Bochvar-McCarthy Logic and Process Algebra

Jan A. Bergstra[1,2] Alban Ponse[1]

[1] *University of Amsterdam, Programming Research Group*
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

[2] *Utrecht University, Department of Philosophy*
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

E-mail: `janb@wins.uva.nl` - `alban@wins.uva.nl`

Abstract

We propose a combination of Bochvar's strict three-valued logic, McCarthy's sequential three-valued logic, and process algebra via the *guarded command* 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 guarded command and the proposed logic.

Key words & Phrases: process algebra, three-valued logic, guarded command.
1991 CR Categories: F.3, F.4.3, I.1.

1 Introduction

An (immediate) error in an algorithm or program, such as reference to a non-existing instruction, or a type clash, is often easily detectable. In order to model this feature in a concurrent setting, we consider process algebra with guarded command 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 $\neg\phi$ 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.

[1] The *chaos* process χ, which stems from CSP [BHR84], 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.

We recall the *guarded command* $\phi :\rightarrow \_$ from [Dij76] (roughly: *if* ϕ *holds, then* _), which was introduced in process algebra with two-valued logic in [BB90] with the following typical laws where T denotes the value *true*, and F stands for *false*:

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

The constant δ (inaction/deadlock) is well-known in ACP based approaches [BK84a, BK84b, BW90], 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 the *conditional* $x \triangleleft \phi \triangleright y$, introduced in [BB90]. Here x, y are processes, and ϕ is a proposition. This operation satisfies (among others) the following axioms:

$$\begin{aligned} x \triangleleft \mathsf{T} \triangleright y &= x, \\ x \triangleleft \mathsf{F} \triangleright y &= y, \\ x \triangleleft \phi \triangleright y &= y \triangleleft \neg\phi \triangleright x. \end{aligned}$$

The notation $\_ \triangleleft \_ \triangleright \_$ stems from [HHJ+87], and in that paper it is argued that $x \triangleleft \phi \triangleright y$ expresses **if** ϕ **then** x **else** y **fi**. The conditional $\_ \triangleleft \_ \triangleright \_$ can be regarded as more basic than the guarded command, as it does not presuppose the existence of the special constant δ. Of course,

$$x \triangleleft \phi \triangleright y = \phi :\rightarrow x + \neg\phi :\rightarrow y.$$

Finally, process algebra with *two-valued* logic can be characterized by the identity $x \triangleleft \phi \triangleright x = x$.

We fix the relation between M and μ with axiom

$$\mathsf{M} :\rightarrow x \quad = \quad \mu.$$

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

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

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 [Boc39].

Another basic law in [BB90] that we want to accommodate to relates repeated application of the guarded command and conjunction:

$$\phi :\rightarrow (\psi :\rightarrow x) = \phi \wedge \psi :\rightarrow x$$

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

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

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 [McC63], with the asymmetric notation ${}_{\circ}\!\!\wedge$ taken from [BBR95]:

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

Here $\phi \mathbin{{}_{\circ}\!\!\wedge} \psi$ expresses that *first* ϕ is evaluated, and *then* ψ. For recent work on McCarthy's logic see e.g. [Kon96]. A sequential version of the law mentioned above is

$$\phi :\rightarrow (\psi :\rightarrow x) = \phi \mathbin{{}_{\circ}\!\!\wedge} \psi :\rightarrow x.$$

We further adopt this identity as the process algebra axiom that reduces repeated application of the guarded command[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 [BBR95]. In that paper, complete axiomatizations for both Bochvar's and McCarthy's three-valued logic can be found.

Contents of the Paper. 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 the next two sections we define an operational semantics and bisimulation equivalence, and we prove a (relative) completeness result.

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:

$$\begin{aligned} \mathsf{M}, \mathsf{T}, \mathsf{F} :& \quad \rightarrow \mathbb{T}_3^{\mathsf{M}} \\ \neg :& \quad \mathbb{T}_3^{\mathsf{M}} \rightarrow \mathbb{T}_3^{\mathsf{M}} \\ \wedge, \vee, {}_{\circ}\!\!\wedge, {}^{\circ}\!\!\vee, \wedge_{\circ}, \vee^{\circ} :& \quad \mathbb{T}_3^{\mathsf{M}} \times \mathbb{T}_3^{\mathsf{M}} \rightarrow \mathbb{T}_3^{\mathsf{M}} \end{aligned}$$

of which $\neg$, $\wedge$, and ${}_{\circ}\!\!\wedge$ 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

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

The remaining operations are all definable:

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

[3]Note that in a two-valued setting, ${}_{\circ}\!\!\wedge$ and $\wedge$ coincide.

