Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

The logic of ACP

A. Ponse, M.B. van der Zwaag

REPORT SEN-R0207 MARCH 31, 2002

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

The Logic of ACP

Alban Ponse[1,2] & Mark B. van der Zwaag[1]

[1] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

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

ABSTRACT
We distinguish two interpretations for the truth value 'undefined' in Kleene's three-valued logic. Combining these two interpretations leads to a four-valued propositional logic that characterizes two particular ingredients of process algebra: "choice" and "inaction". We study two different bases for this logic, and prove some elementary results (on expressiveness and completeness). One has the classical symmetric connective conjunction and negation, while the other one only has a ternary if-then-else connective with a sequential, operational flavor. Combining this four-valued logic with process algebra yields a direct generalization of ACP with conditional composition that establishes the characterization of choice and inaction. For this generalization we present an operational semantics in SOS-style and some completeness results.

2000 Mathematics Subject Classification: 68Q55; 68Q60; 68Q70; 68Q85.
Keywords & Phrases: Process algebra, ACP, logic, many-valued logics, conditional composition.
Note: Work carried out under project SEN2.1 "Process Specification and Analysis".

1 Introduction

Process algebra is a generic term that refers to the study of 'concurrency theory' (or 'process theory') in an algebraic fashion. In this paper we attempt to approach process algebra from a logical perspective. This is, of course, not the intended approach; process algebra is algebraically based, and focuses attention to applications (the specification and verification of distributed systems) and to algebraic (mathematical) results. Nevertheless, we think it is worth the effort to consider the primitives of process algebra from a different angle, and to weigh their merits from a logical perspective because this may further illuminate some particular design choices for the primitives and laws of process algebra.

We shall identify 'process algebra' with ACP (Algebra of Communicating Processes), the modular process algebra framework designed by Bergstra and Klop from 1982 onwards [4] (for an overview of the current state of the art in process algebra we refer to [9]). The most basic part of ACP is called BPA (Basic Process Algebra) and comprises two operations:

Sequential composition, as known from any imperative programming language (usually written ";"), and

Alternative composition, or *choice*, in principle a descriptive feature that is absent in sequen-

tial, imperative programming languages.[1]

The motivation for the alternative composition operation arises if concurrency is approached in an analytical, discrete fashion: if

$$a \,||\, b$$

expresses the concurrent execution of atomic instanteneous behaviors a and b, an observer experiences

- either a followed by b, or
- b followed by a, or
- a and b simultaneously.[2] (This can be thought of as a synchronization or communication between a and b.)

Such atomic, instantenious behaviors are further called *actions*. This approach to concurrency is sometimes referred to as the *interleaving hypothesis*:

> Concurrency can be analyzed or specified in terms of interleaving and synchronization of actions, by means of alternative and sequential composition.

A well-known ACP axiom characterizing the interleaving hypothesis is

$$x \,||\, y = (x \between y + y \between x) + x \mid y,$$

where $+$ stands for alternative composition. It states that in the parallel composition $x \,||\, y$ of x and y, either $x \between y$ is executed, or $y \between x$, or $x \mid y$. Here $x \between y$ is the same as parallel composition with the restriction that the first action stems from x, and $x \mid y$ is the same as parallel composition but with the restriction that the first action is a synchronization between a first action of x and one of y. We note that these operations together have a simple, algebraic axiomatization in ACP (a historical reference is [4]).

Once sequential and alternative composition are accepted as primitives, it makes sense to analyze these operations in detail. While the first one does not raise particular questions, this is not the case for $+$ (choice being further away from the human condition than ordinary sequential composition).

Alternative composition becomes even a much more involved concept if a notion of deadlock or inaction is included as a primitive behavior, that is, once we admit two types of behavioral stability:

Termination (short for *successful termination*): nothing happens anymore because all that should have happened, has happened, and

Inaction or *deadlock*: nothing can happen anymore because execution is stuck and has brought us to a point of no return.

[1] It is the mathematizement of the concept that in $x + y$ either x or y is (or can be) executed, however, we cannot predict which of the two; that is beyond any means of analysis.

[2] If a and b are thought of as colored light-flashes, e.g., yellow and blue, respectively, this makes sense: either yellow/blue, blue/yellow of a green flash may be observed. A second illustration goes with the assumption that actions are short beeps (like those in many operating systems for attracting the user's attention).

Of course, at least one of these kinds of behavioral stability requires explicit notation, and in ACP this is 'inaction', written as δ.

We first explain the difference between inaction and termination in terms of sequential composition, notation $\cdot$, i.e., the multiplication symbol (with the convention to omit this symbol in terms): let a, b be actions, then

$$a\delta = (a\delta)b$$

while, of course, $a \neq ab$.[3] The idea that after inaction nothing can happen is axiomatized by

$$\delta x = \delta$$

and by the assumption that sequential composition is associative,

$$(xy)z = x(yz),$$

an assumption that can hardly be rejected. (Quite naturally, $x\delta$ cannot be further reduced.) So, a represents the execution of the action a after which termination occurs, and $a\delta$ represents the behavior of a followed by inaction.

Having accepted the termination convention described above (explicit notation for inaction), one is faced with the question whether

$$x + \delta$$

can be reduced, and if so, to what. In principle, two reductions seem likely: either x or δ. The axiom for the interleaving hypothesis given above yields

$$a \,||\, \delta = (a \between \delta + \delta \between a) + a \mid \delta,$$

where the right-hand side equals

$$(a\delta + \delta) + \delta,$$

since $\delta \between a = \delta$ holds because the left argument cannot perform an action, and $a \mid \delta = \delta$ holds because δ cannot participate in a synchronization.[4] Hence, the choice $x + \delta = x$ leads to $a \,||\, \delta = a\delta$, while the alternative $x + \delta = \delta$ leads to $a \,||\, \delta = \delta$. Clearly, the latter does not match the interleaving hypothesis, whereas the law $x + \delta = x$ is an axiom of ACP.

So, choice is subsidiary to the ability to perform activity in ACP. One may call this, and thus the axiom $x + \delta = x$ "optimistic choice", pessimistic choice being axiomatized by $x + \delta = \delta$. (The latter option is characterized by the *chaos* constant χ in Hoare's [13], and can be combined with δ in a single framework, for instance as the *meaningless* constant in [6, 5].)

We mentioned earlier that alternative composition is primarily a *descriptive* feature: it is used to put together possible behaviors, while the nature of the choice between alternatives cannot be accessed. However, this reading does not combine well with the law $x + \delta = x$, which implies that δ is not a fair choice.

[3] In terms of the last illustration with beeps for (the occurrence of) actions, deadlock or inaction can be thought of as a soft buzzing, while termination is still associated with total silence.

[4] Referring to the last illustration, $a \,||\, \delta$ would model the situation in which the beep a can be heard, and after that the deadlock buzzing (think of two computers). This is the same behavior as $a\delta$, which indeed equals $a \between \delta$. Further note that communication can be explained in terms of a louder beep (either twintone or not).

On the other hand, our understanding of sequential composition is unproblematic — whether it is read prescriptive or descriptive.

We propose to *generalize* alternative composition, such that it becomes a prescriptive construct: we add information about the choice between alternatives as a side-condition of the composition. Thus, we obtain *conditional composition*; let

$$x +_\phi y$$

stand for the choice between x and y, under the *condition* ϕ. This construction is well-known from imperative programming languages, where it is usually written in the form *if* ϕ *then* x *else* y.

At this point we may adapt a logical perspective: let

$$\mathrm{C}$$

stand for the logical truth value that represents 'either true or false' or 'overdefined', and let

$$\mathrm{D}$$

stand for the logical truth value 'neither true nor false' or 'undefined'.

Now, alternative composition and inaction can be viewed as the instances $+_\mathrm{C}$ and $+_\mathrm{D}$ of conditional composition respectively. We find, with T representing 'true' and F representing 'false':

$$\begin{aligned} x +_\mathrm{C} y &= x + y,\\ x +_\mathrm{T} y &= x,\\ x +_\mathrm{F} y &= y,\\ x +_\mathrm{D} y &= \delta. \end{aligned}$$

We introduce a four-valued propositional logic over the truth values $\{\mathrm{C}, \mathrm{T}, \mathrm{F}, \mathrm{D}\}$ that takes conditional composition as a primitive (*in* the logic), and in which the interplay between conditions can be studied.

It turns out that this logic is both straightforward and elegant, and also has a classical basis. Finally, there is a straightforward correspondence with the process algebraic conditional composition, allowing one to explain the nature of choice in process algebra, and its interplay with δ, from a logical perspective.

Structure of the paper. In Section 2, we introduce the four-valued logic $\mathbb{L}_4$ that has a conditional composition connective as primitive operation. We show that this logic is equivalent with the logic that arises naturally when distinguishing two readings of the truth value for 'undefined' in Kleene's three-valued logic. We present results on expressiveness, and complete axiomatizations.

In Section 3, we generalize process algebra in the manner suggested above, starting with the generalization of alternative composition in BPA, a subsystem of ACP. We prove completeness for the presented axiom system. Furthermore, we establish a correspondence between a class of $\mathbb{L}_4$ identities and process algebra identies. Then, we also introduce a generalization of the parallel composition operation of ACP. We give, as an example of the use of the generalized operations, a specification of a scheduling mechanism for parallel processes.

Section 4 is devoted to a full and detailed proof of the completeness of our $\mathbb{L}_4$ axioms. This (non-trivial) proof essentially uses a normal form representation for open terms.

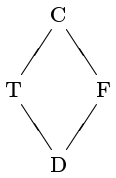
In Section 5, the article is ended with some conclusions that relate its contents with earlier work.

2 Four-valued propositional logic

In this section we introduce two propositional logics over the truth values discussed above. First a logic that takes conditional composition as the only operation, and secondly, one that is based on the classical connectives and can be seen as a natural generalization of Kleene's partial logic. We show that these logics are equal in terms of expressiveness, and provide complete axiomatizations for both.

2.1 The logic $\mathbb{L}_4$

We introduce a four-valued logic with set $T_4 = \{\mathrm{C}, \mathrm{T}, \mathrm{F}, \mathrm{D}\}$ of truth values. These truth values can be partially ordered according to the lattice (1) below, that we call the *information ordering* (see Section 5 for some more comments).



(1)

Here the value D can be read as *undefined* (giving less information than T or F) and C as *overdefined* or being either T or F. Let $x \sqcup y$ represent the least upper bound of x and y in the information ordering.

The primary operation that we consider is the ternary operation $\_ \lhd \_ \rhd \_$, called *conditional composition*,[5] that is defined by

$$x \lhd \mathrm{C} \rhd y = x \sqcup y,$$
$$x \lhd \mathrm{T} \rhd y = x,$$
$$x \lhd \mathrm{F} \rhd y = y,$$
$$x \lhd \mathrm{D} \rhd y = \mathrm{D}.$$

So, the auxiliary operation $\sqcup$ stands for $\lhd \mathrm{C} \rhd$. We prefer to view conditional composition as a primary operation because it corresponds with the process algebraic conditional composition $+_\phi$ (see Section 3.1) and because it has an operational, sequential flavor, i.e., it can be associated with an order of evaluation: in the evaluation of the term $x \lhd y \rhd z$, first y is evaluated, and depending on the outcome, possibly x and/or z. Moreover, a logic with a single operation can be of technical convenience (cf. the proof of Theorem 2.1).

Assume a set V of variables. Terms are formed using the constants from T_4, variables from V, and the operations just introduced. A *valuation* is a mapping from V to T_4. Clearly, every valuation extends to an interpretation mapping from terms to T_4. Two terms are equivalent

[5] This notation stems from [15], in which $x \lhd y \rhd z$ models `if` y `then` x `else` z.

if they have the same interpretation under every valuation. We write $\mathbb{L}_4$ for the resulting logic.

Having introduced the logic $\mathbb{L}_4$, we discuss some of its properties. First, conditional composition distributes over $\sqcup$: let $\mathbf{v}$ abbreviate $v_1 \sqcup v_2$, then

$$\begin{aligned}\mathbf{x} \lhd \mathbf{y} \rhd \mathbf{z} &= (\mathbf{x} \lhd y_1 \rhd \mathbf{z}) \sqcup (\mathbf{x} \lhd y_2 \rhd \mathbf{z}) \\ &= (x_1 \lhd \mathbf{y} \rhd \mathbf{z}) \sqcup (x_2 \lhd \mathbf{y} \rhd \mathbf{z}) \\ &= (\mathbf{x} \lhd \mathbf{y} \rhd z_1) \sqcup (\mathbf{x} \lhd \mathbf{y} \rhd z_2). \end{aligned} \tag{2}$$

Furthermore, we can also define negation:

$$\neg \mathrm{T} = \mathrm{F};\ \neg \mathrm{F} = \mathrm{T};\ \neg \mathrm{C} = \mathrm{C};\ \neg \mathrm{D} = \mathrm{D}.$$

Note that the invariance of C and D under negation follows quite naturally from the reading giving above. Negation is definable from conditional composition and the truth values T and F:

$$\neg x = \mathrm{F} \lhd x \rhd \mathrm{T}.$$

We adopt the following binding convention: negation binds stronger than conditional composition, which binds stronger than $\sqcup$.

