Information Processing Letters 67 (1998) 95–103

Kleene's three-valued logic and process algebra

Jan A. Bergstra [a,b,1], Alban Ponse [a,*]

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

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

Received 23 December 1997
Communicated by H. Ganzinger

Abstract

We propose a combination of Kleene's three-valued logic and ACP process algebra via the *guarded command* construct. We present an operational semantics in SOS-style, and a completeness result. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Process algebra; Three-valued logic; Guarded command; Design of algorithms; Concurrency; Formal languages

1. Introduction

In considering algorithms or programs in an operational manner, there is ample motivation to include a third truth value next to T (true) and F (false). For some illustrative references, see, e.g., [4,13]. Evaluation of the condition in a conditional construct, such as ϕ in

if ϕ **then** P **else** P,

for some program P may turn out divergent, or be distinguished as meaningless (e.g., a type clash, or division by zero). In such a case one certainly does not want to consider P and **if** ϕ **then** P **else** P as equal. Typically, the principle of the excluded middle—*tertium non datur*—is not anymore acceptable. Of course, **if** ϕ **then** P **else** P and **if** $\neg\phi$ **then** P **else** P should be considered the same.

In this paper we view process expressions with conditions as a vehicle to describe concurrent algorithms, and consider the question how to deal with a third truth value D, expressing *divergence*. This value is inspired by Kleene [15], in which it is called *undefined*, and is used to reason about partial recursive predicates being either undefined, true, or false. We rather use 'divergence' instead of 'undefined', as for example a type clash in a program is a kind of undefinedness that we want to distinguish from divergence. Naturally, $\neg\mathsf{D} = \mathsf{D}$, for divergence in the evaluation of a condition also implies divergence of its negation (cf. ϕ in **if** ϕ **then** P **else** P and **if** $\neg\phi$ **then** P **else** P).

We shortly recall the combination of process algebra and logic via the *guarded command*, an operation which stems from [11], and was introduced in process algebra with two-valued logic in [2] with the following typical laws where $\phi :\rightarrow \_$ is the guarded command resembling **if** ϕ **then** $\_$:

$$\mathsf{T} :\rightarrow x = x,$$

$$\mathsf{F} :\rightarrow x = \delta,$$

$$\phi :\rightarrow x + \psi :\rightarrow x = \phi \vee \psi :\rightarrow x.$$

Here $+$ denotes 'choice', and δ denotes 'inaction/deadlock'. The constant δ is well known in ACP based

---

* Corresponding author. Email: alban@wins.uva.nl.

[1] Email: janb@wins.uva.nl.

0020-0190/98/$19.00
PII: S0020-0190(98)00083-0

approaches [6,7,10], and is axiomatized by

$$x + \delta = x \quad \text{“inaction is not considered an alternative,}$$
$$\delta \cdot x = \delta \quad \text{… and is perpetual”.}$$

Here $\cdot$ represents “sequential composition”. We involve the constant D with the axiom

$$\mathsf{D} :\to x = \delta.$$

This preserves the three laws mentioned above in the present three-valued setting. Roughly, the idea is that if evaluation of a condition diverges, there is no point in considering it in the presence of an alternative, whereas it implies deadlock in case there are no alternatives. Now consider the derivations

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

$$\begin{aligned} \delta &= \delta + \delta \\ &= \mathsf{F} :\to x + \mathsf{D} :\to x \\ &= \mathsf{F} \vee \mathsf{D} :\to x. \end{aligned}$$