Disjunction:	$x \vee y \stackrel{def}{=} \neg(\neg x \wedge \neg y)$,
Left-sequential disjunction:	$x \mathbin{{}^{\circ}\!\vee} y \stackrel{def}{=} \neg(\neg x \mathbin{{}_{\circ}\!\wedge} \neg y)$,
Right-sequential conjunction:	$x \mathbin{\wedge_{\circ}} y \stackrel{def}{=} y \mathbin{{}_{\circ}\!\wedge} x$,
Right-sequential disjunction:	$x \mathbin{\vee^{\circ}} y \stackrel{def}{=} y \mathbin{{}^{\circ}\!\vee} x$.

We call the resulting logic $\mathbb{BM}_3(\neg, \wedge, \mathbin{{}_{\circ}\!\wedge})$, 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 $\mathbb{BM}_3(\emptyset)$. In order to extend our evaluation system system to propositions $\phi, \psi, ...$ over $\mathbb{P}$, we use substitution on single proposition symbols: let $p, q \in \mathbb{P}$, then

$$\begin{array}{rcl}
[\phi/p]q & \stackrel{def}{=} & q, \\
[\phi/p]p & \stackrel{def}{=} & \phi, \\
[\phi/p]c & \stackrel{def}{=} & c \text{ for } c \in \{\mathsf{M}, \mathsf{T}, \mathsf{F}\}, \\
[\phi/p]\neg\psi & \stackrel{def}{=} & \neg[\phi/p]\psi, \\
[\phi/p](\psi_1 \diamondsuit \psi_2) & \stackrel{def}{=} & [\phi/p]\psi_1 \diamondsuit [\phi/p]\psi_2 \text{ for } \diamondsuit \in \{\wedge, \vee, \mathbin{{}_{\circ}\!\wedge}, \mathbin{{}^{\circ}\!\vee}, \mathbin{\wedge_{\circ}}, \mathbin{\vee^{\circ}}\},
\end{array}$$

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

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

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

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

if $\phi = \psi$ 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 $\models \phi = \psi$. The identity stated in the following lemma is used in the sequel.

Lemma 2.1. $\mathbb{BM}_3(\mathbb{P}) \models (\phi \vee \mathsf{T}) \mathbin{{}_{\circ}\!\wedge} \phi = \phi$.

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, Algebra of Communicating Processes [BK84a, BK84b, BW90]. In the left column of Table 1 we present a slight modification of ACP(A, γ), where A is a set of atomic actions, and γ a binary communication function that is commutative, thus $\gamma(a,b) = \gamma(b,a)$, and associative: $\gamma(a, \gamma(b,c)) = \gamma(\gamma(a,b),c)$. We take γ *total* on $A \times A \to A_\delta$, where $A_\delta = A \cup \{\delta\}$, and we take the communication merge $|$ commutative (CMC) (by which (CM6) and (CM9), the symmetric variants of (CM5) and (CM8) [BW90], become derivable). We shortly comment on the primitives of ACP(A, γ). 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 $\partial_H(C_1 \parallel ... \parallel C_n)$ generally boils down to representing the possible executions with $+$ and $\cdot$, having applied left-merge ($\,\lfloor\!\lfloor\,$), communication merge, and encapsulation (∂_H, by which communication between components C_i can be enforced). After renaming internal activity to some unobservable action (e.g., using pre-abstraction), this may yield a simple and informative specification of external behaviour. 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, [Mil89]). For example, assume component C_1 can receive a value from finite set $Data = \{d_0, d_1, ..., d_N\}$, and then perform its activity, depending on the value d received via action $r(d)$,

$$C_1 = r(d_0) \cdot \mathit{Activity}(d_0) + r(d_1) \cdot \mathit{Activity}(d_1) + ... + r(d_N) \cdot \mathit{Activity}(d_N),$$

or by commutativity and associativity of $+$, shortly,

$$C_1 = \textstyle\sum_{d \in Data} r(d) \cdot \mathit{Activity}(d),$$

where $\cdot$ binds stronger than $+$. 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 $\mathit{Activity}(d_j)$. After introducing the remaining axioms, we continue our explanation.

Table 1: The axiom system $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$, where $a, b \in A_{t\delta}$, $H, I \subseteq A_t$.

