After renaming internal activity (e.g., using pre-abstraction explained below), this may yield a simple and informative specification of external behavior. Furthermore, in case such a component *reacts* upon external input, choice is a natural specification primitive for representing value-binding (cf. Milner's translation of basic CCS into value-passing CCS [21]). For example, assume component C_1 can receive a value from a finite set $Data = \{d_0, d_1, \ldots, d_N\}$, and then perform further activity that depends on the value d received via action $r(d)$. By commutativity and associativity of $+$, this situation can be characterized by the identity

$$C_1 = r(d_0) \cdot Activity(d_0) + r(d_1) \cdot Activity(d_1) + \cdots + r(d_N) \cdot Activity(d_N),$$

where $\cdot$ binds stronger than $+$, or shortly by

$$C_1 = \sum_{d \in Data} r(d) \cdot Activity(d).$$

In a parallel context in which some value $s(d_j)$ is offered, the intended communication $\gamma(r(d_j), s(d_j))$ can be enforced by encapsulation, after which C_1 has evolved

into *Activity*(d_j). After introducing the remaining axioms on μ and the conditional operations, we continue this explanation.

Table 2: Remaining axioms of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$, where $\varphi, \psi \in \mathbb{BM}_3(\mathbb{P})$, $a, b \in A_{t\delta}$, $I \subseteq A_t$.

(M1)	$x + \mu$	$=$	μ
(M2)	$\mu \cdot x$	$=$	μ
(M3)	$\mu \mid x$	$=$	μ
(GT)	$\mathsf{T} :\!\rightarrow x$	$=$	x
(GF)	$\mathsf{F} :\!\rightarrow x$	$=$	δ
(GM)	$\mathsf{M} :\!\rightarrow x$	$=$	μ
(Cond)	$x \lhd \varphi \rhd y$	$=$	$\varphi :\!\rightarrow x + \neg\varphi :\!\rightarrow y$
(GC1)	$\varphi :\!\rightarrow x + \psi :\!\rightarrow x$	$=$	$\varphi \vee \psi :\!\rightarrow x$
(GC2)	$\varphi :\!\rightarrow x + \varphi :\!\rightarrow y$	$=$	$\varphi :\!\rightarrow (x + y)$
(GC3)	$(\varphi :\!\rightarrow x)y$	$=$	$\varphi :\!\rightarrow xy$
(GCL4)	$\varphi :\!\rightarrow (\psi :\!\rightarrow x)$	$=$	$\varphi \mathbin{\diamond\!\!\wedge} \psi :\!\rightarrow x$
(GC5)	$\varphi :\!\rightarrow x \between y$	$=$	$\varphi :\!\rightarrow (x \between y)$
(GCM6)	$\varphi :\!\rightarrow a \mid \psi :\!\rightarrow b$	$=$	$\varphi \wedge \psi :\!\rightarrow a \mid b$
(GCM7)	$\varphi :\!\rightarrow ax \mid \psi :\!\rightarrow b$	$=$	$\varphi \wedge \psi :\!\rightarrow (a \mid b)x$
(GCM8)	$\varphi :\!\rightarrow ax \mid \psi :\!\rightarrow by$	$=$	$\varphi \wedge \psi :\!\rightarrow (a \mid b)(x \parallel y)$
(DGC)	$\partial_H(\varphi :\!\rightarrow x)$	$=$	$\varphi :\!\rightarrow \partial_H(x)$
(TGC)	$t_I(\varphi :\!\rightarrow x)$	$=$	$\varphi :\!\rightarrow t_I(x)$
(T1)	$t_I(a)$	$=$	a if $a \notin I$
(T2)	$t_I(a)$	$=$	t if $a \in I$
(T3)	$t_I(x + y)$	$=$	$t_I(x) + t_I(y)$
(T4)	$t_I(xy)$	$=$	$t_I(x)t_I(y)$

In Table 2 we provide the additional axioms for the extension of $\mathrm{ACP}(A_t, \gamma)$ with $\mathbb{BM}_3(\mathbb{P})$ (where $\mathbb{P}$ is considered a third parameter). Here φ is taken from $\mathbb{BM}_3(\mathbb{P})$, so for each φ, $\_ \lhd \varphi \rhd \_$ is a binary operation and $\varphi :\!\rightarrow \_$ is a unary operation. The axiom (GCM6) suggests a more general version of (CF1)–(CF2), and (GCM7) and (GCM8) can be seen as generalizations of (CM5) and (CM7), respectively. Furthermore, observe that

$$\varphi :\!\rightarrow x \mid \psi :\!\rightarrow y = \varphi \wedge \psi :\!\rightarrow (x \mid y)$$

would imply inconsistency of our theory ($\mu = \mathsf{T} :\!\rightarrow \mu \mid \mathsf{F} :\!\rightarrow x = \mathsf{T} \wedge \mathsf{F} :\!\rightarrow (\mu \mid x) = \delta$), which explains the weaker axioms (GCM6)–(GCM8). For each $I \subset A_t$ there is a pre-abstraction operation t_I that renames all actions in I to t and that is axiomatized by (T1)–(T4). We use

$$\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$$

to refer to both this axiom system and the signature thus defined. We mostly suppress the $\cdot$ in terms, and brackets according to the following rules: $\cdot$ binds strongest, $:\!\rightarrow$ binds stronger than $\|$, $\underline{\|}$, $|$, all of which in turn bind stronger than $+$. Closed terms over $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ will be further called *process terms*, as these represent processes, and $\mathcal{P}(A_t, \gamma, \mathbb{P})$, or shortly $\mathcal{P}$ if all parameters are fixed, denotes the set of all process terms.