Clearly, the interpretation $\mathsf{D} :\to x = \delta$ leads to the logical consequence

$$\mathsf{T} \vee \mathsf{D} = \mathsf{T},$$

and leaves only two options for the definition of $\mathsf{F} \vee \mathsf{D}$, namely: $\mathsf{F} \vee \mathsf{D} \in \{\mathsf{F}, \mathsf{D}\}$. The only reasonable one seems $\mathsf{F} \vee \mathsf{D} = \mathsf{D}$. [2] So we end up with $\neg$, $\vee$ and its dual $\wedge$ as defined by the following truth tables:

x	$\neg x$
T	F
F	T
D	D

$\vee$	T	F	D
T	T	T	T
F	T	F	D
D	T	D	D

$\wedge$	T	F	D
T	T	F	D
F	F	F	F
D	D	F	D

This precisely entails Kleene’s three-valued logic as defined in [15], which we further call $\mathbb{K}_3$. (Notice that $\mathbb{K}_3$ is not functionally complete: one cannot define f with $f(\mathsf{D}) = \mathsf{F}$ and $f(v) = \mathsf{T}$ for $v \in \{\mathsf{T}, \mathsf{F}\}$.)

[2] By duality, the other option implies $\mathsf{T} \wedge \mathsf{D} = \mathsf{T}$, which indeed seems a rather implausible interpretation of $\wedge$.

Structure of the paper. In the next section we shortly discuss $\mathbb{K}_3$. 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 completeness result.

2. Kleene’s three-valued logic with propositions

Consider Kleene’s three-valued logic $\mathbb{K}_3$ as introduced in the previous section (cf. [15,3]). An equational specification of $\mathbb{K}_3$ follows from [14], and is given in Table 1. As usual, $\wedge$ and $\vee$ are commutative and associative operations. In case we use proposition symbols from set $\mathbb{P}$, we shall write $\mathbb{K}_3(\mathbb{P})$, and for concise notation we shall identify $\mathbb{K}_3$ and $\mathbb{K}_3(\emptyset)$.

Let $\mathbb{T}_3^{\mathsf{D}} = \{\mathsf{T}, \mathsf{F}, \mathsf{D}\}$. In the following we describe a prototypical, generic occurrence of D, starting from considerations that also apply to a two-valued setting. Consider the natural numbers

$$\omega = \{0, S(0), S(S(0)), \ldots\},$$

and write $S^0(x) = x$ and $S^{k+1}(x) = S(S^k(x))$. Let $f : \omega \to \mathbb{T}_3^{\mathsf{D}}$ be some arbitrary function. We define *infinitary f-disjunction*, notation $\bigvee f$, by

$$\bigvee f = f(0) \vee \bigvee (f \circ S).$$

The recursive definition of $\bigvee f$ implies computation of $f(0), f(S(0)), f(S^2(0)), \ldots$ until $f(n) = \mathsf{T}$ for some value n. In the particular case that for all $n \in \omega$, $f(n) = \mathsf{F}$, it makes sense to define $\bigvee f = \mathsf{D}$. We apply this idea in the following example.

Example 2.1. We define equality $\equiv : \omega \times \omega \to \mathbb{T}_3^{\mathsf{D}}$ as a binary infix function by

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

Next, we define the *partial predecessor* function $pprd : \omega \to \omega$ using auxiliary function $g : \omega \times \omega \to \omega$

$$pprd(x) = g(x, 0),$$
$$g(x, y) = \begin{cases} y & \text{if } S(y) \equiv x, \\ g(x, S(y)) & \text{otherwise.} \end{cases}$$

Table 1
Axiomatization of $\mathbb{K}_3$ with conjunction, disjunction, and implication.

(K1)	$\neg\mathsf{T} = \mathsf{F}$	(K6)	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
(K2)	$\neg\mathsf{D} = \mathsf{D}$	(K7)	$\mathsf{T} \wedge x = x$
(K3)	$\neg\neg x = x$	(K8)	$x \vee (x \wedge y) = x$
(K4)	$\neg(x \wedge y) = \neg x \vee \neg y$	(K9)	$x \wedge y = y \wedge x$
(K5)	$x \to y = \neg x \vee y$	(K10)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