(A1)	$x + (y + z)$	$=$	$(x + y) + z$	(M1)	$x + \mu$	$=$	μ
(A2)	$x + y$	$=$	$y + x$	(M2)	$\mu \cdot x$	$=$	μ
(A3)	$x + x$	$=$	x	(M3)	$\mu \mid x$	$=$	μ
(A4)	$(x + y)z$	$=$	$xz + yz$	(GT)	$\mathsf{T} :\rightarrow x$	$=$	x
(A5)	$(xy)z$	$=$	$x(yz)$	(GF)	$\mathsf{F} :\rightarrow x$	$=$	δ
(A6)	$x + \delta$	$=$	x	(GM)	$\mathsf{M} :\rightarrow x$	$=$	μ
(A7)	δx	$=$	δ	(GC1)	$\phi :\rightarrow x + \psi :\rightarrow x$	$=$	$\phi \vee \psi :\rightarrow x$
(CF1)	$a \mid b$	$=$	$\gamma(a, b)$ if $a, b \in A_t$	(GC2)	$\phi :\rightarrow x + \phi :\rightarrow y$	$=$	$\phi :\rightarrow (x + y)$
(CF2)	$a \mid \delta$	$=$	δ	(GC3)	$(\phi :\rightarrow x)y$	$=$	$\phi :\rightarrow xy$
(CM1)	$x \parallel y$	$=$	$(x \lfloor\!\lfloor y + y \lfloor\!\lfloor x) + x \mid y$	(GCL4)	$\phi :\rightarrow (\psi :\rightarrow x)$	$=$	$\phi \mathbin{\wedge\!\!\circ} \psi :\rightarrow x$
(CM2)	$a \lfloor\!\lfloor x$	$=$	ax	(GC5)	$\phi :\rightarrow x \lfloor\!\lfloor y$	$=$	$\phi :\rightarrow (x \lfloor\!\lfloor y)$
(CM3)	$ax \lfloor\!\lfloor y$	$=$	$a(x \parallel y)$	(GCM6)	$\phi :\rightarrow a \mid \psi :\rightarrow b$	$=$	$\phi \wedge \psi :\rightarrow a \mid b$
(CM4)	$(x + y) \lfloor\!\lfloor z$	$=$	$x \lfloor\!\lfloor z + y \lfloor\!\lfloor z$	(GCM7)	$\phi :\rightarrow ax \mid \psi :\rightarrow b$	$=$	$\phi \wedge \psi :\rightarrow (a \mid b)x$
(CMC)	$x \mid y$	$=$	$y \mid x$	(GCM8)	$\phi :\rightarrow ax \mid \psi :\rightarrow by$	$=$	$\phi \wedge \psi :\rightarrow (a \mid b)(x \parallel y)$
(CM5)	$ax \mid b$	$=$	$(a \mid b)x$	(DGC)	$\partial_H(\phi :\rightarrow x)$	$=$	$\phi :\rightarrow \partial_H(x)$
(CM7)	$ax \mid by$	$=$	$(a \mid b)(x \parallel y)$	(TGC)	$t_I(\phi :\rightarrow x)$	$=$	$\phi :\rightarrow t_I(x)$
(CM8)	$(x + y) \mid z$	$=$	$x \mid z + y \mid z$	(T1)	$t_I(a)$	$=$	a if $a \notin I$
(D1)	$\partial_H(a)$	$=$	a if $a \notin H$	(T2)	$t_I(a)$	$=$	t if $a \in I$
(D2)	$\partial_H(a)$	$=$	δ if $a \in H$	(T3)	$t_I(x + y)$	$=$	$t_I(x) + t_I(y)$
(D3)	$\partial_H(x + y)$	$=$	$\partial_H(x) + \partial_H(y)$	(T4)	$t_I(xy)$	$=$	$t_I(x)t_I(y)$
(D4)	$\partial_H(xy)$	$=$	$\partial_H(x)\partial_H(y)$	(Cond)	$x \lhd \phi \rhd y$	$=$	$\phi :\rightarrow x + \neg\phi :\rightarrow y$

In the right column of Table 1 additional axioms on meaningless (μ), pre-abstraction (t_I, i.e., renaming of all actions in I to action t), guarded command ($\phi :\rightarrow$), and conditional are given, where ϕ is taken from $\mathbb{BM}_3(\mathbb{P})$ (so, for each ϕ, $\phi :\rightarrow \_$ is a unary operation). These axioms are parameterized by action set $A_t = A \cup \{t\}$. We mostly suppress the $\cdot$ in process expressions, and brackets according to the following rules: $\cdot$ binds strongest, $:\rightarrow$ binds stronger than $\|$, $\mathbin{\|\!\_}$, $|$, all of which in turn bind stronger than $+$. We continue our explanation: now assume component C_1 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 [Kle56], defined by

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

(See also [BBP94].) 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 [Fok97]). 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

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

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 coarses 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 favour of the conditional. 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 non-deterministic 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 $R = \phi :\rightarrow P$ and $S = \neg\phi :\rightarrow Q$. Using $*$ and $+$, and their axioms, one easily derives $R^*(R^*S) = R^*S$ (this is an example in [BBP94]), proving our claim. So far our explanation of the primitives of ACP(A_t, γ) with three-valued conditions.