We continue our explanation, now involving all operations of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ and binary Kleene star: assume component C_1 as introduced above models a channel that repeatedly transfers values from port 1 to port 2 via actions $r_1(d)$ (receive value d along channel 1), and $s_2(d)$ (send value d along channel 2). We can write

$$C_1 = \left(\textstyle\sum_{d\in Data} r_1(d)\cdot s_2(d)\right)^* \delta.$$

Here * is the binary Kleene star [18], defined by

$$x^*y = x(x^*y) + y.$$

(See also Bergstra, Bethke, and Ponse [4].) In particular, $x^*\delta$ repeatedly performs x, as follows easily from the axioms A6 and A7 and can be defined by $x^\omega = x(x^\omega)$, thus without $+$ and δ (see Fokkink [14]). We further refine C_1 to a component *Ch*, representing a channel that either corrupts a value received or transfers it properly: this can be written as

$$Ch = \left(\textstyle\sum_{d\in Data} r_1(d)\cdot (s_2(d) \lhd \varphi \rhd s_2(\bot)\right)^* \delta,$$

where φ represents the mechanism that determines whether d will be transferred properly or will be corrupted (action $s_2(\bot)$). It may well be that φ depends on courses of evaluation beyond our means of analysis or control or beyond our verification purposes. In this case, our analysis should be performed on a more abstract level and we can write

$$\left(\textstyle\sum_{d\in Data} r_1(d)\cdot (s_2(d) + s_2(\bot)\right)^* \delta,$$

where $+$ is used in favor of conditional composition. A better type of modeling is obtained by specifying

$$\left(\textstyle\sum_{d\in Data} r_1(d)\cdot (t\cdot s_2(d) + t\cdot s_2(\bot)\right)^* \delta,$$

thus guaranteeing that corruption of transfer is nondeterministic from the external point of view. The latter specification is useful for analysis of the operation of *Ch* in the case that no explicit information on φ is available. Finally, we give an example that illustrates that even in case a condition is explicit, it need not be relevant for verification purposes. Consider the program fragment

while φ **do**
  while φ **do**
    P
  od
od ;
Q

which of course equals **while** φ **do** P **od** ; Q, irrespective of φ. This fragment can be specified as

$$R^*(R^*S)$$

with $R = \varphi :\to P$ and $S = \neg\varphi :\to Q$. Using * and $+$, and their axioms, one easily derives $R^*(R^*S) = R^*S$ (this is an example in [4]), proving our claim. This completes our explanation of the primitives of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$.

In the remainder of the paper, we further develop the framework for the case in which conditions play a decisive role and can in particular be evaluated as meaningless. Note that in this case, $x \lhd \varphi \rhd x \neq x$, thus the principle of the excluded middle—*tertium non datur*—is dropped.

In order to use identities in three-valued logic in process algebra, we introduce the 'rule of equivalence'

$$\text{(ROE)} \qquad \frac{\models \varphi = \psi}{\vdash \varphi :\to x = \psi :\to x}$$

This rule reflects the 'rule of consequence' in Hoare's Logic (cf. Apt [1]). In our setting we shall use the following version: $(\mathrm{ROE}_\mathsf{M})$:

$$(\mathrm{ROE}_\mathsf{M}) \qquad \frac{\mathbb{BM}_3(\mathbb{P}) \models \varphi = \psi}{\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) \vdash \varphi :\to x = \psi :\to x}$$

We write $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_\mathsf{M} \vdash x = y$, or shortly $\vdash x = y$, if $x = y$ follows from the axioms of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$, the axioms and rules for $\mathbb{BM}_3(\mathbb{P})$, and the rule of equivalence ROE_M. We end this section with some useful derivabilities.

Lemma 3.1 *The following identities can be derived in* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_\mathsf{M}$*:*

1. $\vdash \varphi :\to \delta + x = \varphi \vee \mathsf{T} :\to x$,
2. $\vdash \varphi :\to x + y = \varphi :\to x + \varphi \vee \mathsf{T} :\to y$,
3. $\vdash \mu \between x = \mu$,
4. $\vdash \partial_H(\mu) = \mu$,
5. $\vdash t_I(\mu) = \mu$.

Proof: As for 1: $\varphi :\to \delta + x = \varphi :\to (\mathsf{F} :\to x) + \mathsf{T} :\to x = ((\varphi \between \mathsf{F}) \vee \mathsf{T}) :\to x$. By Lemma 2.1.1, $\models (\varphi \between \mathsf{F}) \vee \mathsf{T} = \varphi \vee \mathsf{T}$ so ROE_M can be applied.

As for 2: Use $\varphi :\to x = \varphi :\to (x + \delta) = \varphi :\to x + \varphi :\to \delta$, and apply 1 on $\varphi :\to \delta + y$.

As for 3–5: Just replace μ by $\mathsf{M} :\to x$ and apply (GC5), (DGC), and (TGC), respectively. □

4 Operational semantics In this section we define an operational semantics for $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$. This semantics defines the contents of a process term in terms of the atomic actions that can be executed (if any) or its interpretation as meaningless. For example,

$$p :\to a \text{ with } p \in \mathbb{P},\ a \in A_t$$

can either resemble μ, the action a, or δ, depending on the interpretation of p.

Let w range over the *valuations* (interpretations) $\mathcal{W}$ of $\mathbb{P}$ in $\mathbb{T}_3^{\mathsf{M}}$. We extend w to $\mathbb{BM}_3(\mathbb{P})$ in the ususal way:

$$\begin{aligned} w(c) &\stackrel{\text{def}}{=} c \text{ for } c \in \{\mathsf{M}, \mathsf{T}, \mathsf{F}\}, \\ w(\neg\varphi) &\stackrel{\text{def}}{=} \neg(w(\varphi)), \\ w(\varphi \diamond \psi) &\stackrel{\text{def}}{=} w(\varphi) \diamond w(\psi) \text{ for } \diamond \in \{\wedge, \vee, \overset{\circ}{\wedge}, \overset{\circ}{\vee}, \underset{\circ}{\wedge}, \underset{\circ}{\vee}\}. \end{aligned}$$

With the system defined in Section 2, it follows that if $\models w(\varphi) = w(\psi)$ for all $w \in \mathcal{W}$, then $\models \varphi = \psi$. For each $w \in \mathcal{W}$ and $\varphi \in \mathbb{BM}_3(\mathbb{P})$ we define inductively in Table 3 the unary predicate *meaningless*, notation $\mu(w, \_)$ on $\mathcal{P}$, the set of process terms over $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$. This predicate defines which process terms represent the meaningless process μ under a certain valuation w.