One easily sees that

$$pprd(S^{k+1}(x)) \equiv S^k(x).$$

Now consider the case of $pprd(0)$. To model its computation, we define an auxiliary predicate *Aux* as follows:

$$Aux(x, y, z) \Leftrightarrow g(x, y) \equiv z.$$

The recursive definition of *Aux* follows easily from that of g, and falls within $\mathbb{K}_3(\mathbb{P})$:

$$\begin{aligned} Aux(x, y, z) = {} & (S(y) \equiv x \wedge y \equiv z) \\ & \vee \\ & (\neg(S(y) \equiv x) \wedge Aux(x, S(y), z)). \end{aligned}$$

In particular, $Aux(0, 0, z)$ models computation of $pprd(0)$. We have

$$\begin{aligned} Aux(0, 0, z) = {} & (S(0) \equiv 0 \wedge 0 \equiv z) \\ & \vee \\ & (\neg(S(0) \equiv 0) \wedge Aux(0, S(0), z)). \end{aligned}$$

By $\mathsf{T} \wedge x = x$ and $S(x) \equiv 0 = \mathsf{F}$, it follows that

$$\begin{aligned} Aux(0, 0, z) = {} & (S(0) \equiv 0 \wedge 0 \equiv z) \\ & \vee \\ & (S^2(0) \equiv 0 \wedge S(0) \equiv z) \\ & \vee \\ & (S^3(0) \equiv 0 \wedge S^2(0) \equiv z) \\ & \vee \\ & \ldots \end{aligned}$$

so, if $f = \lambda x.(S(x) \equiv 0 \wedge x \equiv z)$, we find

$$Aux(0, 0, z) = \bigvee f.$$

Furthermore, we have for each n that $f(n) = \mathsf{F}$ by axiom $S(x) \equiv 0 = \mathsf{F}$. Hence

$$Aux(0, 0, z) = \mathsf{D},$$

and thus $g(0, 0) \equiv z = \mathsf{D}$. The assumption that

$$pprd(0) = g(0, 0)$$

can be computed to some value z leads to value D of the predicate modeling its computation, irrespective of z. This motivates the following definitions:

$$pprd(0) = \mathsf{D},$$

$$\omega_{\mathsf{D}} = \omega \cup \{\mathsf{D}\},$$

so $pprd : \omega \to \omega_{\mathsf{D}}$. In order to integrate this example with process algebra, we extend the domains of all defined functions to ω_{D} by taking

$$S(\mathsf{D}) = \mathsf{D},$$

$$\mathsf{D} \equiv x = x \equiv \mathsf{D} = \mathsf{D},$$

$$pprd(\mathsf{D}) = \mathsf{D}.$$

We continue with this example after having combined $\mathbb{K}_3(\mathbb{P})$ with process algebra.

3. Process algebra with $\mathbb{K}_3(\mathbb{P})$

In the left column of Table 2 we present a slight modification of ACP(A, γ), the Algebra of Communicating Processes [6,7,10]. Here A is a set of atomic actions, and γ a communication function that is commutative and associative. We take γ *total* on $A \times A \to A_\delta$, where $A_\delta = A \cup \{\delta\}$, and the communication merge $|$ commutative (CMC) (by which (CM6) and (CM9), the symmetric variants of (CM5) and (CM8) [10], become derivable). In the right column additional axioms on pre-abstraction (t_I, i.e., renaming of all actions in I to action t), and guarded command are listed, where ϕ is taken from $\mathbb{K}_3(\mathbb{P})$. These axioms are parameterized by action set $A_t = A \cup \{t\}$. We mostly suppress the $\cdot$

Table 2
The axiom system $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$, where $a, b \in A_{t\delta}$, $H, I \subseteq A_t$.