Finally, negation distributes over $\sqcup$, and it holds that

$$\begin{aligned} x \lhd y \rhd z &= z \lhd \neg y \rhd x, \\ \neg(x \lhd y \rhd z) &= \neg x \lhd y \rhd \neg z. \end{aligned}$$

Expressive adequacy

We show that, with respect to the information ordering (1), the logic $\mathbb{L}_4$ is truth-functionally complete for monotone functions. Recall that an n-ary function f over T_4 is *monotone* with respect to a partial ordering $\leq$ on T_4, if whenever $a_i \leq b_i$ for $1 \leq i \leq n$, then

$$f(a_1, \ldots, a_n) \leq f(b_1, \ldots, b_n).$$

Note that, according to the information ordering lattice, all operations introduced so far are monotone. For conditional composition, this follows from the fact that $x \leq y$ if and only if $x \sqcup y = y$ and that it distributes over $\sqcup$ (see (2)).

Now, a function $f : T_4^n \to T_4$, with $n \geq 1$, can be expressed in $\mathbb{L}_4$ if there is a term t with variables $x_1, \ldots, x_n$, and no others, such that

$$f(a_1, \ldots, a_n) = t[a_1/x_1, \ldots, a_n/x_n]$$

for all $a_1, \ldots, a_n \in T_4$.

If every *monotone* function over T_4 can be expressed in a logic, then that logic is called *expressively adequate* (this terminology is taken from [11]).

Theorem 2.1. The logic $\mathbb{L}_4$ is expressively adequate, that is, every monotone n-ary function over T_4 (with $n \geq 1$) can be expressed in $\mathbb{L}_4$.

Proof. Let $f : T_4^{k+1} \to T_4$ be monotone, and write $\bar{x}, y$ for $k+1$ tuples ($\bar{x}$ may be empty). Then

$$f(\bar{x}, y) = f(\bar{x}, \mathrm{D}) \sqcup f(\bar{x}, \mathrm{T}) \triangleleft y \triangleright f(\bar{x}, \mathrm{F}) \sqcup (\mathrm{D} \triangleleft y \triangleright f(\bar{x}, \mathrm{C})) \triangleleft y \triangleright \mathrm{D}$$

by monotonicity of f. By induction on k, the function f is expressible (because $f(\bar{x}, a)$ is, for all $a \in T_4$). □

Non-monotone functions cannot be expressed in $\mathbb{L}_4$; however, below we show that the inclusion of a single non-monotone operation results in a truth-functionally complete logic (Theorem 2.2).

2.2 Kleene's logic and the logic $\mathbb{K}_4$

In the previous section, we introduced the four-valued propositional logic $\mathbb{L}_4$, that has a single operation that may be considered not so standard. In this section we show that this logic can be obtained also by extending Kleene's three-valued logic [17], that we call $\mathbb{K}_3$, in the following way: we distinguish two interpretations of Kleene's third truth value 'undefined' and show that the resulting logic has exactly the same expressivity as $\mathbb{L}_4$ (where of course Kleene's logic has the familiar primitive operations negation and conjunction).

First we present the three-valued logic $\mathbb{K}_3$ that is also known as *partial* logic. This logic has, besides the classical truth values true (T) and false (F), a third truth value $*$, that may be read as either *undefined* or *overdefined* (being either true or false, but one cannot predict which of the two). Its basic operations are negation and conjunction defined by the truth tables below.

$\neg$	
T	F
F	T
$*$	$*$

$\wedge$	T	F	$*$
T	T	F	$*$
F	F	F	F
$*$	$*$	F	$*$

Other operations, like disjunction and implication, are defined in terms of these in the familiar way; in particular, disjunction is defined by

$$x \vee y = \neg(\neg x \wedge \neg y).$$

Kleene's three-valued logic $\mathbb{K}_3$ was designed in order to deal with partial recursive functions: if a partial function f is not defined for argument a, and the truth value of the term t depends on $f(a)$, then t may be classified as $*$. However, a term may still make sense, that is, have a definite truth value, even if it has indefinite subterms; for example, $\mathrm{F} \wedge t$ equals F, even if t is classified as $*$.

We shall now make an explicit distinction between the two possible readings of the third truth value: we replace the value $*$ by the two distinct truth values C and D, such that the restriction from four to three truth values by identifying these two yields again Kleene's logic. This leads to the following (incomplete) truth tables:

$\neg$	
C	C
T	F
F	T
D	D

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

In the following we argue that $\mathrm{C} \wedge \mathrm{D} = \mathrm{D} \wedge \mathrm{C} = \mathrm{F}$ (and hence that $\mathrm{C} \vee \mathrm{D} = \mathrm{D} \vee \mathrm{C} = \mathrm{T}$) and that there are no more than two possible readings of the third truth value $*$. Observe that absorption ($x = x \wedge (x \vee y)$) is valid in $\mathbb{K}_3$, and so are commutativity, associativity and idempotence of conjunction. Now $\mathrm{C} \wedge \mathrm{D} \notin \{\mathrm{C}, \mathrm{D}\}$ by absorption and the identity $\mathrm{C} \vee \mathrm{D} = \neg(\neg\mathrm{C} \wedge \neg\mathrm{D})$, for suppose $\mathrm{C} \wedge \mathrm{D} = \mathrm{D}$. Then

$$\mathrm{C} = \mathrm{C} \wedge (\mathrm{C} \vee \mathrm{D}) = \mathrm{C} \wedge \neg(\mathrm{C} \wedge \mathrm{D}) = \mathrm{C} \wedge \neg\mathrm{D} = \mathrm{D}.$$

(In the same way, $\mathrm{C} \wedge \mathrm{D} = \mathrm{C}$ can be refuted.) By associativity and idempotence of conjunction, $\mathrm{C} \wedge \mathrm{D} \neq \mathrm{T}$ (consider $\mathrm{C} \wedge \mathrm{C} \wedge \mathrm{D}$). Now assume that $*$ admits a third interpretation, say E, and $\mathrm{C} \wedge \mathrm{D} = \mathrm{E}$ (and thus $\mathrm{C} \vee \mathrm{D} = \mathrm{E}$). Then we derive $\mathrm{E} = \mathrm{C}$ as follows:

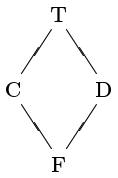
$$\begin{aligned} &\mathrm{C} = \mathrm{C} \wedge (\mathrm{C} \vee \mathrm{D}) = \mathrm{C} \wedge \mathrm{E} \\ &\Rightarrow \mathrm{C} = \mathrm{E} \vee \mathrm{C} \\ &\Rightarrow \mathrm{E} = \mathrm{E} \wedge (\mathrm{E} \vee \mathrm{C}) = \mathrm{E} \wedge \mathrm{C} = \mathrm{C}. \end{aligned}$$

This shows that $\mathrm{C} \wedge \mathrm{D} = \mathrm{F}$, and it remains to be shown that with this identity the assumption above, i.e., the existence of a third reading E, is not compatible with C and D. Suppose the contrary. Then, as above, it follows that $\mathrm{C} \wedge \mathrm{E} = \mathrm{D} \wedge \mathrm{E} = \mathrm{F}$. Because distributivity is valid in $\mathbb{K}_3$, we can derive

$$\mathrm{C} = \mathrm{C} \wedge \mathrm{T} = \mathrm{C} \wedge (\mathrm{D} \vee \mathrm{E}) = (\mathrm{C} \wedge \mathrm{D}) \vee (\mathrm{C} \wedge \mathrm{E}) = \mathrm{F} \vee \mathrm{F} = \mathrm{F},$$

which concludes our argument.

Thus, we have extended $\mathbb{K}_3$ in a natural way to a four-valued logic that we shall refer to as $\mathbb{K}_4$ (this logic was introduced in [7]). We mention some properties of the operations of $\mathbb{K}_4$. First, conjunction and disjunction are the greatest lower bound and the least upper bound according to the ordering (3) depicted below.



(3)

Moreover, this lattice, with $\wedge$ and $\vee$, is distributive, and negation is an *involution* for this lattice (cf. [16]), that is, it holds that

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

In Section 2.4, we shall see that this characterization of the logic as a distributive lattice with involution leads directly to a complete axiomatization.

2.3 Equivalence of $\mathbb{K}_4$ and $\mathbb{L}_4$

We show that the logics $\mathbb{K}_4$ and $\mathbb{L}_4$ have exactly the same expressivity, i.e., their operations can be defined in terms of the operations of the other logic. Hence, the two logics can be considered "the same", but with a different functional basis. So, we can freely use those operations that

seem most appropriate. We adopt the following binding convention: negation binds stronger than conjunction and disjunction, which bind stronger than conditional composition, which binds stronger than $\sqcup$.

The operations negation, conjunction and disjunction can all be defined in terms of conditional composition and the truth values C, T, and F (recall that $\sqcup$ abbreviates $\lhd \mathrm{C} \rhd$):

$$\neg x = \mathrm{F} \lhd x \rhd \mathrm{T}, \tag{4}$$
$$x \wedge y = y \lhd x \rhd \mathrm{F} \sqcup x \lhd y \rhd \mathrm{F}, \tag{5}$$
$$x \vee y = \mathrm{T} \lhd x \rhd y \sqcup \mathrm{T} \lhd y \rhd x. \tag{6}$$

Vice versa, conditional composition can be defined in terms of negation, conjunction, disjunction and the truth value D:

$$x \lhd y \rhd z = ((x \wedge y) \vee (z \wedge \neg y)) \vee (((x \wedge z) \wedge \mathrm{D}) \vee ((y \wedge \neg y) \wedge \mathrm{D})). \tag{7}$$

Functional completeness

Because all operations of $\mathbb{L}_4$ (and thus $\mathbb{K}_4$) are monotone, we cannot express non-monotone functions on the truth values. We show that with the addition of one non-monotone operation, we can express every truth-functional operation. The unary definedness operation $\downarrow$ (see [3]) is defined by

$$\downarrow \mathrm{C} = \mathrm{F};\ \downarrow \mathrm{T} = \mathrm{T};\ \downarrow \mathrm{F} = \mathrm{T};\ \downarrow \mathrm{D} = \mathrm{F}.$$

This operation is not monotone; for example, it holds that $\mathrm{T} \leq \mathrm{C}$ while $\downarrow \mathrm{T} \not\leq \downarrow \mathrm{C}$.

Theorem 2.2. With the addition of the definedness operation $\downarrow$ to $\mathbb{K}_4$ or $\mathbb{L}_4$, we obtain a logic that is truth-functionally complete.

Proof. It is sufficient to prove this for $\mathbb{K}_4$. We introduce auxiliary operations $\kappa_a(\_)$ that satisfy

$$\kappa_a(b) = \begin{cases} \mathrm{T} & \text{if } a = b, \\ \mathrm{F} & \text{otherwise,} \end{cases}$$

for $a, b \in T_4$:

$$\kappa_{\mathrm{C}}(x) = \downarrow((x \wedge \neg x) \vee \mathrm{D}),$$
$$\kappa_{\mathrm{T}}(x) = \downarrow x \wedge x,$$
$$\kappa_{\mathrm{F}}(x) = \kappa_{\mathrm{T}}(\neg x),$$
$$\kappa_{\mathrm{D}}(x) = \downarrow((x \wedge \neg x) \vee \mathrm{C}).$$

Let f be a function from T_4^{k+1} to T_4, for some $k \geq 0$. We apply induction on k. Write $\bar{x}, y$ for $k+1$ tuples. We define

$$f(\bar{x}, y) = \bigvee_{a \in T_4} (\kappa_a(y) \wedge f(\bar{x}, a)).$$

□

Table 1: Axioms of $\mathbb{K}_4$.