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, $x \lhd \phi \rhd x \neq x$, thus the principle of the excluded middle—*tertium non datur*—is dropped. As for the new constant μ, observe that

$$\mu \mid a = \mu$$

is derivable for all $a \in A_{t\delta}$ by (GT), (G4) and (GCM6), as well as $\mu \mid ax = \mu$. The axiom $\mu \mid x = \mu$ (M3) implies that $\mu \parallel x = \mu$ (cf. Lemma 3.1). 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. Finally observe that $\phi :\rightarrow x \mid \psi :\rightarrow y \neq \phi \wedge \psi :\rightarrow (x \mid y)$ (take $x = \mathsf{T} :\rightarrow \mu$ and $y = \mathsf{F} :\rightarrow \delta$). We use

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

both to refer to this axiom system and the signature thus defined.

In order to combine process algebra and three-valued logic, we introduce the 'rule of equivalence'

$$\text{(ROE)} \quad \frac{\models \phi = \psi}{\vdash \phi :\rightarrow x = \psi :\rightarrow x}$$

This rule reflects the 'rule of consequence' in Hoare's Logic (cf. [Apt81]). In our setting we shall use the following version: (ROE_M):

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

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. $\vdash \phi :\rightarrow \delta + x = \phi \vee \mathsf{T} :\rightarrow x,$
2. $\vdash \phi :\rightarrow x + y = \phi :\rightarrow x + \phi \vee \mathsf{T} :\rightarrow y,$
3. $\vdash \mu \,⌊\!⌊\, x = \mu,$
4. $\vdash \partial_H(\mu) = \mu,$
5. $\vdash t_I(\mu) = \mu.$

Proof. As for 1. This is just an application of axioms (GT) and (GC1).
As for 2. Use $\phi :\rightarrow x = \phi :\rightarrow (x + \delta) = \phi :\rightarrow x + \phi :\rightarrow \delta$, and apply 1 on $\phi :\rightarrow \delta + y$.
As for 3–5, just replace μ by $\mathsf{M} :\rightarrow x$ 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 expression in terms of the atomic actions that can be executed (if any), or its interpretation as meaningless. For example,

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

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 $\mathbb{T}_4$. As usual, we extend w to $\mathbb{BM}_3(\mathbb{P})$:

$$\begin{aligned}
w(c) &\stackrel{def}{=} c \text{ for } c \in \{\mathsf{M}, \mathsf{T}, \mathsf{F}\},\\
w(\neg\phi) &\stackrel{def}{=} \neg(w(\phi)),\\
w(\phi \diamondsuit \psi) &\stackrel{def}{=} w(\phi) \diamondsuit w(\psi) \text{ for } \diamondsuit \in \{\wedge, \vee, {}_{\circ}\!\wedge, {}^{\circ}\!\vee, \wedge_{\circ}, \vee^{\circ}\}.
\end{aligned}$$

With the system defined in Section 2, it follows that if $\models w(\phi) = w(\psi)$ for all $w \in \mathcal{W}$, then $\models \phi = \psi$. For each $w \in \mathcal{W}$ and $\phi \in \mathbb{BM}_3(\mathbb{P})$ we define inductively in Table 2 the unary predicate *meaningless*, notation

$$\mu(w, \_)$$

over process terms in $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$. This predicate defines which process expressions represent the meaningless process μ under a certain valuation w.