Table 3: Rules for μ in *panth*-format

μ	$\mu(w, \mu)$
$:\!\rightarrow$	$\mu(w, \varphi :\!\rightarrow x)$ if $w(\varphi) = \mathsf{M}$
	$\dfrac{\mu(w, x)}{\mu(w, \varphi :\!\rightarrow x)}$ if $w(\varphi) = \mathsf{T}$
$+, \cdot, \|, \underline{\|}, \|, \partial_H, t_I$	$\dfrac{\mu(w, x)}{\mu(w, x + y)}$
	$\mu(w, y + x)$
	$\mu(w, x \cdot y)$
	$\mu(w, x \| y)$
	$\mu(w, y \| x)$
	$\mu(w, x \underline{\|} y)$
	$\mu(w, x \mid y)$
	$\mu(w, y \mid x)$
	$\mu(w, \partial_H(x))$
	$\mu(w, t_I(x))$

The axioms and rules for $\mu(w, \_)$ given in Table 3 are extended by axioms and rules given in Table 4 which define for $w \in \mathcal{W}$ and $a \in A_t$ transitions

$$\_ \xrightarrow{w,a} \_ \subseteq \mathcal{P} \times \mathcal{P},$$

and unary 'tick-predicates' or 'termination transitions'

$$\_ \xrightarrow{w,a} \surd \subseteq \mathcal{P}.$$

Transitions characterize under which interpretations a process term defines the possibility to execute an atomic action and what remains to be executed (if anything, otherwise $\surd$ symbolizes successful termination).

Table 4: Transition rules in *panth*-format

$a \in A_t$	$a \xrightarrow{w,a} \surd$	
$\cdot$, $\between$	$\dfrac{x \xrightarrow{w,a} \surd}{x \cdot y \xrightarrow{w,a} y \qquad x \between y \xrightarrow{w,a} y}$	$\dfrac{x \xrightarrow{w,a} x'}{x \cdot y \xrightarrow{w,a} x'y \qquad x \between y \xrightarrow{w,a} x' \parallel y}$
$+$, $\parallel$	$\dfrac{x \xrightarrow{w,a} \surd \quad \neg\mu(w,y)}{x + y \xrightarrow{w,a} \surd \qquad y + x \xrightarrow{w,a} \surd \qquad x \parallel y \xrightarrow{w,a} y \qquad y \parallel x \xrightarrow{w,a} y}$	$\dfrac{x \xrightarrow{w,a} x' \quad \neg\mu(w,y)}{x + y \xrightarrow{w,a} x' \qquad y + x \xrightarrow{w,a} x' \qquad x \parallel y \xrightarrow{w,a} x' \parallel y \qquad y \parallel x \xrightarrow{w,a} y \parallel x'}$
$\mid$, $\parallel$ communication	$\dfrac{x \xrightarrow{w,a} \surd \quad y \xrightarrow{w,b} \surd}{x \mid y \xrightarrow{w,c} \surd \qquad x \parallel y \xrightarrow{w,c} \surd} \; a \mid b = c$	$\dfrac{x \xrightarrow{w,a} \surd \quad y \xrightarrow{w,b} y'}{x \mid y \xrightarrow{w,c} y' \qquad x \parallel y \xrightarrow{w,c} y'} \; a \mid b = c$
	$\dfrac{x \xrightarrow{w,a} x' \quad y \xrightarrow{w,b} \surd}{x \mid y \xrightarrow{w,c} x' \qquad x \parallel y \xrightarrow{w,c} x'} \; a \mid b = c$	$\dfrac{x \xrightarrow{w,a} x' \quad y \xrightarrow{w,b} y'}{x \mid y \xrightarrow{w,c} x' \parallel y' \qquad x \parallel y \xrightarrow{w,c} x' \parallel y'} \; a \mid b = c$
∂_H	$\dfrac{x \xrightarrow{w,a} \surd}{\partial_H(x) \xrightarrow{w,a} \surd}$ if $a \notin H$	$\dfrac{x \xrightarrow{w,a} x'}{\partial_H(x) \xrightarrow{w,a} \partial_H(x')}$ if $a \notin H$
t_I	$\dfrac{x \xrightarrow{w,a} \surd}{t_I(x) \xrightarrow{w,a} \surd}$ if $a \notin I$	$\dfrac{x \xrightarrow{w,a} x'}{t_I(x) \xrightarrow{w,a} t_I(x')}$ if $a \notin I$
	$\dfrac{x \xrightarrow{w,a} \surd}{t_I(x) \xrightarrow{w,t} \surd}$ if $a \in I$	$\dfrac{x \xrightarrow{w,a} x'}{t_I(x) \xrightarrow{w,t} t_I(x')}$ if $a \in I$
$:\!\rightarrow$	$\dfrac{x \xrightarrow{w,a} \surd}{\varphi :\!\rightarrow x \xrightarrow{w,a} \surd}$ if $w(\varphi) = \mathsf{T}$	$\dfrac{x \xrightarrow{w,a} x'}{\varphi :\!\rightarrow x \xrightarrow{w,a} x'}$ if $w(\varphi) = \mathsf{T}$

The following result clarifies the relation between (termination) transitions and the meaningless predicate and follows easily by induction on the structure of the process term involved.

Lemma 4.1 *If* $x \xrightarrow{w,a} x'$ *or* $x \xrightarrow{w,a} \surd$ *for some* w *and* a, *then* $\neg\mu(w, x)$.

Note that the converse implication does not hold (take $x = \delta$).

The axioms and rules in Tables 3 and 4 yield a structured operational semantics (SOS) with negative premises in the style of Groote [16]. Moreover, this SOS satisfies the so-called panth-format, defined by Verhoef [23], which in this case defines the following notion of *bisimulation equivalence*.[4]

Definition 4.2 Let $\mathcal{P}$ be the set of process terms over $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ and $B \subseteq \mathcal{P} \times \mathcal{P}$. Then B is a *bisimulation* if for all P, Q with PBQ the following conditions hold for all $w \in \mathcal{W}$ and $a \in A_t$:

1. $\forall P'\ (P \xrightarrow{w,a} P' \implies \exists Q'(Q \xrightarrow{w,a} Q' \wedge P'BQ'))$,
2. $\forall Q'\ (Q \xrightarrow{w,a} Q' \implies \exists P'(P \xrightarrow{w,a} P' \wedge P'BQ'))$,
3. $P \xrightarrow{w,a} \surd \iff Q \xrightarrow{w,a} \surd$,
4. $\mu(w, P) \iff \mu(w, Q)$.