(N0)	$\neg(x \wedge y) = \neg x \vee \neg y$
(N1)	$\neg\neg x = x$
(N2)	$\neg \mathrm{T} = \mathrm{F}$
(N3)	$\neg \mathrm{C} = \mathrm{C}$
(N4)	$\neg \mathrm{D} = \mathrm{D}$
(K1)	$x \wedge y = y \wedge x$
(K2)	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
(K3)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
(K4)	$x \vee (x \wedge y) = x$
(K5)	$\mathrm{T} \wedge x = x$
(K6)	$\mathrm{C} \wedge \mathrm{D} = \mathrm{F}$

2.4 Axioms for $\mathbb{L}_4$ and $\mathbb{K}_4$

We start with the axiomatization of $\mathbb{K}_4$ presented in Table 1. The axioms K1–K6 reflect that the lattice (3) is distributive. Axioms N0 and N1 reflect that negation is a so-called involution for this lattice. Axiom N0 is, in the presence of axiom N1, equivalent with the definition of disjunction in terms of negation and conjunction.

Theorem 2.3. The axioms for $\mathbb{K}_4$ in Table 1 are complete.

Proof. The following completeness proof for this axiomatization is due to Bas Luttik and Piet Rodenburg; it is based on [16].

Let the $\mathbb{K}_4$ axioms in Table 1 denote the variety of algebras with conjunction, disjunction, negation, and the four constants C, T, F, and D. First, it is easy to see that the initial $\mathbb{K}_4$ algebra is the four element distributive lattice (3) with involution and with distinct zeros C and D. (A zero is an element that equals its own negation.)

We apply the following theorem from [16]:

> Any distributive lattice with involution is isomorphic with a subdirect product of isomorphic images of the four element distributive lattice (3) with involution and with two distinct zeros.

From this theorem it follows that the $\mathbb{K}_4$ axioms completely axiomatize the initial $\mathbb{K}_4$ algebra K. Suppose that $\mathrm{K} \models t = u$. Then this identity holds in any subdirect power of K, and since any $\mathbb{K}_4$ algebra is isomorphic to such a subdirect power, we may conclude that $\mathbb{K}_4 \models t = u$. Hence $\mathbb{K}_4 \vdash t = u$ follows by Birkhoff's completeness theorem for equational logic [10]. □

Axioms for conditional composition

We present an alternative axiomatization for our four-valued logic in Table 2, this time taking conditional composition as primitive operation.

Based on the completeness of $\mathbb{K}_4$, we can prove completeness of the $\mathbb{L}_4$ axiomatization by exploiting translations in the following way. If the translation of each $\mathbb{K}_4$ axiom is derivable in $\mathbb{L}_4$, then each $\mathbb{K}_4$ derivation can be mimicked in $\mathbb{L}_4$. Providing an argument on the derivability

Table 2: Axiomatization of $\mathbb{L}_4$.

(L1)	$x \lhd (x' \lhd y \rhd z') \rhd z = (x \lhd x' \rhd z) \lhd y \rhd (x \lhd z' \rhd z)$
(L2)	$(x \lhd y \rhd z) \lhd y' \rhd (x' \lhd y \rhd z') = (x \lhd y' \rhd x') \lhd y \rhd (z \lhd y' \rhd z')$
(L3)	$(x \lhd y \rhd x') \lhd y \rhd z = x \lhd y \rhd (x' \lhd y \rhd z)$
(L4)	$\mathrm{T} \lhd x \rhd \mathrm{F} = x$
(LT)	$x \lhd \mathrm{T} \rhd y = x$
(LF)	$x \lhd \mathrm{F} \rhd y = y$
(LD)	$x \lhd \mathrm{D} \rhd y = \mathrm{D}$
(LC1)	$x \lhd \mathrm{C} \rhd y = y \lhd \mathrm{C} \rhd x$
(LC2)	$x \lhd \mathrm{C} \rhd \mathrm{D} = x$
(LC3)	$\mathrm{C} \lhd \mathrm{C} \rhd x = \mathrm{C}$

of the invariance of the translations then completes the proof. We explain this in some more detail: for t a term in the $\mathbb{L}_4$ signature, we write t' for its translation to $\mathbb{K}_4$ (cf. identity (7)), and for t a term in the $\mathbb{K}_4$ signature, we write t^* for its translation to $\mathbb{L}_4$ (cf. (4), (5) and (6)). Now assume $\mathbb{L}_4 \models u = v$. Then, by translation and the completeness of $\mathbb{K}_4$ we have $\mathbb{K}_4 \vdash u' = v'$. So, $\mathbb{L}_4 \vdash (u')^* = (v')^*$. Finally, invariance of our back and forth translation, i.e., $\mathbb{L}_4 \vdash t = (t')^*$, yields $\mathbb{L}_4 \vdash u = v$, as was to be shown. Section 4 is devoted to our detailed (and somewhat long) proof of the completeness of our $\mathbb{L}_4$ axiomatization.

3 Generalized process algebra

In this section we first introduce a generalization of a simple process algebra system. The system BPA_δ (Basic Process Algebra with deadlock) has two binary operations: alternative composition, or *choice*, and sequential composition. Furthermore, BPA_δ comprises the constant δ that represents deadlock (or inaction). Both alternative composition and deadlock can be seen as special instances of *process algebraic* conditional composition. We provide an operational semantics and a complete set of axioms for our generalization of BPA_δ that comprises conditional composition. Then, we turn to ACP (i.e., the supersystem of BPA_δ that includes operations for parallelism), and discuss a generalization of the remaining operations as well. We extend the operational semantics to this setting, and provide a complete set of axioms. Finally, we use the generalized ACP operations to provide an example on the scheduling of parallel components.

3.1 Generalized BPA with deadlock

Basic Process Algebra with deadlock (BPA_δ) has two binary operations: alternative composition ($+$) and sequential composition ($\cdot$). We parameterize the alternative composition with $\mathbb{L}_4$ terms ϕ, hence obtaining the binary operation $+_\phi$, that we call *conditional composition*[6].

[6]Recall that the $\mathbb{L}_4$ operation $\lhd\ \rhd$ was called conditional composition as well. We now have both a process algebraic and a logical conditional composition. We reserve the notation $\lhd\ \rhd$ for $\mathbb{L}_4$.

Table 3: The $g\mathrm{BPA}_\delta$ axioms.

(G1)	$x +_{\phi\triangleleft\psi\triangleright\chi} y = (x +_\phi y) +_\psi (x +_\chi y)$
(G2)	$(x + y) +_\phi (x' + y') = (x +_\phi x') + (y +_\phi y')$
(G3)	$x +_\phi (y +_\phi z) = (x +_\phi y) +_\phi z$
(G4)	$(x +_\phi y)z = xz +_\phi yz$
(G5)	$(xy)z = x(yz)$
(GT)	$x +_{\mathrm{T}} y = x$
(GF)	$x +_{\mathrm{F}} y = y$
(GD)	$x +_{\mathrm{D}} y = \delta$

Alternative composition can now be seen as the instance $+_{\mathrm{C}}$ of conditional composition, while δ corresponds to $+_{\mathrm{D}}$. We write $g\mathrm{BPA}_\delta$ for this generalization of BPA_δ.

Assume as given a nonempty finite set A of action symbols; then the $g\mathrm{BPA}_\delta$ terms are generated by the grammar

$$p ::= a \mid \delta \mid x \mid p +_\phi p \mid p \cdot p,$$

where a ranges over A, x ranges over process variables, and ϕ ranges over $\mathbb{L}_4$ terms. To avoid confusion with process terms, we shall use the letters ϕ, ψ, χ both for $\mathbb{L}_4$ terms and for $\mathbb{L}_4$ variables (recall that in the previous sections we used x, y, z for proposition variables and t, u for $\mathbb{L}_4$ terms). We may write $+$ for $+_{\mathrm{C}}$ and we omit the symbol $\cdot$ from expressions. We let sequential composition bind stronger than conditional composition.

The axiom system $g\mathrm{BPA}_\delta$ consists of the axioms in Table 3. As proof system we use two-sorted equational logic in the following way:

$$\mathbb{L}_4 \vdash \phi = \psi \;\text{ implies }\; g\mathrm{BPA}_\delta \vdash x +_\phi y = x +_\psi y,$$

where x, y are process variables. For example, one can derive that

$$x +_{\phi\triangleleft\phi\triangleright\phi} y = x +_\phi y.$$

(See Example 4.1 for a derivation of $\phi \triangleleft \phi \triangleright \phi = \phi$ in $\mathbb{L}_4$.)

Alternative composition

We claimed that alternative composition can be seen as the instance $+_{\mathrm{C}}$ of conditional composition. We support this claim by showing that the axioms of BPA_δ are derivable in $g\mathrm{BPA}_\delta$.

Commutativity of alternative composition (axiom A1) is derived by

$$x +_{\mathrm{C}} y \overset{(\mathrm{GT},\mathrm{GF})}{=} (y +_{\mathrm{F}} x) +_{\mathrm{C}} (y +_{\mathrm{T}} x) \overset{(\mathrm{G1})}{=} y +_{\mathrm{F}\triangleleft\mathrm{C}\triangleright\mathrm{T}} x \overset{(\mathrm{LC1},\mathrm{L4})}{=} y +_{\mathrm{C}} x.$$

Associativity of alternative composition (axiom A2) is an instance of axiom G3. Idempotency of alternative composition (axiom A3) can be derived by

$$x +_{\mathrm{C}} x \overset{(\mathrm{GT})}{=} (x +_{\mathrm{T}} y) +_{\mathrm{C}} (x +_{\mathrm{T}} y) \overset{(\mathrm{G1})}{=} x +_{\mathrm{T}\triangleleft\mathrm{C}\triangleright\mathrm{T}} y \overset{(15,\mathrm{GT})}{=} x.$$

Right-distributivity of sequential composition over alternative composition (axiom A4) is an instance of axiom G4. Associativity of sequential composition (axiom A5) occurs here as axiom G5. The axiom $x + \delta = x$ (A6) can be derived by

$$x +_{\mathrm{C}} \delta \overset{(\mathrm{G_T},\mathrm{G_D})}{=} (x +_{\mathrm{T}} y) + (x +_{\mathrm{D}} y) \overset{(\mathrm{G1})}{=} x +_{\mathrm{T} \lhd \mathrm{C} \rhd \mathrm{D}} y \overset{(\mathrm{L_C2},\mathrm{G_T})}{=} x.$$

Finally, the axiom $\delta x = \delta$ (A7) can be derived by

$$\delta x \overset{(\mathrm{G_D})}{=} (y +_{\mathrm{D}} z)x \overset{(\mathrm{G4},\mathrm{G_D})}{=} \delta.$$

Guarded command

The *guarded command* construct [12] is defined by

$$\phi :\to x = x +_\phi \delta.$$

It expresses the instruction to execute process x if the condition ϕ holds. We use this construct in the next section because it allows a more elegant normal form representation than is possible with conditional composition. The following identities can be derived straightforwardly.

$$x +_\phi y = \phi :\to x + \neg\phi :\to y \tag{8}$$
$$\phi :\to (x + y) = \phi :\to x + \phi :\to y \tag{9}$$
$$(\phi :\to x)y = \phi :\to xy \tag{10}$$

Clearly, the following equations are derivable:

$$\mathrm{C} :\to x = \mathrm{T} :\to x = x; \quad \mathrm{F} :\to x = \mathrm{D} :\to x = \delta. \tag{11}$$

We see that, *as a guard*, the truth values C and T have the same behaviour, and so do F and D. Consequently, the guarded command has nicer distribution properties over the logical operations than conditional composition:

$$\phi \vee \psi :\to x = \phi :\to x + \psi :\to x \tag{12}$$
$$\phi \wedge \psi :\to x = \phi :\to (\psi :\to x) \tag{13}$$
$$\phi \lhd \psi \rhd \chi :\to x = \psi \wedge \phi :\to x + \neg\psi \wedge \chi :\to x. \tag{14}$$

These identities can all be derived straightforwardly.

3.2 Operational semantics

Let a set A of action symbols be given. We write P for the set of *process-closed* $g\mathrm{BPA}_\delta$ terms, that is, of process terms that do not contain process variables, but that may contain proposition variables. We give an SOS-style semantics for the elements of P.