Table 2: Rules for μ in *panth*-format

μ	$\mu(w, \mu)$	
$:\rightarrow$	$\mu(w, \phi :\rightarrow x)$ if $w(\phi) = \mathsf{M}$	$\dfrac{\mu(w,x)}{\mu(w, \phi :\rightarrow x)}$ if $w(\phi) = \mathsf{T}$
$+, \cdot, \Vert, ⌊⌊, \vert, \partial_H, t_I$	$\dfrac{\mu(w,x)}{\begin{array}{l}\mu(w, x+y)\\ \mu(w, y+x)\\ \mu(w, x \cdot y)\\ \mu(w, x \parallel y)\\ \mu(w, x ⌊⌊ y)\\ \mu(w, x \mid y)\\ \mu(w, \partial_H(x))\\ \mu(w, t_I(x))\end{array}}$	

The axioms and rules for $\mu(w, \_)$ given in Table 2 are extended by axioms and rules given in Table 3, which define for $w \in \mathcal{W}$ and $a \in A_t$ transitions

$$\_ \xrightarrow{w,a} \_ \subseteq \mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) \times \mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}),$$

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

$$\_ \xrightarrow{w,a} \surd \subseteq \mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}).$$

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

The following facts follow easily by induction on the structure of the process expression involved, and clarify the relations between (termination) transitions and the predicate $\mu(w, \_)$:

Table 3: Transition rules in *panth*-format

$a \in A_t$	$a \xrightarrow{w,a} \surd$	
$\cdot, \lfloor\!\lfloor$	$\dfrac{x \xrightarrow{w,a} \surd}{\begin{array}{c} x \cdot y \xrightarrow{w,a} y \\ x \lfloor\!\lfloor y \xrightarrow{w,a} y \end{array}}$	$\dfrac{x \xrightarrow{w,a} x'}{\begin{array}{c} x \cdot y \xrightarrow{w,a} x'y \\ x \lfloor\!\lfloor y \xrightarrow{w,a} x' \parallel y \end{array}}$
$+, \parallel$	$\dfrac{x \xrightarrow{w,a} \surd \quad \neg\mu(w,y)}{\begin{array}{c} x + y \xrightarrow{w,a} \surd \\ y + x \xrightarrow{w,a} \surd \\ x \parallel y \xrightarrow{w,a} y \\ y \parallel x \xrightarrow{w,a} y \end{array}}$	$\dfrac{x \xrightarrow{w,a} x' \quad \neg\mu(w,y)}{\begin{array}{c} x + y \xrightarrow{w,a} x' \\ y + x \xrightarrow{w,a} x' \\ x \parallel y \xrightarrow{w,a} x' \parallel y \\ y \parallel x \xrightarrow{w,a} y \parallel x' \end{array}}$
$\mid, \parallel$ (communication)	$\dfrac{x \xrightarrow{w,a} \surd \quad y \xrightarrow{w,b} \surd}{\begin{array}{c} x \mid y \xrightarrow{w,c} \surd \\ x \parallel y \xrightarrow{w,c} \surd \end{array}} \quad a \mid b = c$	$\dfrac{x \xrightarrow{w,a} \surd \quad y \xrightarrow{w,b} y'}{\begin{array}{c} x \mid y \xrightarrow{w,c} y' \\ x \parallel y \xrightarrow{w,c} y' \end{array}} \quad a \mid b = c$
	$\dfrac{x \xrightarrow{w,a} x' \quad y \xrightarrow{w,b} \surd}{\begin{array}{c} x \mid y \xrightarrow{w,c} x' \\ x \parallel y \xrightarrow{w,c} x' \end{array}} \quad a \mid b = c$	$\dfrac{x \xrightarrow{w,a} x' \quad y \xrightarrow{w,b} y'}{\begin{array}{c} x \mid y \xrightarrow{w,c} x' \parallel y' \\ x \parallel y \xrightarrow{w,c} x' \parallel y' \end{array}} \quad a \mid b = c$
∂_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 $a \in I$
$:\to$	$\dfrac{x \xrightarrow{w,a} \surd}{\phi :\to x \xrightarrow{w,a} \surd}$ if $w(\phi) = \mathsf{T}$	$\dfrac{x \xrightarrow{w,a} x'}{\phi :\to x \xrightarrow{w,a} x'}$ if $w(\phi) = \mathsf{T}$

Lemma 4.1. *If $x \xrightarrow{w,a} x'$ or $x \xrightarrow{w,a} \surd$ for some w and a, then $\neg\mu(w,x)$.*
If $\mu(w,x)$, then $x \not\xrightarrow{w,a} x'$ and $x \not\xrightarrow{w,a} \surd$ for all $a \in A_t$.

Note that the converse implications do not hold (take $x = \delta$). The axioms and rules in Tables 2 and 3 yield a structured operational semantics (SOS) with negative premises. Furthermore, its transitions and meaningless instances are uniquely determined. This can be established with help of the following simple *stratification* S:

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

and the results of Groote [Gro93], Verhoef [Ver95], and Fokkink and Van Glabbeek [FG96]. Moreover, this SOS satisfies the so called *panth-format*, defined by Verhoef [Ver95], and implying that the notion of bisimulation equivalence induced by our SOS (see below) is a *congruence*.