Two process terms P and Q are *bisimilar*, notation

$$P \leftrightarroweq Q,$$

if there exists a bisimulation B containing the pair (P, Q).

Furthermore, from Fokkink and van Glabbeek [15] and [23] it follows that its transitions and meaningless instances are uniquely determined. This can be established with help of the following simple *stratification* S:

$$S(\mu(w, x)) = 0,$$
$$S(x \xrightarrow{w,a} x') = S(x \xrightarrow{w,a} \surd) = 1,$$

As a consequence, we can apply the main result of [23]: bisimilarity is a *congruence* for all operations involved.

Lemma 4.3 *The system* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_{\mathsf{M}}$ *is sound with respect to bisimulation: for all process terms* P, Q *over* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$,

$$\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_{\mathsf{M}} \vdash P = Q \implies P \leftrightarroweq Q.$$

Proof: Because bisimulation equivalence is a congruence, it is left to show that each axiom in Table 1 and the rule $\mathrm{ROE}_{\mathsf{M}}$ yield bisimilar instances. For all axioms except (A5), (CM1), (CMC), (T4), and (D4), this can be checked by taking

$$B \stackrel{\text{def}}{=} \Delta \cup \{(i_l, i_r)\},$$

where Δ is the diagonal in $\mathcal{P} \times \mathcal{P}$, and the pair (i_l, i_r) represents the instance to be checked. As an example, consider an instance of (GCM7), say

$$\varphi :\rightarrow aP \mid \psi :\rightarrow b = \varphi \wedge \psi :\rightarrow (a \mid b)P.$$

Now $\mu(w, \_)$ either holds for both $\varphi :\to aP \mid \psi :\to b$ and $\varphi \wedge \psi :\to (a \mid b)P$ or not. This follows easily from the rules in Table 3. In the first case we are done; in the second case, either $\gamma(a, b) = \delta$, and both process terms cannot perform any transition, or both can perform

$$\xrightarrow{w,a|b} P$$

by the rules in Table 4. Thus B is a bisimulation and

$$(\varphi :\to aP \mid \psi :\to b,\ \varphi \wedge \psi :\to (a \mid b)P) \in B.$$

For instances of (A5), (CM1), (CMC), (T4), and (D4) a witnessing bisimulation is slightly more complex. As for (A5), assume for some process term P that

$$P \xrightarrow{w,a} P'.$$

Then $(PQ)R \xrightarrow{w,a} (P'Q)R$ and $P(QR) \xrightarrow{w,a} P'(QR)$. A sufficient witnessing bisimulation is in this case the diagonal extended with *all* appropriate associative variants:

$$B \stackrel{\text{def}}{=} \Delta \cup \{((xy)z, x(yz)) \mid x, y, z \in \mathcal{P}\}.$$

It is not hard to check that B is a bisimulation that contains each instance of (A5).

As for axiom (CM1) consider an arbitrary instance

$$P \parallel Q = (P \mathbin{\underline{\parallel}} Q + Q \mathbin{\underline{\parallel}} P) + P \mid Q.$$

Let

$$B \stackrel{\text{def}}{=} \Delta \cup \{(P \parallel Q, (P \mathbin{\underline{\parallel}} Q + Q \mathbin{\underline{\parallel}} P) + P \mid Q)\} \cup \{(x \parallel y, y \parallel x) \mid x, y \in \mathcal{P}\}.$$

Then B witnesses the arbitrary instance. In particular, if $Q \xrightarrow{w,a} Q'$ then $P \parallel Q \xrightarrow{w,a} P \parallel Q'$ and $(P \mathbin{\underline{\parallel}} Q + Q \mathbin{\underline{\parallel}} P) + P \mid Q \xrightarrow{w,a} Q' \parallel P$. Again, it is easily checked that B is a bisimulation. Furthermore, in this case B witnesses each instance of (CMC).

As for instances of (T4) with fixed $I \subseteq$, let

$$B \stackrel{\text{def}}{=} \Delta \cup \{t_I(xy), t_i(x)t_I(y) \mid x, y \in \mathcal{P}\}.$$

It quickly follows that B is a bisimulation that contains each instance of (T4). For instances of (D4) a similar argument applies.

Finally, $\varphi :\to P \leftrightarrows \psi :\to P$ if $\models \varphi = \psi$ by definition of the meaningless predicate, which proves the soundness of $\mathrm{ROE}_{\mathsf{M}}$. □

5 Completeness In this section we prove completeness of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_{\mathsf{M}}$, that is, if for process terms P and Q it holds that $P \leftrightarrows Q$, then $P = Q$ can be derived. We prove this by distinguishing a term representation of bisimilar processes that implies derivability in a direct way.

Definition 5.1 A process term P over $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ is a *basic term* if

$$P \equiv \sum_{i \in I} \varphi_i :\to Q_i$$

where $\equiv$ is used for syntactic equivalence, I is a finite, nonempty index set, $\varphi_i \in \mathbb{BM}_3(\mathbb{P})$, and $Q_i \in \{\delta, a, aR \mid a \in A_t,\ R \text{ a basic term}\}$.