Let W be the set of valuations for $\mathbb{L}_4$ terms (given some set of proposition variables), ranged over by w. In Table 4, we give transition rules for the relations

$$\_ \xrightarrow{(\_,\_)} \_ \subseteq P \times (A \times W) \times P,$$

Table 4: Transition rules for $g\mathrm{BPA}_\delta$.

$$a \xrightarrow{a,w} \surd \qquad \frac{x \xrightarrow{a,w} \surd}{xy \xrightarrow{a,w} y} \qquad \frac{x \xrightarrow{a,w} x'}{xy \xrightarrow{a,w} x'y}$$

$$\frac{x \xrightarrow{a,w} \surd,\ w(\phi) \in \{\mathrm{C},\mathrm{T}\}}{x +_\phi y \xrightarrow{a,w} \surd} \qquad \frac{x \xrightarrow{a,w} \surd,\ w(\phi) \in \{\mathrm{C},\mathrm{F}\}}{y +_\phi x \xrightarrow{a,w} \surd}$$

$$\frac{x \xrightarrow{a,w} x',\ w(\phi) \in \{\mathrm{C},\mathrm{T}\}}{x +_\phi y \xrightarrow{a,w} x'} \qquad \frac{x \xrightarrow{a,w} x',\ w(\phi) \in \{\mathrm{C},\mathrm{F}\}}{y +_\phi x \xrightarrow{a,w} x'}$$

and

$$\_ \xrightarrow{(\_,\_)} \surd \subseteq P \times (A \times W).$$

The transitions are labelled with an action and a valuation; if

$$p \xrightarrow{a,w} p'$$

then p has the option to execute action a under valuation w, and by this execution p evolves into p'. The symbol $\surd$ is used to indicate succesful termination; for example, it holds that

$$a \xrightarrow{a,w} \surd$$

for every a and w.

Definition 3.1. A binary relation R on P is a *bisimulation* if it is symmetric, and whenever $p\ R\ q$, then for all a and w

1. if $p \xrightarrow{a,w} \surd$, then $q \xrightarrow{a,w} \surd$, and,
2. if $p \xrightarrow{a,w} p'$ for some p', then $q \xrightarrow{a,w} q'$ for some q' with $p'\ R\ q'$.

Process-closed process terms p and q are bisimilar, notation $p \leftrightarroweq q$, if they are related by a bisimulation.

Bisimilar terms have matching action steps for every possible valuation. Hence, we cater for the inclusion of (user-defined) propositions in the logic, the evaluation of which may not be constant throughout the execution of a process. This equivalence may be called *dynamic*, while *static* bisimilarity would be defined as bisimilarity with respect to one, fixed, valuation.

The transition rules are in the *panth-format* (cf. [20]), from which it follows that bisimilarity is a congruence relation. Furthermore, it is straightforward to verify that the axioms in Table 3 are sound. In the following, we prove that these axioms are complete, that is, that it holds for all process-closed process terms p and q that $p \leftrightarroweq q$ if and only if p and q are derivably equal.

More generally, we may write

$$g\mathrm{BPA}_\delta/\leftrightarroweq \models t_1(\vec{x}) = t_2(\vec{x}),$$

if it holds that $t_1(\vec{p}) \leftrightarroweq t_2(\vec{p})$ for all closed instantiations $\vec{p}$ of $\vec{x}$.

3.3 Completeness and correspondence

We write $\sum_{i\in I} p_i$, where I is a finite set of indices, for the alternative composition of the processes p_i with $i \in I$; we define $\sum_{i\in\emptyset} p_i = \delta$.

Definition 3.2. Basic terms are terms of the form

$$\sum_{i\in I} \phi_i :\to p_i,$$

where $p_i \in \{a, a\,q \mid a \in A,\ q \text{ a basic term}\}$ for all $i \in I$.

Lemma 3.1. For all process-closed terms p and basic terms q, the sequential composition pq is derivably equal to a basic term.

Proof. We apply induction on the structure of p. If $p \equiv a \in A$, then $a\,q$ equals the basic term $\top :\to a\,q$ by (11). If $p \equiv \delta$, then pq equals the basic term δ by A7.

If $p \equiv p_1 +_\phi p_2$, then

$$pq \overset{(8,\mathrm{G4},10)}{=} \phi :\to p_1 q + \neg\phi :\to p_2\, q.$$

It follows from the induction hypothesis that there are basic terms

$$p' \equiv \sum_i \psi_i :\to r_i \quad \text{and} \quad p'' \equiv \sum_j \psi_j :\to r_j$$

with $p' = p_1 q$ and $p'' = p_2 q$. Using (9) and (13), we derive that pq equals the basic term

$$\sum_i \phi \wedge \psi_i :\to r_i + \sum_j \neg\phi \wedge \psi_j :\to r_j.$$

If $p \equiv p_1 p_2$, then there are basic terms r', r, with

$$pq \equiv (p_1 p_2)\, q \overset{(\mathrm{G5})}{=} p_1(p_2 q) \overset{(\mathrm{IH})}{=} p_1 r' \overset{(\mathrm{IH})}{=} r.$$

□

Lemma 3.2. Every process-closed process term p is derivably equal to a basic term.

Proof. We apply induction on the structure of p. If $p \equiv \delta$, then p equals an empty summation by definition. If $p \equiv a \in A$, then p equals the basic term $\top :\to a$ by (11).

If $p \equiv p_1 +_\phi p_2$, then by induction hypothesis there are basic terms

$$p_1' \equiv \sum_i \psi_i :\to p_i, \quad p_2' \equiv \sum_j \psi_j :\to p_j,$$

with $p_1 = p_1'$ and $p_2 = p_2'$. By (8), we find that p equals

$$\phi :\to p_1' + \neg\phi :\to p_2'$$

Using (13) and (9) we get that this term equals the basic term

$$\sum_i \phi \wedge \psi_i :\to p_i + \sum_j \neg\phi \wedge \psi_j :\to p_j.$$

If $p \equiv p_1 p_2$, then there are basic terms q, r, with

$$p \equiv p_1 p_2 \overset{(\mathrm{IH})}{=} p_1 q \overset{(3.1)}{=} r.$$

□

We define the *height* of basic terms:

$$\begin{aligned} h(a) &= 1, \\ h(\delta) &= 0, \\ h(\phi :\to p) &= h(p), \\ h(p + q) &= \max(h(p), h(q)), \\ h(ap) &= 1 + h(p). \end{aligned}$$

Lemma 3.3. Every basic term p is derivably equal to a basic term

$$q \equiv \sum_{i \in I} \phi_i :\to q_i,$$

with the following properties:

1. $h(q) \leq h(p)$,
2. for all distinct $i, j \in I$ with $q_i, q_j \in A$ it holds that $q_i \neq q_j$,
3. for all $i \in I$ it holds that $\vdash \phi_i = \phi_i \wedge \mathrm{C}$,
4. for all $i \in I$ it holds that $\nvdash \phi_i = \mathrm{F}$.

Proof. Starting from p written

$$p \equiv \sum_{i} \psi_i :\to p_i,$$

we first join summands $\psi_i :\to p_i$ and $\psi_j :\to p_j$ with $p_i = p_j = a \in A$ to a single summand $\psi_i \vee \psi_j :\to a$ using (12). Observe that this does not change the height of the term, so property 1 is preserved. The resulting term satisfies property 2.

Then, we add a conjunct C to all conditions ψ: we can derive that

$$\psi :\to p_i \overset{(11)}{=} \psi :\to (\mathrm{C} :\to p_i) \overset{(13)}{=} \psi \wedge \mathrm{C} :\to p_i.$$

The resulting term satisfies property 3. Observe that this does not change the height of the term, so property 1 is preserved. Also, property 2 is preserved.

Finally, if the condition of one of the summands in the resulting term is derivably equal to F, then we can omit that summand from the summation. The resulting term satisfies property 4. Observe that properties 1–3 are preserved. □

Theorem 3.4. For all process-closed terms p_1 and p_2, it holds that $p_1 \leftrightarrows p_2$ implies $\vdash p_1 = p_2$.

Proof. Without loss of generality (Lemma 3.2) we assume that p_1 and p_2 are basic terms. We apply induction on $h = \max(h(p_1), h(p_2))$. If $h = 0$, then it must be that $p_1 \equiv p_2 \equiv \delta$.

Let $h > 0$. Using Lemma 3.3, we may assume that for $k = 1, 2$, the term

$$p_k \equiv \sum_{i \in I_k} \phi_{k,i} :\to p_{k,i}$$

satisfies the properties 1–4 from Lemma 3.3. For $k = 1, 2$, we make the following observations.

(a) We may assume that for all distinct $i, j \in I_k$, it holds that $p_{k,i} \not\leftrightarrows p_{k,j}$.

If $p_{k,i}$ and $p_{k,j}$ in A, then this holds by Lemma 3.3.2. Otherwise, let $p_{k,i} \equiv aq$ and $p_{k,j} \equiv ar$ and $q \leftrightarrows r$. By induction hypothesis, we find that $\vdash q = r$. Hence, the summands $\phi_{k,i} :\to p_{k,i}$ and $\phi_{k,j} :\to p_{k,j}$ could have been joined to the single summand $\phi_{k,i} \vee \phi_{k,j} :\to p_{k,i}$ using (12). This does not increase the height of p_k.

(b) We may assume, using idempotency of $+$, that all summands of p_k are unique.

(c) For every $w \in W$ and $i \in I_k$, it holds by Lemma 3.3.3 that either $w(\phi_{k,i}) = \mathrm{C}$ or $w(\phi_{k,i}) = \mathrm{F}$.

(d) For all $i \in I_k$, it holds (using Lemma 3.3.4 and (c)) that $w(\phi_{k,i}) = \mathrm{C}$ for at least one $w \in W$.

We show that each summand in p_k is derivably equal to a unique summand in p_{3-k}. Take an arbitrary $i \in I_k$.

- Case $p_{k,i} \equiv a \in A$. Using Lemma 3.3.2 and (c), we find that $p_k \xrightarrow{a,w} \surd$ iff $w(\phi_{k,i}) = \mathrm{C}$, and, since $p_k \leftrightarrows p_{3-k}$, also $p_{3-k} \xrightarrow{a,w} \surd$ iff $w(\phi_{k,i}) = \mathrm{C}$. Using (d), we find that for some unique $j \in I_{3-k}$ it holds that $p_{3-k,j} \equiv a$. It follows that $w(\phi_{k,i}) = \mathrm{C}$ iff $w(\phi_{3-k,j}) = \mathrm{C}$, and so by (c), it holds that $\models \phi_{k,i} = \phi_{3-k,j}$ and hence $\vdash \phi_{k,i} = \phi_{3-k,j}$.

- Case $p_{k,i} \equiv aq$. Using (c), we find that $p_k \xrightarrow{a,w} q$ iff $w(\phi_{k,i}) = \mathrm{C}$. Then, since $p_k \leftrightarrows p_{3-k}$, also $p_{3-k} \xrightarrow{a,w} r$ for some r with $q \leftrightarrows r$ iff $w(\phi_{k,i}) = \mathrm{C}$. By (d), we find that for some unique (using (a)) $j \in I_{3-k}$ it holds that $p_{3-k,j} \equiv ar$. It follows that $w(\phi_{k,i}) = \mathrm{C}$ iff $w(\phi_{3-k,j}) = \mathrm{C}$, and so by (c), it holds that $\models \phi_{k,i} = \phi_{3-k,j}$ and hence $\vdash \phi_{k,i} = \phi_{3-k,j}$.

  Finally, it holds that $\vdash p_{k,i} = p_{3-k,j}$, since it follows by IH from $q \leftrightarrows r$ that $\vdash q = r$.

□

We end this section with some reflections on correspondence between gBPA_δ and $\mathbb{L}_4$. Clearly, process algebraic conditional composition and its logical counterpart are quite similar, as becomes apparent when comparing the axioms G1–G3 with the axioms L1–L3, and GT, GF, and GD with LT, LF, and LD, respectively. The attentive reader may wonder whether the generalization of G2 to L2's counterpart

$$(x +_\psi y) +_\phi (x' +_\psi y') = (x +_\phi x') +_\psi (y +_\phi y')$$

is sound. This is indeed the case and, more generally, the following correspondence result holds:

Proposition 3.5. Let $t_1(\vec{x},\vec{\varphi}) = t_2(\vec{x},\vec{\varphi})$ be a process identity with process variables $\vec{x}$ and condition variables $\vec{\varphi}$ in which the only constants are in T_4 and the only operation is $x +_\phi y$, written as $x \lhd \phi \rhd y$. Then

$$g\mathrm{BPA}_\delta/\leftrightarrows \models t_1(\vec{x},\vec{\varphi}) = t_2(\vec{x},\vec{\varphi})$$

if and only if

$$\mathbb{L}_4 \models t_1(\vec{x},\vec{\varphi}) = t_2(\vec{x},\vec{\varphi}),$$

where in the latter statement, $\vec{x}$ also represents condition variables.

Finally, this result implies that $\mathbb{L}_4$ (and thus also $\mathbb{K}_4$) characterizes the axiom $x + \delta = x$ of BPA_δ (by axiom Lc2), and thus the interplay between choice and deadlock as discussed in Introduction, from a logical perpsective.