(A1)	$x + (y + z) = (x + y) + z$		
(A2)	$x + y = y + x$		
(A3)	$x + x = x$		
(A4)	$(x + y)z = xz + yz$	(GT)	$\mathsf{T} :\rightarrow x = x$
(A5)	$(xy)z = x(yz)$	(GF)	$\mathsf{F} :\rightarrow x = \delta$
(A6)	$x + \delta = x$	(GD)	$\mathsf{D} :\rightarrow x = \delta$
(A7)	$\delta x = \delta$		
(CF1)	$a \mid b = \gamma(a, b)$ if $a, b \in A_t$	(GC1)	$\phi :\rightarrow x + \psi :\rightarrow x = \phi \vee \psi :\rightarrow x$
(CF2)	$a \mid \delta = \delta$	(GC2)	$\phi :\rightarrow x + \phi :\rightarrow y = \phi :\rightarrow (x + y)$
		(GC3)	$(\phi :\rightarrow x)y = \phi :\rightarrow xy$
(CM1)	$x \parallel y = (x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x) + x \mid y$	(GC4)	$\phi :\rightarrow (\psi :\rightarrow x) = \phi \wedge \psi :\rightarrow x$
(CM2)	$a \mathbin{⌊\!⌊} x = ax$	(GC5)	$\phi :\rightarrow x \mathbin{⌊\!⌊} y = \phi :\rightarrow (x \mathbin{⌊\!⌊} y)$
(CM3)	$ax \mathbin{⌊\!⌊} y = a(x \parallel y)$	(GCD)	$\phi :\rightarrow x \mid \psi :\rightarrow y = \phi \wedge \psi :\rightarrow (x \mid y)$
(CM4)	$(x + y) \mathbin{⌊\!⌊} z = x \mathbin{⌊\!⌊} z + y \mathbin{⌊\!⌊} z$		
(CMC)	$x \mid y = y \mid x$		
(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)	$(x + y) \mid z = x \mid z + y \mid z$		
(D1)	$\partial_H(a) = a$ if $a \notin H$	(T1)	$t_I(a) = a$ if $a \notin I$
(D2)	$\partial_H(a) = \delta$ if $a \in H$	(T2)	$t_I(a) = t$ if $a \in I$
(D3)	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	(T3)	$t_I(x + y) = t_I(x) + t_I(y)$
(D4)	$\partial_H(xy) = \partial_H(x)\partial_H(y)$	(T4)	$t_I(xy) = t_I(x)t_I(y)$

in process expressions, and brackets according to the following rules: $\cdot$ binds strongest, $:\rightarrow$ binds stronger than $\parallel$, $⌊\!⌊$, $\mid$, all of which in turn bind stronger than $+$. We use

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

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

$$\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P}) \vdash x = y,$$

or shortly $\vdash x = y$, if $x = y$ follows from the axioms of $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$ and $\mathbb{K}_3(\mathbb{P})$. The following derivabilities turn out to be useful:

Lemma 3.1.
(1) $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P}) \vdash \phi :\rightarrow \delta = \delta$,
(2) $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P}) \vdash \phi :\rightarrow x = \phi \vee \mathsf{D} :\rightarrow x$.

Proof. As for (1), $\phi :\rightarrow \delta = \phi :\rightarrow \delta + \mathsf{T} :\rightarrow \delta = \phi \vee \mathsf{T} :\rightarrow \delta = \mathsf{T} :\rightarrow \delta = \delta$.

As for (2), $\phi :\rightarrow x = \phi :\rightarrow x + \delta = \phi :\rightarrow x + \mathsf{D} :\rightarrow x = \phi \vee \mathsf{D} :\rightarrow x$. $\square$

We end this section by using the functions defined in Example 2.1 in a process algebraic setting.

Example 3.2. Recall the data type ω_{D}, and consider the following counter-like process with parameter in ω_{D}:

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

Here, action $r(up)$ models "receive command to increase", action $r(down)$ represents "receive command to decrease", action $r(set\_zero)$ can be used to reset the counter to $C(0)$, and action $r(is\_zero)$ indicates that the counter value equals 0. We find:

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

Clearly, this modeling is preferred to the case in which $pprd$ is replaced by $prd : \omega \to \omega$ with $prd(0) = 0$ and $prd(S(x)) = x$, which mixes up the number of $r(down)$ and $r(up)$ actions in the case of $C(0)$.