Lemma 5.2 *All process terms over* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ *can be proved equal to a basic term.*

Proof: Standard induction on term complexity. □

For $a \in A_t$ and $\varphi \in \mathbb{BM}_3(\mathbb{P})$, the *height* of a basic term is defined by

$$\begin{aligned} h(\delta) &= 0, \\ h(a) &= 1, \\ h(\varphi :\to x) &= h(x), \\ h(x + y) &= max(h(x), h(y)), \\ h(a \cdot x) &= 1 + h(x). \end{aligned}$$

Lemma 5.3 *If P is a basic term, there is a basic term P' with $\vdash P = P'$, $h(P') \leq h(P)$, and P' has either the form*

$$\psi :\to \delta, \tag{1}$$

or the form

$$\sum_{i \in I} \psi_i :\to Q_i \tag{2}$$

with (*i*) *for all $i, j \in I$, $Q_i \not\equiv \delta$, and $Q_i, Q_j \in A_t \Rightarrow Q_i \not\equiv Q_j$ if $i \neq j$,*

(*ii*) *if $\exists i \in I, w \in \mathcal{W}$ such that $w(\psi_i) = \mathsf{M}$, then $\forall j \in I$, $w(\psi_j) = \mathsf{M}$,*

(*iii*) *for each $i \in I$ there is $w \in \mathcal{W}$ such that $w(\psi_i) = \mathsf{T}$.*

Proof: Let

$$P \equiv \textstyle\sum_{i=1}^{n} \varphi_i :\to Q_i$$

for some $n \geq 1$. By Lemma 3.1.1 and axiom GCL4 we may assume that for all i either $Q_i \not\equiv \delta$ or $Q_i \equiv \delta$. In the latter case this yields with axiom GC1, form 1. In the first case we may assume that each single action occurs at most once (by (GC1)). This proves property (*i*) of form 2. Let

$$\overline{\varphi} \equiv (\varphi_1 \vee \mathsf{T}) \mathbin{\diamond\!\!\wedge} \cdots \mathbin{\diamond\!\!\wedge} (\varphi_n \vee \mathsf{T}).$$

(Recall that $\mathbin{\diamond\!\!\wedge}$ is associative.) Observe that for each $w \in \mathcal{W}$, $w(\overline{\varphi}) \in \{\mathsf{T}, \mathsf{M}\}$. Let furthermore,

$$\begin{aligned} \psi_i &\equiv \overline{\varphi} \mathbin{\diamond\!\!\wedge} \varphi_i, \\ P'' &\equiv \textstyle\sum_{i=1}^{n} \psi_i :\to Q_i. \end{aligned}$$

Note that if $w(\psi_i) = \mathsf{M}$ for some i and w, then $w(\psi_j) = \mathsf{M}$ for all $j \in \{1, \ldots, n\}$. We show that

$$\vdash P = P'' \tag{3}$$

by induction on n.

$\underline{n=1:}$ This follows immediately from Lemma 2.1.2.

$\underline{n=k+1:}$ Let $\overline{\overline{\varphi}} \equiv (\varphi_2 \vee \mathsf{T}) \between \cdots \between (\varphi_n \vee \mathsf{T})$ and $\overline{\overline{\varphi_i}} \equiv \overline{\overline{\varphi}} \between \varphi_i$ for $i = 2, \ldots n$. By induction we have that $\vdash P = \varphi_1 :\to Q_1 + \sum_{i=2}^{n} \overline{\overline{\varphi_i}} :\to Q_i$.

With k applications of Lemma 3.1.2 and (GCL4), we obtain

$$\vdash P = \varphi_1 :\to Q_1 + \sum_{i=2}^{n} \psi_i :\to Q_i.$$

Doing the same once more yields

$$\vdash P = (\psi_2 \vee \mathsf{T}) \between \varphi_1 :\to Q_1 + \textstyle\sum_{i=2}^{n} \psi_i :\to Q_i.$$

Now it follows easily that $\models \psi_1 = (\psi_2 \vee \mathsf{T}) \between \varphi_1$ (recall that $\psi_2 \equiv \overline{\varphi} \between \varphi_2$ and $w(\psi) \in \{\mathsf{T}, \mathsf{M}\}$). This finishes the proof of (3) and proves properties (i) and (ii) of form 2 for P''.

Next we consider all summands from P'' for which no valuation makes the condition true. For each such summand $\psi_i :\to Q_i$ it holds that $\models \psi_i = \psi_i \between \mathsf{F}$, and thus

$$\begin{aligned} \vdash \psi_i :\to Q_i &= \psi_i \between \mathsf{F} :\to Q_i \\ &= \psi_i :\to (\mathsf{F} :\to Q_i) \\ &= \psi_i :\to \delta. \end{aligned}$$

In case all summands can be proved equal to $\psi_j :\to \delta$ in this way, we are finished using (3):

$$\vdash P = \psi_1 \vee \cdots \vee \psi_n :\to \delta.$$

In the other case, $w(\psi_i) = \mathsf{T}$ for certain w, i. If $\vdash \psi_j :\to Q_j = \psi_j :\to \delta$ for some j, then by Lemma 3.1.2, $\vdash \psi_j :\to \delta + \psi_i :\to Q_i = (\psi_j \vee \mathsf{T}) \between \psi_i :\to Q_i$. Now $\models \psi_i = (\psi_j \vee \mathsf{T}) \between \psi_i$ as was already used in the proof of (3). Hence we obtain

$$\vdash P = \textstyle\sum_{i=1}^{k} \psi_i :\to Q_i$$

with $k \leq n$ (and possibly some rearrangement of indices), and for each $i \in \{1, \ldots, k\}$ there is a valuation w with $w(\psi_i) = \mathsf{T}$. This proves property (iii) of form 2 and preserves properties (i) and (ii). □

Lemma 5.4 *Let P_1, P_2 be basic terms. Then*

$$P_1 \leftrightarrows P_2 \implies \vdash P_1 = P_2.$$

Proof: By the previous Lemma 5.3, we may assume that both P_1 and P_2 satisfy either form 1 or form 2 given there. We proceed by induction on $h = max(h(P_1), h(P_2))$.

Let $h = 0$. Then $P_n \equiv \varphi_n :\to \delta$ for $n = 1, 2$. So $\vdash \varphi_1 :\to \delta \leftrightarrows \varphi_2 :\to \delta$ and $w(\varphi_1) = \mathsf{M} \iff w(\varphi_2) = \mathsf{M}$. This implies that $\mathbb{BM}_3(\mathbb{P}) \models \varphi_1 \between \mathsf{F} = \varphi_2 \between \mathsf{F}$. Now $\varphi_n :\to \delta = \varphi_n :\to (\mathsf{F} :\to \delta) = \varphi_n \between \mathsf{F} :\to \delta$. Consequently, $\vdash P_1 = P_2$.