3.4 Generalized ACP

We extend generalized BPA_δ to a generalized version of ACP (Algebra of Communicating Processes, see, e.g., [4, 1, 14]). The parameterized parallel composition $x \,{}_\phi||_\psi\, y$ denotes the parallel execution of x and y under conditions ϕ and ψ. Here, the condition ϕ covers the choice between interleaving and synchronization, and ψ determines the order of interleaving and synchronization. For example, the parallel composition operation $||$ of ACP equals ${}_{\mathrm{C}}||_{\mathrm{C}}$.

The auxiliary operations of generalized ACP are the following:

Parameterized left merge: $x \,{}_\phi|\!\!\lfloor_\psi\, y$ denotes $x \,{}_\phi||_\psi\, y$ with the restriction that the first action stems from x.

Parameterized communication merge: $x \,{}_\phi|_\psi\, y$ denotes $x \,{}_\phi||_\psi\, y$ with the restriction that the first action is a synchronization of both x and y.

Parameterized left communication merge: $x \,{}_\phi\lfloor_\psi\, y$ is used to define $x \,{}_\phi|_\psi\, y$.

Encapsulation: $\partial_H(x)$ (where $H \subseteq A$) renames atoms in H to δ and distributes over conditional composition and sequential composition.

In ACP, the commutative and associative communication function $| : A \times A \to A \cup \{\delta\}$ is given (and extended to process terms). The axioms of our generalization of ACP are those of $g\mathrm{BPA}_\delta$ and those in Table 5. We adopt the convention that $+_\phi$ binds weakest and $\cdot$ binds strongest, and denote the resulting system by gACP, or by $g\mathrm{ACP}(A,|)$ if we want to make the parameters of the theory explicit.

Observe that the operation ${}_{\mathrm{T}}||_{\mathrm{C}}$ restricts parallel composition to interleaving only, that is, to the so-called free merge, while ${}_{\mathrm{F}}||_\diamond$ for $\diamond \in \{\mathrm{C},\mathrm{T},\mathrm{F}\}$ defines "synchronous ACP" and ${}_{\mathrm{T}}||_{\mathrm{T}}$ represents sequential composition.

Like in ACP, the parallel composition operations can be eliminated from terms. As an example, we give the terms resulting from the elimination of the parameterized parallel composition in $a \,{}_\phi||_\psi\, b$ in Figure 1.

Some typical gACP identities are:

$$x \,{}_\phi||_\psi\, y = y \,{}_\phi||_{\neg\psi}\, x,$$
$$x \,{}_\phi|_\psi\, y = y \,{}_\phi|_{\neg\psi}\, x,$$
$$\delta \,{}_\phi|_\psi\, x = \delta.$$

Table 5: Additional axioms of $g\mathrm{ACP}(A,|)$; $a,b,c \in A$ and $H \subseteq A$.

(C1)	$a \mid b = b \mid a$	
(C2)	$(a \mid b) \mid c = a \mid (b \mid c)$	
(GD1)	$\partial_H(a) = a$	if $a \notin H$
(GD2)	$\partial_H(a) = \delta$	if $a \in H$
(GD3)	$\partial_H(x +_\phi y) = \partial_H(x) +_\phi \partial_H(y)$	
(GD4)	$\partial_H(xy) = \partial_H(x)\partial_H(y)$	
(GM1)	$x \,{}_\phi\|\|_\psi\, y = (x \,{}_\phi\lfloor\!\lfloor_\psi\, y +_\psi y \,{}_\phi\lfloor\!\lfloor_{\neg\psi}\, x) +_\phi x \,{}_\phi\lfloor_\psi\, y$	
(GM2)	$a \,{}_\phi\lfloor\!\lfloor_\psi\, x = ax$	
(GM3)	$ax \,{}_\phi\lfloor\!\lfloor_\psi\, y = a(x \,{}_\phi\|\|_\psi\, y)$	
(GM4)	$(x +_\phi y) \,{}_\psi\lfloor\!\lfloor_\chi\, z = x \,{}_\psi\lfloor\!\lfloor_\chi\, z +_\phi y \,{}_\psi\lfloor\!\lfloor_\chi\, z$	
(GM5)	$x \,{}_\phi\lfloor_\psi\, y = x \,{}_\phi\lfloor_\psi\, y +_\psi y \,{}_\phi\lfloor_{\neg\psi}\, x$	
(GM6)	$ax \,{}_\phi\lfloor_\psi\, y = a \,{}_\phi\lfloor_\psi (y \,{}_\phi\lfloor\!\lfloor_{\neg\psi}\, x)$	
(GM7)	$a \,{}_\phi\lfloor_\psi\, b = a \mid b$	
(GM8)	$a \,{}_\phi\lfloor_\psi\, bx = (a \mid b)x$	
(GM9)	$a \,{}_\phi\lfloor_\psi (x +_\chi y) = a \,{}_\phi\lfloor_\psi\, x +_\chi a \,{}_\phi\lfloor_\psi\, y$	
(GM10)	$(x +_\phi y) \,{}_\psi\lfloor_\chi\, z = x \,{}_\psi\lfloor_\chi\, z +_\phi y \,{}_\psi\lfloor_\chi\, z$	

3.5 Operational semantics and completeness

Write P for the set of process-closed process terms. In Table 6 we give additional rules for the transition rules defined in Table 4. We stick to bisimulation equivalence as defined in Definition 3.1, and as before it follows that bisimilarity is a congruence for all operations involved.

Notation. In these rules, we let $x'/\surd$ and $y'/\surd$ range over $P \cup \{\surd\}$ (we stress that the symbol $\surd$ is not a process term). In order to keep the presentation of the rules short, we use the notational convention

$$x \,{}_\phi\|\|_\psi \surd \equiv \surd \,{}_\phi\|\|_\psi\, x \equiv x, \quad \text{and} \quad \surd \,{}_\phi\|\|_\psi \surd \equiv \partial_H(\surd) \equiv \surd.$$

It is not difficult (but tedious) to establish that in the bisimulation model thus obtained all equations of Table 5 are true. Furthermore, each process-closed process term over gACP

$a \,{}_\phi\|\|_\psi\, b$	C	T	F	D	ψ
C	$ab + ba + c$	$ab + c$	$ba + c$	δ	
T	$ab + ba$	ab	ba	δ	
F	c	c	c	δ	
D	δ	δ	δ	δ	
ϕ					

Figure 1: Example; let $a \mid b = c$.

Table 6: Additional transition rules for $g\mathrm{ACP}(A,|)$ in *panth*-format.

$$\frac{x \xrightarrow{a,w} x'/\surd,\ w(\phi) \in \{\mathrm{C},\mathrm{T}\},\ w(\psi) \in \{\mathrm{C},\mathrm{T}\}}{x \,{}_{\phi}||_{\psi}\, y \xrightarrow{a,w} (x'/\surd) \,{}_{\phi}||_{\psi}\, y}$$

$$\frac{x \xrightarrow{a,w} x'/\surd,\ w(\phi) \in \{\mathrm{C},\mathrm{T}\},\ w(\psi) \in \{\mathrm{C},\mathrm{F}\}}{y \,{}_{\phi}||_{\psi}\, x \xrightarrow{a,w} y \,{}_{\phi}||_{\psi} (x'/\surd)}$$

$$\frac{x \xrightarrow{a,w} x'/\surd,\ y \xrightarrow{b,w} y'/\surd,\ a \mid b = c,\ w(\phi)\in\{\mathrm{C},\mathrm{F}\},\ w(\psi)\in\{\mathrm{C},\mathrm{T},\mathrm{F}\}}{x \,{}_{\phi}||_{\psi}\, y \xrightarrow{c,w} (x'/\surd) \,{}_{\phi}||_{\psi} (y'/\surd)}$$

$$\frac{x \xrightarrow{a,w} x'/\surd,\ y \xrightarrow{b,w} y'/\surd,\ a \mid b = c,\ w(\psi) \in \{\mathrm{C},\mathrm{T},\mathrm{F}\}}{x \,{}_{\phi}|_{\psi}\, y \xrightarrow{c,w} (x'/\surd) \,{}_{\phi}||_{\psi} (y'/\surd)}$$

$$\frac{x \xrightarrow{a,w} x'/\surd,\ y \xrightarrow{b,w} y'/\surd,\ a \mid b = c}{x \,{}_{\phi}\lfloor_{\psi}\, y \xrightarrow{c,w} (x'/\surd) \,{}_{\phi}||_{\psi} (y'/\surd)}$$

$$\frac{x \xrightarrow{a,w} x'/\surd}{x \,{}_{\phi}\!\perp\!\!\!\perp_{\psi}\, y \xrightarrow{a,w} (x'/\surd) \,{}_{\phi}||_{\psi}\, y} \qquad \frac{x \xrightarrow{a,w} x'/\surd,\ a \notin H}{\partial_H(x) \xrightarrow{a,w} \partial_H(x'/\surd)}$$

is provably equal to, and thus bisimilar with, a generalized basic term (see Definition 3.2). Hence:

Theorem 3.6. The system $g\mathrm{ACP}$ is complete with respect to bisimulation equivalence.

3.6 Example: The minimal history operator

In the following we provide an example in which the generalized operations are used.[7] The *minimal history* operator H_0 keeps track of the number of actions that a process has performed since initialization and increases stepwise its index. The knowledge of the history of a process is minimal in the sense that we only count the actions that are performed. For example, we find that

$$H_0(abc) \xrightarrow{a} H_1(bc) \xrightarrow{b} H_2(c) \xrightarrow{c} \surd.$$

In this section, we shall use the history of processes in the condition parameters of the operations of $g\mathrm{ACP}$; hence, we shall be able to program a scheduling mechanism for parallel processes.

Let In be the assertion which is true of the initial state of a process and false thereafter. Furthermore, let $\mathsf{P}(\phi)$ be the assertion that ϕ is valid in all the previous states (i.e., the states before the last action); if there is no such state, then $\mathsf{P}(\phi) = \mathrm{D}$.

[7] This example is based on a similar one in [6].

Though P is a modality, we have

$$\begin{aligned}\mathsf{P}(\mathrm{T}) &= \neg\mathsf{In} \vee \mathrm{D},\\ \mathsf{P}(\neg\phi) &= \neg\mathsf{P}(\phi),\\ \mathsf{P}(\phi \wedge \psi) &= \mathsf{P}(\phi) \wedge \mathsf{P}(\psi),\end{aligned}$$

and one can set

$$\begin{aligned}\mathsf{P}(\mathrm{C}) &= \mathrm{D} \triangleleft \mathsf{In} \triangleright \mathrm{C},\\ \mathsf{P}(\mathrm{D}) &= \mathrm{D}.\end{aligned}$$

It then follows that P can be removed from finite expressions except for atoms of the form $\mathsf{P}^n(\mathsf{In})$ for $n \in \mathbb{N}$.

The minimal history operator H_n is, for $n \in \mathbb{N}$, defined by

$$\begin{aligned}H_n(a) &= a \text{ for } a \in A \cup \{\delta\},\\ H_n(ax) &= a \cdot H_{n+1}(x) \text{ for } a \in A,\\ H_n(x +_\phi y) &= H_n(x) +_{H_n(\phi)} H_n(y),\end{aligned}$$

and on conditions (as occurring in the last line above), by

$$\begin{aligned}H_n(c) &= c \text{ for } c \in T_4,\\ H_n(\mathsf{In}) &= \begin{cases}\mathrm{T} & \text{if } n = 0,\\ \mathrm{F} & \text{otherwise,}\end{cases}\\ H_0(\mathsf{P}(\phi)) &= \mathrm{D},\\ H_{n+1}(\mathsf{P}(\phi)) &= H_n(\phi),\\ H_n(\neg\phi) &= \neg H_n(\phi),\\ H_n(\phi \wedge \psi) &= H_n(\phi) \wedge H_n(\psi).\end{aligned}$$

As an example, consider

$$\begin{aligned}\Phi = \mathsf{In} \vee\\ (\neg\mathsf{P}(\mathsf{In}) \wedge \mathsf{P}^2(\mathsf{In})) \vee\\ (\neg\mathsf{P}(\mathsf{In}) \wedge \neg\mathsf{P}^2(\mathsf{In}) \wedge \neg\mathsf{P}^3(\mathsf{In}) \wedge \mathsf{P}^4(\mathsf{In})).\end{aligned}$$

The assertion Φ is true in states where the action history length is 0, 2, or 4, and false otherwise. We assume that all communications are δ. Now consider $H_0(P_1 \,||\, P_2)$, where

$$\begin{aligned}P_1 &= (\Phi :\to a)(\Phi :\to a)(\Phi :\to b),\\ P_2 &= (\neg\Phi :\to c)(\neg\Phi :\to d).\end{aligned}$$

It follows that $H_0(P_1 \,||\, P_2) = acadb$. The history operator in cooperation with Φ schedules $P_1 \,||\, P_2$ as an alternation of steps, beginning with P_1.