4. Operational semantics

In this section we provide $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$ with an operational semantics. Of course this semantics depends on interpretations of the propositions occurring in a process expression.

Assume a (non-empty) set $\mathbb{P}$ of proposition symbols, and let w range over the *valuations* (interpretations) $\mathcal{W}$ of $\mathbb{P}$ in $\mathbb{T}_3^{\mathsf{D}}$. In the usual way we extend w to $\mathbb{K}_3(\mathbb{P})$:

$$w(c) \stackrel{\triangle}{=} c \quad \text{for } c \in \{\mathsf{T}, \mathsf{F}, \mathsf{D}\},$$

$$w(\neg\phi) \stackrel{\triangle}{=} \neg(w(\phi)),$$

$$w(\phi \diamond \psi) \stackrel{\triangle}{=} w(\phi) \diamond w(\psi) \quad \text{for } \diamond \in \{\wedge, \vee\}.$$

It follows that if

$$\models w(\phi) = w(\psi)$$

for all $w \in \mathcal{W}$, then $\models \phi = \psi$, and thus $\vdash \phi = \psi$.

In Table 3 we give axioms and rules that define transitions

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

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

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

for all $w \in \mathcal{W}$ and $a \in A_t$. 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). So, a process expression either resembles deadlock (δ), or defines outgoing transitions with labels taken from $\mathcal{W} \times A_t$.

The axioms and rules in Table 3 yield a structured operational semantics (SOS) based on the work described by Groote and Vaandrager in [12]. In particular, this SOS satisfies the so-called *path-format* (see Baeten and Verhoef [9]), going with the following notion of bisimulation equivalence:

Definition 4.1. Let $B \subseteq \mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) \times \mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$. Then B is a *bisimulation* if for all $P,\ Q$ with PBQ the following conditions hold for all transitions $\_ \xrightarrow{l} \_$ and $\_ \xrightarrow{l} \surd$:

- $\forall P'\ (P \xrightarrow{l} P' \implies \exists Q'(Q \xrightarrow{l} Q' \wedge P'BQ'))$,
- $\forall Q'\ (Q \xrightarrow{l} Q' \implies \exists P'(P \xrightarrow{l} P' \wedge P'BQ'))$,
- $P \xrightarrow{l} \surd \iff Q \xrightarrow{l} \surd$,

Two processes $P,\ Q$ are *bisimilar*, notation

$$P \leftrightarrow Q,$$

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

According to [9], bisimilarity is a *congruence* relation. It is not difficult to establish with induction on the size of terms that in the bisimulation model thus obtained all equations of Table 2 are true. Hence we conclude:

Lemma 4.2. *The system* $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P})$ *is sound with respect to bisimulation*:

for all $P, Q \in \mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$,

$$\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P}) \vdash P = Q \implies P \leftrightarrow Q.$$

Table 3
Transition rules in *path*-format.

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

5. Completeness

In this section we prove completeness of $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P})$, i.e.,

$$P \leftrightarrow Q \quad \Longleftrightarrow \quad \mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P}) \vdash P = Q.$$

Our proof is based on a representation of process expressions for which bisimilarity implies derivability in a straightforward way.

Definition 5.1. A process expression $P \in \mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P})$ 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, $\phi_i \in \mathbb{K}_3(\mathbb{P})$, and $Q_i \in \{\delta, a, aR \mid a \in A_t,\ R \text{ a basic term}\}$.

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

Proof. Standard induction on term complexity. $\square$

For $a \in A_t$ and $\phi \in \mathbb{K}_3(\mathbb{P})$, 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 $\vdash P = P'$, $h(P') \leqslant h(P)$, and P' has either the form*

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

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) *for each $i \in I$ there is $w \in \mathcal{W}$ such that $w(\psi_i) = \mathsf{T}$,*
(iii) *for no $i \in I$ and valuation w, $w(\psi_i) = \mathsf{F}$.*

Proof. Assume

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

for some $n \geqslant 1$. By Lemma 3.1(1) we may assume that $Q_i \not\equiv \delta$ for all $i \in \{1, \ldots, n\}$. With (GC1) we easily obtain that each single action occurs at most once. This proves property (i) of the form (2).