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

- $\forall P' \; (P \xrightarrow{w,a} P' \implies \exists Q'(Q \xrightarrow{w,a} Q' \wedge P'BQ'))$,
- $\forall Q' \; (Q \xrightarrow{w,a} Q' \implies \exists P'(P \xrightarrow{w,a} P' \wedge P'BQ'))$,
- $P \xrightarrow{w,a} \surd \iff Q \xrightarrow{w,a} \surd$,
- $\mu(w,P) \iff \mu(w,Q)$.

Two processes P, Q are bisimilar, notation

$$P \leftrightarrows Q,$$

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

We obtain:

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

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

Proof. It is left to show that each axiom in Table 1 yields bisimilar instances. This can be checked by taking $B \stackrel{def}{=} \Delta \cup \{(i_l, i_r)\}$, where Δ is the diagonal, and the pair (i_l, i_r) represents the instance to be checked. As an example, consider (GCM7), i.e.,

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

Now $\mu(w, \_)$ either holds for both $\phi :\rightarrow a \mid \psi :\rightarrow bP$ and $\phi \wedge \psi :\rightarrow (a \mid b)P$, or not. This follows easily from the rules in Table 2. In the first case we are done; in the second case, either $\gamma(a,b) = \delta$, and both expressions cannot perform any transition, or both can perform

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

by the rules in Table 3. Thus B is a bisimulation, and $(\phi :\rightarrow a \mid \psi :\rightarrow bP,\; \phi \wedge \psi :\rightarrow (a \mid b)P) \in B$. ∎

5 Completeness

In this section we prove completeness of $\mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P}) + \mathrm{ROE}_\mathsf{M}$, i.e., if for process expressions 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 expression $P \in \mathrm{ACP}_\mu(A_t, \gamma, \mathbb{P})$ is a basic term if*

$$P \equiv \sum_{i \in I} \phi_i :\rightarrow Q_i$$

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

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

Proof. Standard induction on term complexity. ∎

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

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

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*

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

or the form

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

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

$$P \equiv \textstyle\sum_{i=1}^n \phi_i :\rightarrow Q_i$$

for some $n \geq 1$. By Lemma 3.1.1 we may assume that for all i either $Q_i \not\equiv \delta$ or $Q_i \equiv \delta$, which yields with axiom (GC1) form (1). Also with (GC1), each single action occurs at most once. This proves property (i) of form (2). Let

$$\overline{\phi} \equiv (\phi_1 \vee \mathsf{T}) \mathbin{\curlywedge} \ldots \mathbin{\curlywedge} (\phi_n \vee \mathsf{T}).$$

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

$$\begin{array}{lcl} \psi_i & \equiv & \overline{\phi} \mathbin{{}_{\circ}\!\!\wedge} \phi_i, \\ P'' & \equiv & \sum_{i=1}^{n} \psi_i :\rightarrow Q_i. \end{array}$$

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

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

by induction on n.
<u>$n = 1$.</u> This follows immediately from Lemma 2.1.
<u>$n = k + 1$.</u> Let $\overline{\overline{\phi}} \equiv (\phi_2 \vee \mathsf{T}) \mathbin{{}_{\circ}\!\!\wedge} ... \mathbin{{}_{\circ}\!\!\wedge} (\phi_n \vee \mathsf{T})$, and $\overline{\overline{\phi_i}} \equiv \overline{\overline{\phi}} \mathbin{{}_{\circ}\!\!\wedge} \phi_i$ for $i = 2, ...n$. By induction we have that

$$\vdash P = \phi_1 :\rightarrow Q_1 + \textstyle\sum_{i=2}^{n} \overline{\overline{\phi_i}} :\rightarrow Q_i.$$

With k applications of Lemma 3.1.2 and (GCL4), we obtain $\vdash P = \phi_1 :\rightarrow Q_1 + \sum_{i=2}^{n} \psi_i :\rightarrow Q_i$. Doing the same once more yields

$$\vdash P = (\psi_2 \vee \mathsf{T}) \mathbin{{}_{\circ}\!\!\wedge} \phi_1 :\rightarrow Q_1 + \textstyle\sum_{i=2}^{n} \psi_i :\rightarrow Q_i.$$