In process algebra, one often considers potentially nonterminating processes that can be specified with $*$, the binary Kleene star [18], defined by

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

(See also [2].) In particular, $x^*\delta$ repeatedly performs x, as follows easily from the axioms. An obvious question is how to provide scheduling guards for potentially nonterminating processes. This leads us to infinitary propositions, which can be defined by recursion. As an example, let

$$\Phi_{even} = \mathsf{In} \vee \neg\mathsf{P}(\Phi_{even}).$$

Thus Φ_{even} will hold for even step numbers, and it easily follows that

$$H_0((\Phi_{even} :\rightarrow a)^*\delta \,||\, (\neg\Phi_{even} :\rightarrow b)^*\delta) = (ab)^*\delta.$$

For another example, let

$$\Psi = \mathsf{In} \vee (\neg\mathsf{P}(\mathsf{In}) \wedge \neg\mathsf{P}^2(\mathsf{In}) \wedge \mathsf{P}^3(\Psi)).$$

So Ψ holds if the action history length is a multiple of 3. In order to give a somewhat more real-life example on scheduling, we consider Υ, the "negation" of Ψ, and Φ, which holds if the action history length modulo 3 is either 0 or 2. These infinitary propositions can be recursively defined by

$$\Upsilon = \neg\mathsf{In} \wedge (\mathsf{P}(\mathsf{In}) \vee \mathsf{P}^2(\mathsf{In}) \vee \mathsf{P}^3(\Upsilon)),$$
$$\Phi = \mathsf{In} \vee \mathsf{P}(\Upsilon).$$

Now consider the processes

$$S = \Big(\sum_{d\in D} r_1(d)\cdot s_2(d)\Big)^*\delta,$$
$$R = \Big(\sum_{d\in D} r_2(d)\cdot s_3(d)\Big)^*\delta.$$

The idea is that sender S receives a datum from some finite domain along channel 1 from the environment and then sends this datum via channel 2, while receiver R receives data along channel 2 and propagates these along channel 3. Now, using Φ and Ψ, the parallel composition of S and R can be scheduled in such a way that only communications (data transmissions) can occur along channel 2: it is not hard to show that for $k \in \mathbb{N}$,

$$H_{3k}(S\,{}_\Phi||_\Psi\, R) = \Big(\sum_{d\in D} r_1(d)\cdot(r_2(d) \mid s_2(d))\cdot s_3(d)\Big)\cdot H_{3k+3}(S\,{}_\Phi||_\Psi\, R).$$

So, for naturals k, and in particular for $k = 0$, we find that $H_{3k}(S\,{}_\Phi||_\Psi\, R)$ describes the intended scheduling.

4 Completeness of the $\mathbb{L}_4$ axioms

In this section we give a full proof of the completeness of the $\mathbb{L}_4$ axioms. First, we establish a normal form representation, which is then used to derive the translations of the $\mathbb{K}_4$ axioms in $\mathbb{L}_4$. Finally we argue that translating a term from $\mathbb{L}_4$ to $\mathbb{K}_4$, and translating back the result yields a provably equal term, which completes our proof.

Table 7: Axiomatization of $\mathbb{L}_4$.

(L1)	$x \lhd (x' \lhd y \rhd z') \rhd z = (x \lhd x' \rhd z) \lhd y \rhd (x \lhd z' \rhd z)$
(L2)	$(x \lhd y \rhd z) \lhd y' \rhd (x' \lhd y \rhd z') = (x \lhd y' \rhd x') \lhd y \rhd (z \lhd y' \rhd z')$
(L3)	$(x \lhd y \rhd x') \lhd y \rhd z = x \lhd y \rhd (x' \lhd y \rhd z)$
(L4)	$\mathrm{T} \lhd x \rhd \mathrm{F} = x$
(LT)	$x \lhd \mathrm{T} \rhd y = x$
(LF)	$x \lhd \mathrm{F} \rhd y = y$
(LD)	$x \lhd \mathrm{D} \rhd y = \mathrm{D}$
(Lc1)	$x \lhd \mathrm{C} \rhd y = y \lhd \mathrm{C} \rhd x$
(Lc2)	$x \lhd \mathrm{C} \rhd \mathrm{D} = x$
(Lc3)	$\mathrm{C} \lhd \mathrm{C} \rhd x = \mathrm{C}$

4.1 Axioms and normal forms

In Table 7 we recall the axiomatization for $\mathbb{L}_4$ given earlier in Table 2.

We shall freely use the fact that the binary operation $\sqcup$ (the abbreviation of $\lhd \mathrm{C} \rhd$) is idempotent (identity (15) below), commutative (axiom Lc1), and associative (axiom L3).

Lemma 4.1. The following identities are derivable, where (16)–(18) refer to (2).

$$x \sqcup x = x \tag{15}$$
$$(x \sqcup x') \lhd y \rhd z = x \lhd y \rhd z \sqcup x' \lhd y \rhd z \tag{16}$$
$$x \lhd (y \sqcup y') \rhd z = x \lhd y \rhd z \sqcup x \lhd y' \rhd z \tag{17}$$
$$x \lhd y \rhd (z \sqcup z') = x \lhd y \rhd z \sqcup x \lhd y \rhd z' \tag{18}$$

Proof. Identity (15) is derived by

$$x \lhd \mathrm{C} \rhd x \overset{(\mathrm{Lc2})}{=} (x \lhd \mathrm{C} \rhd \mathrm{D}) \lhd \mathrm{C} \rhd (x \lhd \mathrm{C} \rhd \mathrm{D}) \overset{(\mathrm{L1,Lc3,Lc2})}{=} x.$$

Identities (16) and (18) are derived using (15) and axiom L2. Identity (17) is an instance of axiom L1. □

The identities in the following lemma are used in the section below, where we introduce normal forms for $\mathbb{L}_4$.

Lemma 4.2. The following identities are derivable.

$$\mathrm{D} \lhd x \rhd \mathrm{D} = \mathrm{D} \tag{19}$$
$$x \lhd y \rhd z = x \lhd y \rhd \mathrm{D} \sqcup \mathrm{D} \lhd y \rhd z \tag{20}$$
$$x \lhd y \rhd z = z \lhd (\mathrm{F} \lhd y \rhd \mathrm{T}) \rhd x \tag{21}$$
$$(x \lhd z \rhd \mathrm{D}) \lhd y \rhd \mathrm{D} = (x \lhd y \rhd \mathrm{D}) \lhd z \rhd \mathrm{D} \tag{22}$$
$$(y \lhd x \rhd \mathrm{D}) \lhd x \rhd \mathrm{D} = y \lhd x \rhd \mathrm{D} \tag{23}$$

Proof. Identity (19) is derived by

$$\mathrm{D} \lhd x \rhd \mathrm{D} \overset{(\mathrm{LD})}{=} (y \lhd \mathrm{D} \rhd y) \lhd x \rhd (y \lhd \mathrm{D} \rhd y) \overset{(\mathrm{L2,LD})}{=} \mathrm{D}.$$

Identity (20) is derived by

$$x \lhd y \rhd z \overset{(\mathrm{Lc1,Lc2})}{=} (x \sqcup \mathrm{D}) \lhd y \rhd (\mathrm{D} \sqcup z) \overset{(\mathrm{L2})}{=} \text{rhs}.$$

Identity (21) is derived using the axioms L1, LT, and LF. Identity (22) is derived by

$$\text{lhs} \overset{(19)}{=} (x \lhd z \rhd \mathrm{D}) \lhd y \rhd (\mathrm{D} \lhd z \rhd \mathrm{D}) \overset{(\mathrm{L2,19})}{=} \text{rhs}.$$

Finally, identity (23) is derived using axiom L3 and (19). □

4.2 Normal forms

We define *simple normal forms* as follows: the truth values T and F are simple normal forms; if t is a simple normal form, then $t \lhd u \rhd \mathrm{D}$ is a simple normal form for any term u.

A *normal form* is a least upper bound

$$\bigsqcup_{i \in I} t_i$$

of simple normal forms t_i, where I is a finite set of indices; we define $\bigsqcup_{i \in \emptyset} t_i = \mathrm{D}$.

Every simple normal form is of the form

$$((\cdots (a \lhd u_n \rhd \mathrm{D}) \cdots) \lhd u_2 \rhd \mathrm{D}) \lhd u_1 \rhd \mathrm{D},$$

where $a \in \{\mathrm{T}, \mathrm{F}\}$, for some $n \geq 0$. We call the u_i the *guards* of the simple normal form. Using identities (22) and (23), we see that the order of the guards can be changed, and that double occurrences of the same guard can be identified. Hence, we shall write these simple normals forms with the set of guards notation

$$\{u_1, \ldots, u_n\}a.$$

Proposition 4.3. It holds that

$$\{u_1, \ldots, u_n\}a = a \lhd (u_1 \wedge \cdots \wedge u_n) \rhd \mathrm{D},$$

for all terms $u_1, \ldots, u_n$ and $a \in \{\mathrm{T}, \mathrm{F}\}$.

A normal form consists of a T-*part* and a F-*part*: it can be written as

$$\bigsqcup_i \alpha_i \mathrm{T} \sqcup \bigsqcup_j \alpha_j \mathrm{F},$$

where the α_i, α_j are finite sets of terms. As an example, we give a normal form for the variable x:

$$x \overset{(\mathrm{L4})}{=} \mathrm{T} \lhd x \rhd \mathrm{F} \overset{(20)}{=} \mathrm{T} \lhd x \rhd \mathrm{D} \sqcup \mathrm{D} \lhd x \rhd \mathrm{F} \overset{(21)}{=} \{x\}\mathrm{T} \sqcup \{\mathrm{F} \lhd x \rhd \mathrm{T}\}\mathrm{F}, \tag{24}$$

where the right-hand side is a normal form. The following theorem is a consequence of (24):

Theorem 4.4. Every $\mathbb{L}_4$ term is derivably equal to a normal form.

A simple normal form is *optimal*, if all its guards are either variables or negated variables: a simple normal form $\alpha\mathrm{T}$ or $\alpha\mathrm{F}$ is optimal if every element of α is either a variable or of the form $\mathrm{F} \lhd x \rhd \mathrm{T}$ for some variable x. We shall further abbreviate $\mathrm{F} \lhd x \rhd \mathrm{T}$ by $\bar{x}$. It is not difficult to prove that every term is derivably equal to an optimal normal form, that is, to a least upper bound of optimal simple normal forms.

Finding optimal normal forms is a straightforward procedure, as is illustrated by (24) and

$$\begin{aligned} x \lhd y \rhd z &\overset{(20,21)}{=} x \lhd y \rhd \mathrm{D} \sqcup z \lhd \bar{y} \rhd \mathrm{D} \\ &\overset{(24)}{=} (\{x\}\mathrm{T} \sqcup \{\bar{x}\}\mathrm{F}) \lhd y \rhd \mathrm{D} \sqcup (\{z\}\mathrm{T} \sqcup \{\bar{z}\}\mathrm{F}) \lhd \bar{y} \rhd \mathrm{D} \\ &\overset{(16)}{=} \{y, x\}\mathrm{T} \sqcup \{y, \bar{x}\}\mathrm{F} \sqcup \{\bar{y}, z\}\mathrm{T} \sqcup \{\bar{y}, \bar{z}\}\mathrm{F}. \end{aligned} \tag{25}$$

Lemma 4.5. Let α and β be finite sets of terms. We can derive the following identities.

$$x \lhd \alpha\mathrm{T} \rhd y = x \lhd \alpha\mathrm{T} \rhd \mathrm{D} \tag{26}$$
$$x \lhd \alpha\mathrm{F} \rhd y = \mathrm{D} \lhd \alpha\mathrm{F} \rhd y \tag{27}$$
$$\beta\mathrm{T} \lhd \alpha\mathrm{T} \rhd \mathrm{D} = (\alpha \cup \beta)\mathrm{T} \tag{28}$$
$$\beta\mathrm{F} \lhd \alpha\mathrm{T} \rhd \mathrm{D} = (\alpha \cup \beta)\mathrm{F} \tag{29}$$
$$\mathrm{D} \lhd \alpha\mathrm{F} \rhd x = x \lhd \alpha\mathrm{T} \rhd \mathrm{D} \tag{30}$$

Proof. We prove (26) using induction on $|\alpha|$. If $\alpha = \emptyset$, then $\alpha\mathrm{T} = \mathrm{T}$ and the identity follows from axiom $\mathrm{L}\mathrm{T}$.