Next we consider all summands from P for which no valuation makes the condition true. For each such summand $\phi_i :\rightarrow Q_i$ it holds that $\models \phi_i = \phi_i \wedge \mathsf{D}$, and thus $\vdash \phi_i = \phi_i \wedge \mathsf{D}$, by which

$$\begin{aligned}
\vdash \phi_i :\rightarrow Q_i &= \phi_i \wedge \mathsf{D} :\rightarrow Q_i\\
&= \phi_i :\rightarrow (\mathsf{D} :\rightarrow Q_i)\\
&= \phi_i :\rightarrow \delta\\
&= \delta.
\end{aligned}$$

In case all summands can be proved equal to $\phi_j :\rightarrow \delta$ in this way, we are done. In the other case we obtain

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

with $k \leqslant n$ (and possibly some rearrangement of indices), and for each $i \in \{1, \ldots, k\}$ there is a valuation w with $w(\phi_i) = \mathsf{T}$. This proves property (ii), and preserves property (i) for P. Finally we define

$$\psi_i \equiv \phi_i \vee \mathsf{D}$$

$$P' \equiv \sum_{i=1}^{k} \psi_i :\rightarrow Q_i.$$

By Lemma 3.1(2) we obtain

$$\vdash P = P'.$$

By definition of ψ_i it follows that $w(\psi_i) \neq \mathsf{F}$ for all w, i, which proves property (iii) for P'. (Properties (i) and (ii) are preserved for P'.) $\square$

With these two lemma's we can prove completeness:

Theorem 5.4. *The system* $\mathrm{ACP}_{\mathsf{D}}(A_t, \gamma, \mathbb{P}) + \mathbb{K}_3(\mathbb{P})$ *is complete with respect to bisimulation.*

Proof. Let $P_1 \leftrightarrow P_2$. By soundness, we may assume that both P_1 and P_2 satisfy the representation format defined in Lemma 5.3. We proceed by induction on $h = \max(h(P_1), h(P_2))$.

Case $h = 0$. By Lemma 3.1(1), $\vdash P_n = \delta$ for $n = 1, 2$, so $\vdash P_1 = P_2$.

Case $h > 0$. Let $P_n \equiv \sum_{i \in I_n} \psi_{n,i} :\to Q_{n,i}$ for $n = 1, 2$, so the P_n satisfy form (2) given in Lemma 5.3. 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) can be applied.

Now each summand of P_1 can be proved equal to one in P_2, and by Lemma 5.3, each such summand yields a transition for a certain $w \in \mathcal{W}$.

- Assume that $P_1 \xrightarrow{w,a} \surd$ for some w, a. Thus $w(\psi_{1,i}) = \mathsf{T}$ for some unique $i \in I_1$. By $P_1 \leftrightarrow P_2$, there is a unique $j \in I_2$ for which $P_2 \xrightarrow{w,a} \surd$ and $\models \psi_{1,i} = \psi_{2,j}$ (the latter derivability follows from Lemma 5.3 and the non-bisimilarity of different summands). Thus

  $$\vdash \psi_{1,i} :\to a = \psi_{2,j} :\to a.$$

- Assume that $P_1 \xrightarrow{w,a} R_{1,i}$ for some w, a and unique $i \in I_1$. Thus $w(\psi_{1,i}) = \mathsf{T}$. By $P_1 \leftrightarrow P_2$, there must be some unique $j \in I_2$ for which $P_2 \xrightarrow{w,a} R_{2,j}$ and $R_{1,i} \leftrightarrow R_{2,j}$, and for which $\models \psi_{1,i} = \psi_{2,j}$ follows from Lemma 5.3. By induction we find $\vdash R_{1,i} = R_{2,j}$, and therefore $\vdash aR_{1,i} = aR_{2,j}$ and hence

  $$\vdash \psi_{1,i} :\to aR_{1,i} = \psi_{2,j} :\to aR_{2,j}.$$

By the derivabilities above and symmetry, $\vdash P_1 = P_2$ quickly follows. $\Box$