Now it follows easily that $\models \psi_1 = (\psi_2 \vee \mathsf{T}) \mathbin{{}_{\circ}\!\!\wedge} \phi_1$ (recall that $\psi_2 \equiv \psi \mathbin{{}_{\circ}\!\!\wedge} \phi_2$ 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 $\psi_i :\rightarrow Q_i$ it holds that $\models \psi_i = \psi_i \mathbin{{}_{\circ}\!\!\wedge} \mathsf{F}$, and thus

$$\begin{array}{lcl} \vdash \psi_i :\rightarrow Q_i & = & \psi_i \mathbin{{}_{\circ}\!\!\wedge} \mathsf{F} :\rightarrow Q_i \\ & = & \psi_i :\rightarrow (\mathsf{F} :\rightarrow Q_i) \\ & = & \psi_i :\rightarrow \delta. \end{array}$$

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

$$\vdash P = \psi_1 \vee ... \vee \psi_n :\rightarrow \delta.$$

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

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

with $k \leq n$ (and possibly some rearrangement of indices), and for each $i \in \{1, ..., k\}$ 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*

$$P_1 \mathrel{\underline{\leftrightarrow}} P_2 \implies \vdash P_1 = P_2.$$

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 $h = max(h(P_1), h(P_2))$.

<u>Let $h = 0$</u>, then $P_n \equiv \phi_n :\rightarrow \delta$ for $n = 1, 2$. So $\vdash \phi_1 :\rightarrow \delta \mathrel{\underline{\leftrightarrow}} \phi_2 :\rightarrow \delta$ and $w(\phi_1) = \mathsf{M} \Leftrightarrow w(\phi_2) = \mathsf{M}$.

This implies that $\mathbb{BM}_3(\mathbb{P}) \models \phi_1 \mathbin{\&\!\!\wedge} \mathsf{F} = \phi_2 \mathbin{\&\!\!\wedge} \mathsf{F}$. Now $\phi_n :\to \delta = \phi_n :\to (\mathsf{F} :\to \delta) = \phi_n \mathbin{\&\!\!\wedge} \mathsf{F} :\to \delta$. 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 $i \in I_n$, $Q_{n,i} \not\leftrightarrow Q_{n,j}$ 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: $R_{n,i} \leftrightarrow R_{n,j}$ 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 $w \in \mathcal{W}$.

- 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 $P_1 \leftrightarrow P_2$, 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 non-bisimilarity of different summands). Thus

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

- 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 $P_1 \leftrightarrow P_2$, there must be some unique $j \in I_{3-n}$ for which $P_{3-n} \xrightarrow{w,a} R_{3-n,j}$ and $R_{n,i} \leftrightarrow R_{3-n,j}$, 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:

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

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

6 Parameterized Actions and Non-Classical 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 values M (cf. [BS96]). 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

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

Let $Data^{0\text{-}5} \stackrel{def}{=} \{\rho \in Data^* \mid length(\rho) \leq 5\}$, 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

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

and let the following functions be defined:

$$\begin{array}{llll}
app & : Data \times Data^{0\text{-}5}_{\mathsf{M}} \to Data^{0\text{-}5}_{\mathsf{M}}, & app(d,\rho) = & \mathsf{M} \quad \text{if } length(\rho) = 5 \text{ or } \rho = \mathsf{M}, \\
head & : Data^{0\text{-}5}_{\mathsf{M}} \to Data, & head(\rho) = & \begin{cases} \mathsf{M} & \text{if } \rho \in \{\epsilon, \mathsf{M}\} \\ d & \text{if } \rho = app(d,\rho'), \rho' \in Data^{0\text{-}5}, \end{cases} \\
tail & : Data^{0\text{-}5}_{\mathsf{M}} \to Data^{0\text{-}5}_{\mathsf{M}}, & tail(\rho) = & \begin{cases} \mathsf{M} & \text{if } \rho \in \{\epsilon, \mathsf{M}\} \\ \rho' & \text{if } \rho = app(d,\rho'), \rho' \in Data^{0\text{-}5} \end{cases} \\
empty & : Data^{0\text{-}5}_{\mathsf{M}} \to \mathbb{T}^{\mathsf{M}}_3, & empty(\rho) = & \begin{cases} \mathsf{T} & \text{if } \rho = \epsilon \\ \mathsf{M} & \text{if } \rho = \mathsf{M} \\ \mathsf{F} & \text{otherwise,} \end{cases} \\
full & : Data^{0\text{-}5}_{\mathsf{M}} \to \mathbb{T}^{\mathsf{M}}_3, & full(\rho) = & \begin{cases} \mathsf{T} & \text{if } length(\rho) = 5 \\ \mathsf{M} & \text{if } \rho = \mathsf{M} \\ \mathsf{F} & \text{otherwise.} \end{cases}
\end{array}$$

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 expression $Q(x)$ describes manipulation with stack object $x \in Data^{0\text{-}5}$, including all "special cases" that evolve from $Data^{0\text{-}5}_{\mathsf{M}}$:

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

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 $\partial_H(Q(\mathsf{M}) \parallel R) = \mu$.
End example.