If $\alpha = \alpha' \cup \{u\}$, for some $u \notin \alpha'$, then

$$\begin{aligned} x \lhd \alpha\mathrm{T} \rhd y &= x \lhd (\alpha'\mathrm{T} \lhd u \rhd \mathrm{D}) \rhd y \\ &\overset{(\mathrm{L1},\mathrm{L}\mathrm{D})}{=} (x \lhd \alpha'\mathrm{T} \rhd y) \lhd u \rhd \mathrm{D} \\ &\overset{(\mathrm{IH})}{=} (x \lhd \alpha'\mathrm{T} \rhd \mathrm{D}) \lhd u \rhd \mathrm{D} = x \lhd \alpha\mathrm{T} \rhd \mathrm{D}. \end{aligned}$$

We prove (28) using induction on $|\alpha|$. If $\alpha = \emptyset$, then $\alpha\mathrm{T} = \mathrm{T}$ and the identity follows from axiom $\mathrm{L}\mathrm{T}$.

If $\alpha = \alpha' \cup \{u\}$, for some $u \notin \alpha'$, then

$$\begin{aligned} \beta\mathrm{T} \lhd \alpha\mathrm{T} \rhd \mathrm{D} &= \beta\mathrm{T} \lhd (\alpha'\mathrm{T} \lhd u \rhd \mathrm{D}) \rhd \mathrm{D} \\ &\overset{(\mathrm{L1},\mathrm{L}\mathrm{D})}{=} (\beta\mathrm{T} \lhd \alpha'\mathrm{T} \rhd \mathrm{D}) \lhd u \rhd \mathrm{D} \\ &\overset{(\mathrm{IH})}{=} (\alpha' \cup \beta)\mathrm{T} \lhd u \rhd \mathrm{D} = (\alpha \cup \beta)\mathrm{T}. \end{aligned}$$

The proofs of the other identities are similar. □

Lemma 4.6 (Absorption). If α and β are finite sets of terms, and $\alpha \subseteq \beta$, then we can derive

$$\alpha\mathrm{T} \sqcup \beta\mathrm{T} = \alpha\mathrm{T} \quad \text{and} \quad \alpha\mathrm{F} \sqcup \beta\mathrm{F} = \alpha\mathrm{F}. \tag{Abs}$$

Proof. We derive

$$\begin{aligned}\alpha\mathrm{T} \sqcup (\alpha \cup \beta)\mathrm{T} &\overset{(\mathrm{Lc2,28})}{=} \alpha\mathrm{T} \lhd \mathrm{C} \rhd \mathrm{D} \sqcup \alpha\mathrm{T} \lhd \beta\mathrm{T} \rhd \mathrm{D} \\ &\overset{(\mathrm{L1})}{=} \alpha\mathrm{T} \lhd (\mathrm{C} \sqcup \beta\mathrm{T}) \rhd \mathrm{D} \\ &\overset{(\mathrm{Lc3,Lc2})}{=} \alpha\mathrm{T}.\end{aligned}$$

The proof of the second part of the lemma is similar using (29). □

Example 4.1. We derive the identity $x \lhd x \rhd x = x$ by writing both sides as optimal normal forms and applying absorption:

$$\begin{aligned}x \lhd x \rhd x &\overset{(25)}{=} \{x\}\mathrm{T} \sqcup \{x, \bar{x}\}\mathrm{F} \sqcup \{x, \bar{x}\}\mathrm{T} \sqcup \{\bar{x}\}\mathrm{F} \\ &\overset{(\mathrm{Abs})}{=} \{x\}\mathrm{T} \sqcup \{\bar{x}\}\mathrm{F} \\ &\overset{(24)}{=} x.\end{aligned}$$

4.3 Derivation of the $\mathbb{K}_4$ axioms in $\mathbb{L}_4$

In this section, we show that the axioms of the logic $\mathbb{K}_4$ are, after translation to $\mathbb{L}_4$, derivable from the $\mathbb{L}_4$ axioms. Together with the proof of the translation invariance presented in the next section, this constitutes a completeness proof for $\mathbb{L}_4$.

For convenience, we repeat our translation from $\mathbb{K}_4$ to $\mathbb{L}_4$ (identities (4), (5), and (6)):

$$\begin{aligned}(\neg x)^* &= \mathrm{F} \lhd x \rhd \mathrm{T}, \\ (x \wedge y)^* &= y \lhd x \rhd \mathrm{F} \sqcup x \lhd y \rhd \mathrm{F}, \\ (x \vee y)^* &= \mathrm{T} \lhd x \rhd y \sqcup \mathrm{T} \lhd y \rhd x.\end{aligned}$$

We further write $\bar{x}$ for $\mathrm{F} \lhd x \rhd \mathrm{T}$ in the setting of $\mathbb{L}_4$.

We list the derivations for the $\mathbb{K}_4$ axioms. All derivations are straightforward. In the cases of the axioms K2–K4, we use rewriting to optimal normal forms, after which application of absorption (Abs) yields the required identity.

N0: this axiom translates to

$$\mathrm{F} \lhd (y \lhd x \rhd \mathrm{F} \sqcup x \lhd y \rhd \mathrm{F}) \rhd \mathrm{T} = \mathrm{T} \lhd \bar{x} \rhd \bar{y} \sqcup \mathrm{T} \lhd \bar{y} \rhd \bar{x}.$$

We derive

$$\begin{aligned}\mathrm{lhs} &= \mathrm{F} \lhd (y \lhd x \rhd \mathrm{F}) \rhd \mathrm{T} \sqcup \mathrm{F} \lhd (x \lhd y \rhd \mathrm{F}) \rhd \mathrm{T} \\ &\overset{(\mathrm{L1,LF})}{=} (\mathrm{F} \lhd y \rhd \mathrm{T}) \lhd x \rhd \mathrm{T} \sqcup (\mathrm{F} \lhd x \rhd \mathrm{T}) \lhd y \rhd \mathrm{T} \overset{(21)}{=} \mathrm{rhs}.\end{aligned}$$

N1: this axiom translates to $\mathrm{F} \lhd (\mathrm{F} \lhd x \rhd \mathrm{T}) \rhd \mathrm{T} = x$, which is derived using axioms L1, LT, LF, and L4.

N2: this axiom translates to $\mathrm{F} \lhd \mathrm{T} \rhd \mathrm{T} = \mathrm{F}$ which is an instance of axiom LT.

N3: this axiom translates to $\mathrm{F} \lhd \mathrm{C} \rhd \mathrm{T} = \mathrm{C}$ which is derived using axioms Lc1 and L4.

N4: this axiom translates to $\mathrm{F} \lhd \mathrm{D} \rhd \mathrm{T} = \mathrm{D}$ which is an instance of axiom LD.

K1: this axiom translates to an instance of axiom LC1.

K2: the left-hand side translates to

$$z \lhd (x \lhd y \rhd \mathrm{F} \sqcup y \lhd x \rhd \mathrm{F}) \rhd \mathrm{F} \sqcup (x \lhd y \rhd \mathrm{F} \sqcup y \lhd x \rhd \mathrm{F}) \lhd z \rhd \mathrm{F},$$

while the right-hand side translates to

$$(y \lhd z \rhd \mathrm{F} \sqcup z \lhd y \rhd \mathrm{F}) \lhd x \rhd \mathrm{F} \sqcup x \lhd (y \lhd z \rhd \mathrm{F} \sqcup z \lhd y \rhd \mathrm{F}) \rhd \mathrm{F}.$$

Straightforward computation yields that both sides equal the optimal normal form $\{x, y, z\}_\mathrm{T} \sqcup \{\bar{x}\}_\mathrm{F} \sqcup \{\bar{y}\}_\mathrm{F} \sqcup \{\bar{z}\}_\mathrm{F}$.

K3: the left-hand side translates to

$$(\mathrm{T} \lhd y \rhd z \sqcup \mathrm{T} \lhd z \rhd y) \lhd x \rhd \mathrm{F} \sqcup x \lhd (\mathrm{T} \lhd y \rhd z \sqcup \mathrm{T} \lhd z \rhd y) \rhd \mathrm{F},$$

while the right-hand side translates to

$$\mathrm{T} \lhd (x \lhd y \rhd \mathrm{F} \sqcup y \lhd x \rhd \mathrm{F}) \rhd (x \lhd z \rhd \mathrm{F} \sqcup z \lhd x \rhd \mathrm{F}) \sqcup$$
$$\mathrm{T} \lhd (x \lhd z \rhd \mathrm{F} \sqcup z \lhd x \rhd \mathrm{F}) \rhd (x \lhd y \rhd \mathrm{F} \sqcup y \lhd x \rhd \mathrm{F}).$$

Straightforward computation yields that both sides equal the optimal normal form $\{\bar{x}\}_\mathrm{F} \sqcup \{\bar{y}, \bar{z}\}_\mathrm{F} \sqcup \{x, y\}_\mathrm{T} \sqcup \{x, z\}_\mathrm{T}$.

K4: this axiom translates to

$$\mathrm{T} \lhd x \rhd (y \lhd x \rhd \mathrm{F} \sqcup x \lhd y \rhd \mathrm{F}) \sqcup \mathrm{T} \lhd (y \lhd x \rhd \mathrm{F} \sqcup x \lhd y \rhd \mathrm{F}) \rhd x = x.$$

Straightforward computation yields that both sides equal the optimal normal form $\{x\}_\mathrm{T} \sqcup \{\bar{x}\}_\mathrm{F}$.

K5: this axiom translates to $x \lhd \mathrm{T} \rhd \mathrm{F} \sqcup \mathrm{T} \lhd x \rhd \mathrm{F} = x$, which is derivable using axioms L4 and LT, and identity (15).

K6: this axiom translates to $\mathrm{D} \lhd \mathrm{C} \rhd \mathrm{F} \sqcup \mathrm{C} \lhd \mathrm{D} \rhd \mathrm{F} = \mathrm{F}$, which is derived using axioms LC1, LC2, and LD.

4.4 Translation invariance

For t a term in the $\mathbb{L}_4$ signature, we write t' for its translation to $\mathbb{K}_4$ (by identity (7)), and for t a term in the $\mathbb{K}_4$ signature, we write t^* for its translation to $\mathbb{L}_4$ (by identities (4), (5) and (6)). We give a proof of the translation invariance: we show that every $\mathbb{L}_4$ term t is derivably equal to $(t')^*$.

Consider the term $t = u \lhd v \rhd w$, where u, v, and w are arbitrary terms. We show that $(t')^* = t$ is derivable in $\mathbb{L}_4$. We apply induction on terms: we assume (IH) that $(x')^* = x$ is derivable for $x = u, v, w$.