Let $h > 0$ and $P_n \equiv \sum_{i \in I_n} \psi_{n,i} :\to Q_{n,i}$ for $n = 1, 2$. By the previous Lemma 5.3, we may assume that P_n satisfies form 2 given there. Furthermore, we may assume that for all $i \in I_n$, $Q_{n,i} \not\leftrightarrows Q_{n,j}$ for $j \in I_n \setminus \{i\}$. For the case $Q_{n,i} \equiv aR_{n,i}$ and $Q_{n,j} \equiv aR_{n,j}$ this follows by induction: $R_{n,i} \leftrightarrows R_{n,j}$ implies $\vdash R_{n,i} = R_{n,j}$, so $\vdash aR_{n,i} = aR_{n,j}$, and thus (GC1) could have been applied.

Now each summand of P_n can be proved equal to one in P_{3-n} and by Lemma 5.3, each summand yields a transition for a certain $w \in \mathcal{W}$.

1. Assume that $P_n \xrightarrow{w,a} \surd$ for some w, a. Thus $w(\psi_{n,i}) = \mathsf{T}$ for some unique $i \in I_n$. By $P_1 \leftrightarrows P_2$, there is a unique $j \in I_{3-n}$ for which $P_{3-n} \xrightarrow{w,a} \surd$ and $\models \psi_{n,i} = \psi_{3-n,j}$ (the latter derivability follows from the representation as defined in Lemma 5.3 and the nonbisimilarity of different summands). Thus

$$\vdash \psi_{n,i} :\to a = \psi_{3-n,j} :\to a.$$

2. Assume that $P_n \xrightarrow{w,a} R_{n,i}$ for some w, a and unique $i \in I_n$. Thus $w(\psi_{n,i}) = \mathsf{T}$. By $P_1 \leftrightarrows P_2$, there must be some unique $j \in I_{3-n}$ for which $P_{3-n} \xrightarrow{w,a} R_{3-n,j}$ and $R_{n,i} \leftrightarrows R_{3-n,j}$, and for which $\models \psi_{n,i} = \psi_{3-n,j}$ follows from Lemma 5.3. By induction we find $\vdash R_{n,i} = R_{3-n,j}$, and therefore $\vdash aR_{n,i} = aR_{3-n,j}$ and hence

$$\vdash \psi_{n,i} :\to aR_{n,i} = \psi_{3-n,j} :\to aR_{3-n,j}.$$

By symmetry, it follows that each summand of P_n is provably equal to one P_{n-3}. Consequently, $\vdash P_1 = P_2$. □

With Lemmas 5.2, 5.3, 5.4, and soundness (Lemma 4.3) we obtain the following theorem.

Theorem 5.5 *The system* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_\mathsf{M}$ *is complete with respect to bisimulation equivalence: for all process terms* P, Q *over* $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$,

$$\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_\mathsf{M} \vdash P = Q \iff P \leftrightarrows Q.$$

6 Parameterized actions and nonclassical values When dealing with actions $a(x)$ parameterized by x over some data type, it makes sense to consider the case in which data can also take value M. If so, one faces the question how to interpret $a(\mathsf{M})$. Given the preceding interpretation of conditions, a natural choice is to take

$$a(\mathsf{M}) = \mu.$$

(So $a(x)$ is an atomic action in case $x \neq \mathsf{M}$.) We end this section with two examples on the use of M and μ.

Example 6.1 As an example of the use of process algebra with the proposed three-valued propositional logic $\mathbb{BM}_3(\mathbb{P})$, we consider a process that manipulates bounded stacks over data sort

$$Data \stackrel{\text{def}}{=} \{d, e, f, g, h\}.$$

Let $Data^{0\text{-}5} \stackrel{\text{def}}{=} \{\rho \in Data^* \mid length(\rho) \leq 5\}$, where we assume that ϵ is the empty string in $Data^*$, and *length* is a given function that yields the length of a string in $Data^*$. Let furthermore

$$\begin{aligned} Data_{\mathsf{M}} &\stackrel{\text{def}}{=} Data \cup \{\mathsf{M}\}, \\ Data_{\mathsf{M}}^{0\text{-}5} &\stackrel{\text{def}}{=} Data^{0\text{-}5} \cup \{\mathsf{M}\}, \end{aligned}$$

and let the following functions be defined:

$$\begin{aligned} app &: Data \times Data_{\mathsf{M}}^{0\text{-}5} \to Data_{\mathsf{M}}^{0\text{-}5}, \\ &\quad app(d, \rho) = \mathsf{M} \text{ if } length(\rho) = 5 \text{ or if } \rho = \mathsf{M}, \end{aligned}$$

$$\begin{aligned} head &: Data_{\mathsf{M}}^{0\text{-}5} \to Data, \\ &\quad head(\rho) = \begin{cases} \mathsf{M} & \text{if } \rho \in \{\epsilon, \mathsf{M}\} \\ d & \text{if } \rho = app(d, \rho'), \rho' \in Data^{0\text{-}5}, \end{cases} \end{aligned}$$

$$\begin{aligned} tail &: Data_{\mathsf{M}}^{0\text{-}5} \to Data_{\mathsf{M}}^{0\text{-}5}, \\ &\quad tail(\rho) = \begin{cases} \mathsf{M} & \text{if } \rho \in \{\epsilon, \mathsf{M}\} \\ \rho' & \text{if } \rho = app(d, \rho'), \rho' \in Data^{0\text{-}5}, \end{cases} \end{aligned}$$

$$\begin{aligned} empty &: Data_{\mathsf{M}}^{0\text{-}5} \to \mathbb{T}_3^{\mathsf{M}}, \\ &\quad empty(\rho) = \begin{cases} \mathsf{T} & \text{if } \rho = \epsilon \\ \mathsf{M} & \text{if } \rho = \mathsf{M} \\ \mathsf{F} & \text{otherwise,} \end{cases} \end{aligned}$$

$$\begin{aligned} full &: Data_{\mathsf{M}}^{0\text{-}5} \to \mathbb{T}_3^{\mathsf{M}}, \\ &\quad full(\rho) = \begin{cases} \mathsf{T} & \text{if } length(\rho) = 5 \\ \mathsf{M} & \text{if } \rho = \mathsf{M} \\ \mathsf{F} & \text{otherwise.} \end{cases} \end{aligned}$$

We think these functions characterize straightforwardly the role of *meaningless*:

$$head(\epsilon) = tail(\epsilon) = \mathsf{M},$$

and all are monotonic in M.