6. Conclusion

The extension of process algebra with guarded command to a setting with Kleene's three-valued logic seems a modest one, and can be characterized as giving up the principle of the excluded middle, and hence giving up the identity

$$x = \phi :\to x + \neg\phi :\to x,$$

but otherwise no surprising identities arise: D and F often play the same role in guarded commands. This matches with the intuition that a process like

$$(\mathsf{D} :\to a) \parallel bc$$

equals $bc\delta$. The deadlock, caused by a divergence, is postponed until all alternative behaviour has been executed.

We have argued that divergence arises from considerations about partial predicates (cf. [15]), and can be involved in process algebra by $\mathsf{D} :\to x = \delta$. Of course, in the case that the *process* of evaluation is prominent in the algorithm represented as a process expression, evaluation rather should be modeled as a process (which possibly diverges) than as a condition.

References

[1] K.R. Apt, Ten years of Hoare's logic, a survey, Part I, ACM Trans. Programming Languages Systems 3 (4) (1981) 431–483.

[2] J.C.M. Baeten, J.A. Bergstra, Process algebra with signals and conditions, in: M. Broy (Ed.), Programming and Mathematical Method, Proceedings Summer School Marktoberdorf, 1990 NATO ASI Series F, Springer, Berlin, 1992, pp. 273–323.

[3] J.A. Bergstra, I. Bethke, P.H. Rodenburg, A propositional logic with 4 values: true, false, divergent and meaningless, J. Appl. Non-Classical Logics 5 (1995) 199–217.

[4] H. Barringer, J.H. Cheng, C.B. Jones, A logic covering undefinedness in program proofs, Acta Inform. 21 (1984) 251–269.

[5] S.D. Brookes, C.A.R. Hoare, A.W. Roscoe, A theory of communicating sequential processes, J. ACM 31 (3) (1984) 560–599.

[6] J.A. Bergstra, J.W. Klop, The algebra of recursively defined processes and the algebra of regular processes, in: A. Ponse, C. Verhoef, S.F.M. van Vlijmen (Eds.), Algebra of Communicating Processes, Utrecht 1994, Workshops in Computing, Springer, Berlin, 1995, pp. 1–25. An extended abstract appeared in: J. Paredaens (Ed.), Proceedings 11th ICALP, Antwerp, Lecture Notes in Computer Science, Vol. 172, Springer, Berlin, 1984, pp. 82–95.

[7] J.A. Bergstra, J.W. Klop, Process algebra for synchronous communication, Inform. and Comput. 60 (1–3) (1984) 109–137.

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

[9] J.C.M. Baeten, C. Verhoef, A congruence theorem for structured operational semantics with predicates, in: E. Best (Ed.), Proceedings CONCUR 93, Hildesheim, Germany, Lecture Notes in Computer Science, Vol. 715, Springer, Berlin, 1993, pp. 477–492.

[10] J.C.M. Baeten, W.P. Weijland, Process Algebra, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

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

[12] J.F. Groote, F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence, Inform. and Comput. 100 (2) (1992) 202–260.

[13] C.B. Jones, C.A. Middelburg, A typed logic of partial functions reconstructed classically, Acta Inform. 31 (5) (1994) 399–430.

[14] J. Kalman, Lattices with involution, Trans. Amer. Math. Soc. 87 (1958) 485–491.

[15] S.C. Kleene, On a notation for ordinal numbers, J. Symbolic Logic 3 (1938) 150–155.