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

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

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 counter-like process:

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

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 $r(is\_zero)$ indicates that the counter value equals 0. It follows that

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

So, if in case of $C(0)$ the action $r(down)$ is performed, the counter evolves into μ.
End example.

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 $\mathsf{M} :\rightarrow x = \mu$. As a guard, M is totally persistent: whenever a condition to be evaluated in a process expression equals M, the process expression behaves as μ. In a parallel setting, this is much stronger than occurrence of F in a condition, e.g.,

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

In our set-up, $\mathsf{F} :\rightarrow \mu = \delta$ holds. This clearly illustrates that the guarded command reflects a certain order: first the condition is evaluated, then its right argument is considered. So, in this respect the guarded command really guards its process expression. 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 behaviour when we have one of P.

References

[Apt81] K.R. Apt. Ten years of Hoare's logic, a survey, part I. *ACM Transactions on Programming Languages and Systems*, 3(4):431–483, 1981.

[BB90] J.C.M. Baeten and J.A. Bergstra. Process algebra with signals and conditions. In M. Broy, editor, Programming and Mathematical Method, *Proceedings Summer School Marktoberdorf*, 1990 NATO ASI Series F, pages 273–323, Springer-Verlag, 1992.

[BBP94] J.A. Bergstra, I. Bethke, and A. Ponse. Process algebra with iteration and nesting. *Computer Journal*, 37(4):243–258, 1994.

[BBR95] J.A. Bergstra, I. Bethke, and P.H. Rodenburg. A propositional logic with 4 values: true, false, divergent and meaningless. *Journal of Applied Non-Classical Logics*, 5:199–217, 1995.

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

[BK84a] J.A. Bergstra and J.W. Klop. The algebra of recursively defined processes and the algebra of regular processes. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, *Algebra of Communicating Processes, Utrecht 1994*, Workshops in Computing, pages 1–25. Springer-Verlag, 1995. An extended abstract appeared in J. Paredaens, editor, *Proceedings* 11^{th} *ICALP,* Antwerp, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984.

[BK84b] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60(1/3):109–137, 1984.

[Boc39] D.A. Bochvar. On a 3-valued logical calculus and its application to the analysis of contradictions (in Russian). *Matématičeskij sbornik*, 4:287–308, 1939.

[BS96] J.A. Bergstra and M.P.A. Sellink. Sequential data algebra primitives. Technical Report P9602b, Programming Research Group, University of Amsterdam, 1996.

[BV93] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, editor, *Proceedings CONCUR 93,* Hildesheim, Germany, volume 715 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, 1993.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[Dij76] E.W. Dijkstra. *A Discipline of Programming.* Prentice Hall International, Englewood Cliff, 1976.

[FG96] W.J. Fokkink and R.J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126:1–10, 1996.

[Fok97] W.J. Fokkink. Axiomatizations for the perpetual loop in process algebra. In P. Degano, R. Gorrieri and A. Marchetti-Spaccamela (eds.), *Proc. 24th Colloquium on Automata, Languages and Programming - ICALP'97*, pages 571–581. Lecture Notes in Computer Science 1256, Springer-Verlag, 1997.

[Gro93] J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[HHJ+87] C.A.R. Hoare, I.J. Hayes, He Jifeng, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey, and B.A. Sufrin. Laws of programming. *Communications of the ACM*, 30(8):672-686, August 1987.

[Kal58] J. Kalman. Lattices with involution. *Trans. Am. Math. Soc.*, 87:485–491, 1958.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, pages 3–41. Princeton University Press, Princeton NJ, 1956.

[Kon96] B. Konikowska. McCarthy algebras: a model of McCarthy's logical calculus. *Fundamenta Informaticae*, 26:167–203, 1996.

[McC63] J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D. Hirshberg (eds.), *Computer Programming and Formal Systems*, pages 33-70, North-Holland, Amsterdam, 1963.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[Ver95] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274-302, 1995.