The following process term $Q(x)$ describes manipulation with stack object $x \in Data^{0\text{-}5}$ including all "special cases" that evolve from $Data^{0\text{-}5}_{\mathsf{M}}$.

$$\begin{aligned} Q(x) \;=\;& \neg full(x) :\to \textstyle\sum_{d\in Data} r_1(d)\cdot Q(app(d,x)) \\ &+ \neg empty(x) :\to (s_2(head(x)) + s_3(head(x)))\cdot Q(tail(x)) \\ &+ \neg empty(x) \mathbin{\diamond\!\!\wedge} empty(tail(x)) :\to s_4(head(x))\cdot Q(\epsilon) \\ &+ empty(x) :\to s_5(is\_empty)\cdot Q(\epsilon). \end{aligned}$$

Here actions $r_j(\_)$ and $s_j(\_)$ model receive and send actions, respectively, of data along channel j. So, $Q(x)$ can be 'pushed' in case it is not full and be 'popped' in case it is not empty in three ways: either it can send its head value along channel 2 or 3, and evolve into $Q(tail(x))$, or in case of the one-element stack, it can send this value along channel 4. The empty stack value ϵ can be observed via action $s_5(is\_empty)$. We consider $Q(x)$ in parallel with a *reset* process R defined by

$$R \;=\; r_6(set\_empty)\cdot\left(\textstyle\sum_{d\in Data} r_3(d)\right)^* r_5(is\_empty)\big).$$

Here $\gamma(r_j(x), c_j(x)) = c_j(x)$ for all values of $x \in Data_{\mathsf{M}} \cup \{is\_empty\}$. The idea is that R can be triggered to reset $Q(x)$ to $Q(\epsilon)$ via action $r_6(set\_empty)$. Let $H = \{r_j(x), s_j(x) \mid j = 3, 4,\ x \in Data_{\mathsf{M}} \cup \{set\_empty, is\_empty\}\}$. Then

$$\partial_H(Q(x) \parallel R)$$

models this reset. We still have $\partial_H(Q(\mathsf{M}) \parallel R) = \mu$.

Example 6.2 Consider the data type $\omega = \{0, S(0), S(S(0)), \ldots\}$, with equality $\equiv$: $\omega \times \omega \to \mathbb{T}_3^{\mathsf{M}}$ (binary infix) defined by

$$\begin{aligned} 0 \equiv 0 &= \mathsf{T}, \\ 0 \equiv S(x) &= \mathsf{F}, \\ S(x) \equiv 0 &= \mathsf{F}, \\ S(x) \equiv S(y) &= x \equiv y. \end{aligned}$$

Let the predecessor function $P : \omega \to \omega \cup \{\mathsf{M}\}$ be defined by $P(0) = \mathsf{M}$, and $P(S(x)) = x$. We extend the domains of S, $\equiv$, and P with value M by defining

$$S(\mathsf{M}) = x \equiv \mathsf{M} = \mathsf{M} \equiv x = P(\mathsf{M}) = \mathsf{M}.$$

Now consider the following counterlike process:

$$\begin{aligned} C(x) \;=\;& r(up)\cdot C(S(x)) + r(down)\cdot C(P(x)) + r(set\_zero)\cdot C(0) \\ &+ x \equiv 0 :\to r(is\_zero)\cdot C(x). \end{aligned}$$

With the action $r(up)$, a command to increase can be received, and a command to decrease is modeled by the action $r(down)$. The action $r(set\_zero)$ models a reset of

the counter to $C(0)$, and action $r(\mathit{is\_zero})$ indicates that the counter value equals 0. It follows that

$$\begin{aligned} C(\mathsf{M}) &= \mu, \\ C(0) &= r(\mathit{up}) \cdot C(S(0)) + r(\mathit{down}) \cdot C(\mathsf{M}) + r(\mathit{set\_zero}) \cdot C(0) \\ &\quad + r(\mathit{is\_zero}) \cdot C(0), \\ C(S^{k+1}(0)) &= r(\mathit{up}) \cdot C(S^{k+2}(0)) + r(\mathit{down}) \cdot C(S^k(0)) + r(\mathit{set\_zero}) \cdot C(0). \end{aligned}$$

So, if in case of $C(0)$ the action $r(\mathit{down})$ is performed, the counter evolves into μ.

7 Conclusion The extension of process algebra to a three-valued setting with *meaningless* entails the introduction of a new process constant μ and the defining axiom $\mathsf{M} :\to x = \mu$. As a guard, M is totally persistent: whenever a condition to be evaluated in a process term equals M, the process term equals μ. In a parallel setting, this is much stronger than occurrence of F in a condition, for example,

$$(\mathsf{F} :\to a) \parallel bc = bc\delta, \text{ whereas } (\mathsf{M} :\to a) \parallel bc = \mu.$$

In our set-up, $\mathsf{F} :\to \mu = \delta$ holds. This clearly illustrates that the conditional guard construct reflects a certain order: first the condition is evaluated, then its right argument is considered. So, in this respect the conditional guard construct really guards its process term. This agrees with the intuition as exemplified by the following program fragment:

if T **then** P **else** $\langle$***type-clash***$\rangle$ **fi**,

where we certainly have a clue of its operational behavior when we have one of P.

Note 7.1 Recently we published some more papers on process algebra with nonclassical logics [8, 10, 9]. The most intricate of these starts from a five-valued logic that comprises in addition to T and F not only the value M, but also values D for *divergence*, and C for *choice* or *undetermined*. Typically, $\mathsf{D} :\to x = \delta$ and $x \lhd \mathsf{C} \rhd y = x + y$. (So, with conditional composition, C gives a counterpart of the $+$ operation in process algebra.)

We think that the present paper describes an approach that in its own right deserves publication, and we hope that it provides an elegant introduction to the general subject of process algebra with nonclassical logics, or perhaps—quoting a referee—"error handling in parallel and distributed software systems."

Acknowledgments We thank a referee for suggesting some improvements.

NOTES

1. The *chaos* process χ, which stems from CSP [17], seems to be characterized by exactly the same laws as given for μ. But χ characterizes the effect of infinite internal activity and its laws capture an intuition that is not useful in our set-up. Modeling internal activity explicitly would distinguish μ and χ. Therefore we introduce this new constant.

2. Of course, $\delta = \mu$ implies inconsistency of our theory: $x = x + \delta = x + \mu = \mu$.

3. Note that in a two-valued setting, $\owedge$ and $\wedge$ coincide.

4. A general reference to bisimulation equivalence is Park [22]. Its role in process algebra is overviewed in [3].

REFERENCES

[1] Apt, K. R., "Ten years of Hoare's logic, a survey, part I," *ACM Transactions on Programming Languages and Systems*, vol. 3 (1981), pp. 431–83.

[2] Baeten, J. C. M., and J. A. Bergstra, "Process algebra with signals and conditions," pp. 273–323 in *Programming and Mathematical Method,Proceedings Summer School Marktoberdorf*, 1990 NATO ASI Series F, edited by M. Broy, Springer-Verlag, 1992.

[3] Baeten, J. C. M., and W. P. Weijland, *Process algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, Cambridge, 1990.

[4] Bergstra, J. A., I. Bethke, and A. Ponse, "Process algebra with iteration and nesting," *Computer Journal*, vol. 37 (1994), pp. 243–58.

[5] Bergstra, J. A., I. Bethke, and P. H. Rodenburg, "A propositional logic with 4 values: true, false, divergent and meaningless," *Journal of Applied Non-Classical Logics*, vol. 5 (1995), pp. 199–217.

[6] Bergstra, J. A., and J. W. Klop, "The algebra of recursively defined processes and the algebra of regular processes," pp. 1–25 in *Algebra of Communicating Processes, Utrecht 1994*, edited by A. Ponse, C. Verhoef, and S. F. M. van Vlijmen, Springer-Verlag, 1995.

[7] Bergstra, J. A., and J. W. Klop, "Process algebra for synchronous communication," *Information and Control*, vol. 60 (1984), pp. 109–37.

[8] Bergstra, J. A., and A. Ponse, "Kleene's three-valued logic and process algebra," *Information Processing Letters*, vol. 67 (1998), pp. 95–103.

[9] Bergstra, J. A., and A. Ponse, "Process algebra with five-valued conditions," pp. 128–43, in *Combinatorics, Complexity, and Logic, Proceedings of DMTCS'99 and CATS'99*, vol. 21, nr. 3 of Australian Computer Science Communications, edited by C. S. Calude and M. J. Dinneen, Springer-Verlag, Singapore, 1999.

[10] Bergstra, J.A., and A. Ponse, "Process algebra with four-valued logic," *Journal of Applied Non-Classical Logics*, vol. 10 (2000), pp. 27–53.

[11] Bochvar, D. A., "On a 3-valued logical calculus and its application to the analysis of contradictions (in Russian)," *Matématičeskij sbornik*, vol. 4 (1939), pp. 287–308.

[12] Brookes, S. D., C. A. R. Hoare, and A. W. Roscoe, "A theory of communicating sequential processes," *Journal of the ACM*, vol. 31 (1984), pp. 560–99.

[13] Dijkstra, E. W., *A Discipline of Programming*, Prentice Hall International, Englewood Cliffs, 1976.

[14] Fokkink, W. J., "Axiomatizations for the perpetual loop in process algebra," pp. 571–81 in *Proceedings of the 24th Colloquium on Automata, Languages and Programming—ICALP'97*, Lecture Notes in Computer Science 1256, edited by P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, Springer-Verlag, 1997.

[15] Fokkink, W. J., and R. J. van Glabbeek, "Ntyft/ntyxt rules reduce to ntree rules," *Information and Computation*, vol. 126 (1996), pp. 1–10.

[16] Groote, J. F., "Transition system specifications with negative premises," *Theoretical Computer Science*, vol. 118 (1993), pp. 263–99.

[17] Hoare, C. A. R., I. J. Hayes, He Jifeng, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorensen, J. M. Spivey, and B. A. Sufrin, "Laws of programming," *Communications of the ACM*, vol. 30 (1987) pp. 672–86.

[18] Kleene, S. C., "Representation of events in nerve nets and finite automata," *Automata Studies*, pp. 3–41, edited by C. Shannon and J. McCarthy, Princeton University Press, Princeton, 1956.

[19] Konikowska, B., "McCarthy algebras: a model of McCarthy's logical calculus," *Fundamenta Informaticae*, vol. 26 (1996), pp. 167–203.

[20] McCarthy, J., "A basis for a mathematical theory of computation," pp. 33–70 in *Computer Programming and Formal Systems*, edited by P. Braffort and D. Hirshberg, North-Holland, Amsterdam, 1963.

[21] Milner, R., *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, 1989.

[22] Park, D. M. R., "Concurrency and automata on infinite sequences," pp. 167–83 in *Proceedings of the 5th GI (Gesellschaft für Informatik) Conference*, Karlsruhe, LNCS 104, edited by P. Deussen, Springer-Verlag, 1981.

[23] Verhoef, C., "A congruence theorem for structured operational semantics with predicates and negative premises," *Nordic Journal of Computing*, vol. 2 (1995), pp. 274–302.

Programming Research Group
University of Amsterdam
Kruislaan 403
1098 SJ Amsterdam
THE NETHERLANDS
email: {janb,alban}@science.uva.nl

Department of Philosophy
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
THE NETHERLANDS
email: Jan.Bergstra@phil.uu.nl

CWI
Kruislaan 413
1098 SJ Amsterdam
THE NETHERLANDS
email: Alban.Ponse@cwi.nl