First, we translate t according to (7):

$$t' = (s_1 \vee s_2) \vee (s_3 \vee s_4),$$

where

$$\begin{aligned} s_1 &= u' \wedge v', \\ s_2 &= w' \wedge \neg v', \\ s_3 &= (u' \wedge w') \wedge \mathrm{D}, \\ s_4 &= (v' \wedge \neg v') \wedge \mathrm{D}. \end{aligned}$$

Then we translate t' back to $\mathbb{L}_4$, and show that the result is derivably equal to t. We apply the translation to $\mathbb{L}_4$ bottom-up: we first translate the s_i to $\mathbb{L}_4$ terms. We find

$$\begin{aligned} s_1^* &= (u' \wedge v')^* \\ &= (v')^* \triangleleft (u')^* \triangleright \mathrm{F} \sqcup (u')^* \triangleleft (v')^* \triangleright \mathrm{F} \\ &\overset{\text{(IH)}}{=} v \triangleleft u \triangleright \mathrm{F} \sqcup u \triangleleft v \triangleright \mathrm{F}. \end{aligned}$$

Similarly, using the induction hypothesis, we find

$$\begin{aligned} s_2^* &= \bar{v} \triangleleft w \triangleright \mathrm{F} \sqcup w \triangleleft \bar{v} \triangleright \mathrm{F}, \\ s_3^* &= \mathrm{D} \triangleleft \big(w \triangleleft u \triangleright \mathrm{F} \sqcup u \triangleleft w \triangleright \mathrm{F}\big) \triangleright \mathrm{F}, \\ s_4^* &= \mathrm{D} \triangleleft \big(\bar{v} \triangleleft v \triangleright \mathrm{F} \sqcup v \triangleleft \bar{v} \triangleright \mathrm{F}\big) \triangleright \mathrm{F}, \end{aligned}$$

where $\bar{v}$ stands for $\mathrm{F} \triangleleft v \triangleright \mathrm{T}$. Normal forms for the s_i^* terms:

$$\begin{aligned} s_1^* &= \{u, v\}\mathrm{T} \sqcup \{\bar{u}\}\mathrm{F} \sqcup \{\bar{v}\}\mathrm{F}, \\ s_2^* &= \{w, \bar{v}\}\mathrm{T} \sqcup \{\bar{w}\}\mathrm{F} \sqcup \{v\}\mathrm{F}, \\ s_3^* &= \{\bar{u}\}\mathrm{F} \sqcup \{\bar{w}\}\mathrm{F}, \\ s_4^* &= \{v\}\mathrm{F} \sqcup \{\bar{v}\}\mathrm{F}. \end{aligned}$$

Now, we compute a normal form for $(s_1 \vee s_2)^*$. We find that

$$(s_1 \vee s_2)^* = \mathrm{T} \triangleleft s_1^* \triangleright s_2^* \sqcup \mathrm{T} \triangleleft s_2^* \triangleright s_1^*.$$

We derive

$$\begin{aligned} \mathrm{T} \triangleleft s_1^* \triangleright s_2^* &= \mathrm{T} \triangleleft \big(\{u, v\}\mathrm{T} \sqcup \{\bar{u}\}\mathrm{F} \sqcup \{\bar{v}\}\mathrm{F}\big) \triangleright s_2^* \\ &= \mathrm{T} \triangleleft \{u, v\}\mathrm{T} \triangleright s_2^* \sqcup \mathrm{T} \triangleleft \{\bar{u}\}\mathrm{F} \triangleright s_2^* \sqcup \mathrm{T} \triangleleft \{\bar{v}\}\mathrm{F} \triangleright s_2^* \\ &\overset{(26,27)}{=} \mathrm{T} \triangleleft \{u, v\}\mathrm{T} \triangleright \mathrm{D} \sqcup \mathrm{D} \triangleleft \{\bar{u}\}\mathrm{F} \triangleright s_2^* \sqcup \mathrm{D} \triangleleft \{\bar{v}\}\mathrm{F} \triangleright s_2^* \\ &\overset{(28,30)}{=} \{u, v\}\mathrm{T} \sqcup s_2^* \triangleleft \{\bar{u}\}\mathrm{T} \triangleright \mathrm{D} \sqcup s_2^* \triangleleft \{\bar{v}\}\mathrm{T} \triangleright \mathrm{D} \sqcup \\ &= \{u, v\}\mathrm{T} \sqcup \\ &\qquad (\{w, \bar{v}\}\mathrm{T} \sqcup \{\bar{w}\}\mathrm{F} \sqcup \{v\}\mathrm{F}) \triangleleft \{\bar{u}\}\mathrm{T} \triangleright \mathrm{D} \sqcup \\ &\qquad (\{w, \bar{v}\}\mathrm{T} \sqcup \{\bar{w}\}\mathrm{F} \sqcup \{v\}\mathrm{F}) \triangleleft \{\bar{v}\}\mathrm{T} \triangleright \mathrm{D} \sqcup \\ &\overset{(16,28,29)}{=} \{u, v\}\mathrm{T} \sqcup \{\bar{u}, w, \bar{v}\}\mathrm{T} \sqcup \{\bar{u}, \bar{w}\}\mathrm{F} \sqcup \{\bar{u}, v\}\mathrm{F} \sqcup \\ &\qquad \{\bar{v}, w\}\mathrm{T} \sqcup \{\bar{v}, \bar{w}\}\mathrm{F} \sqcup \{\bar{v}, v\}\mathrm{F}. \end{aligned}$$

Similarly, we derive

$$\begin{aligned}\mathrm{T} \lhd s_2^* \rhd s_1^* = {} & \{w, \bar{v}\}_\mathrm{T} \sqcup \{\bar{w}, u, v\}_\mathrm{T} \sqcup \{\bar{w}, \bar{u}\}_\mathrm{F} \sqcup \\ & \{\bar{w}, \bar{v}\}_\mathrm{F} \sqcup \{u, v\}_\mathrm{T} \sqcup \{v, \bar{u}\}_\mathrm{F} \sqcup \{v, \bar{v}\}_\mathrm{F}.\end{aligned}$$

Combining these results, we find by application of aborption (Abs) that

$$\begin{aligned}(s_1 \vee s_2)^* = {} & \{u, v\}_\mathrm{T} \sqcup \{\bar{u}, \bar{w}\}_\mathrm{F} \sqcup \\ & \{\bar{u}, v\}_\mathrm{F} \sqcup \{\bar{v}, \bar{w}\}_\mathrm{F} \sqcup \{\bar{v}, v\}_\mathrm{F} \sqcup \{w, \bar{v}\}_\mathrm{T}.\end{aligned} \tag{31}$$

For $(s_3 \vee s_4)^*$ we find:

$$(s_3 \vee s_4)^* = \mathrm{T} \lhd s_3^* \rhd s_4^* \sqcup \mathrm{T} \lhd s_4^* \rhd s_3^*.$$

We derive

$$\begin{aligned}\mathrm{T} \lhd s_3^* \rhd s_4^* &= \mathrm{T} \lhd (\{\bar{u}\}_\mathrm{F} \sqcup \{\bar{w}\}_\mathrm{F}) \rhd s_4^* \\ &\overset{(\mathrm{L1})}{=} \mathrm{T} \lhd \{\bar{u}\}_\mathrm{F} \rhd s_4^* \sqcup \mathrm{T} \lhd \{\bar{w}\}_\mathrm{F} \rhd s_4^* \\ &\overset{(27,30)}{=} s_4^* \lhd \{\bar{u}\}_\mathrm{T} \rhd \mathrm{D} \sqcup s_4^* \lhd \{\bar{w}\}_\mathrm{T} \rhd \mathrm{D} \\ &= (\{v\}_\mathrm{F} \sqcup \{\bar{v}\}_\mathrm{F}) \lhd \{\bar{u}\}_\mathrm{T} \rhd \mathrm{D} \sqcup (\{v\}_\mathrm{F} \sqcup \{\bar{v}\}_\mathrm{F}) \lhd \{\bar{w}\}_\mathrm{T} \rhd \mathrm{D} \\ &\overset{(16,29)}{=} \{\bar{u}, v\}_\mathrm{F} \sqcup \{\bar{u}, \bar{v}\}_\mathrm{F} \sqcup \{\bar{w}, v\}_\mathrm{F} \sqcup \{\bar{w}, \bar{v}\}_\mathrm{F}.\end{aligned}$$

A similar derivation yields the same normal form for $\mathrm{T} \lhd s_4^* \rhd s_3^*$. Hence,

$$\begin{aligned}(s_3 \vee s_4)^* = {} & \{\bar{u}, v\}_\mathrm{F} \sqcup \{\bar{u}, \bar{v}\}_\mathrm{F} \sqcup \{\bar{w}, v\}_\mathrm{F} \sqcup \{\bar{w}, \bar{v}\}_\mathrm{F}.\end{aligned} \tag{32}$$

We are now ready to compute

$$((s_1 \vee s_2) \vee (s_3 \vee s_4))^*,$$

which equals

$$\mathrm{T} \lhd (s_1 \vee s_2)^* \rhd (s_3 \vee s_4)^* \sqcup \mathrm{T} \lhd (s_3 \vee s_4)^* \rhd (s_1 \vee s_2)^*.$$

We derive

$$\begin{aligned}\mathrm{T} \lhd (s_1 \vee s_2)^* \rhd (s_3 \vee s_4)^* &\overset{(31)}{=} \mathrm{T} \lhd (\{u, v\}_\mathrm{T} \sqcup \{\bar{u}, \bar{w}\}_\mathrm{F} \sqcup \{\bar{u}, v\}_\mathrm{F} \sqcup \\ &\qquad \{\bar{v}, \bar{w}\}_\mathrm{F} \sqcup \{\bar{v}, v\}_\mathrm{F} \sqcup \{w, \bar{v}\}_\mathrm{T}) \rhd (s_3 \vee s_4)^* \\ &\overset{(\mathrm{L1},30,26,27)}{=} \mathrm{T} \lhd \{u, v\}_\mathrm{T} \rhd \mathrm{D} \sqcup \\ &\qquad (s_3 \vee s_4)^* \lhd \{\bar{u}, \bar{w}\}_\mathrm{T} \rhd \mathrm{D} \sqcup \\ &\qquad (s_3 \vee s_4)^* \lhd \{\bar{u}, v\}_\mathrm{T} \rhd \mathrm{D} \sqcup \\ &\qquad (s_3 \vee s_4)^* \lhd \{\bar{v}, \bar{w}\}_\mathrm{T} \rhd \mathrm{D} \sqcup \\ &\qquad (s_3 \vee s_4)^* \lhd \{\bar{v}, v\}_\mathrm{T} \rhd \mathrm{D} \sqcup \\ &\qquad \mathrm{T} \lhd \{w, \bar{v}\}_\mathrm{T} \rhd \mathrm{D} \\ &\overset{(32,16,28,29,\mathrm{Abs})}{=} \{u, v\}_\mathrm{T} \sqcup \{\bar{u}, v\}_\mathrm{F} \sqcup \{\bar{v}, \bar{w}\}_\mathrm{F} \sqcup \{w, \bar{v}\}_\mathrm{T}.\end{aligned}$$

Similarly, we find

$$\mathrm{T} \lhd (s_3 \vee s_4)^* \rhd (s_1 \vee s_2)^* = \{u, \bar{u}, v\}\mathrm{T} \sqcup \{\bar{u}, v\}\mathrm{F} \sqcup \{w, \bar{v}, \bar{u}\}\mathrm{T} \sqcup \{u, \bar{w}, v\}\mathrm{T} \sqcup \{\bar{v}, \bar{w}\}\mathrm{F} \sqcup \{w, \bar{w}, \bar{v}\}\mathrm{T}.$$

Combining these results we find using absorption (Abs):

$$((s_1 \vee s_2) \vee (s_3 \vee s_4))^* = \{u, v\}\mathrm{T} \sqcup \{\bar{u}, v\}\mathrm{F} \sqcup \{\bar{v}, \bar{w}\}\mathrm{F} \sqcup \{w, \bar{v}\}\mathrm{T},$$

where the right-hand side is easily shown to be equal to $u \lhd v \rhd w$.

5 Conclusions

This article follows a number of papers about the combination of process algebra and non-standard propositional logics, among which [5, 6, 7, 8]. Motivation for this line of research is the characterization of erroneous behavior using propositional logics with non-standard truth values.

In [3], Bergstra, Bethke and Rodenburg introduced a four-valued propositional logic comprising the special values D and M (meaningless) and proposed the 'information ordering lattice' where M majorizes T and F, while D is their greatest lower bound. Furthermore, in the spirit of McCarthy [19], these authors introduced special connectives for the sequential interpretation of the usual connectives (instead of directing the evaluation of these, as is done in [19][8]). In particular, *left sequential conjunction*, notation $\owedge$, can be motivated as providing an interpretation of conjunction with an operational, sequential flavor (for instance suitable to represent lazy, left sequential evaluation of conditions in imperative programming). Finally, the truth value M represents a catastrophic notion of 'meaningless': typically, $x \wedge \mathrm{M} = x \vee \mathrm{M} = \neg\mathrm{M} = \mathrm{M}$, whereas for instance $\mathrm{F} \owedge \mathrm{M} = \mathrm{F}$. The truth values D and M can be motivated as covering all types of "errors" that one would want to characterize in error modelling. This four-valued logic, with truth values $\{\mathrm{M}, \mathrm{T}, \mathrm{F}, \mathrm{D}\}$, is combined with process algebra in [6], where a strict correspondence between the truth value D and inaction δ is established.[9]

In [7], a five valued logic with truth values $\{\mathrm{M}, \mathrm{C}, \mathrm{T}, \mathrm{F}, \mathrm{D}\}$ is introduced: in that paper it is observed that Kleene's partial logic admits another interpretation of the 'undefined' value (in our setting: D), namely that of the value C. The question whether this interpretation has any association with a relevant phenomenon, and if so, how the associated truth value can be combined with the values previously distinguished is settled in that paper. (Moreover, the truth value C is added to the information ordering lattice as the least upper bound of T and F, and is majorized by M.) Finally, conditional composition is introduced as a logical operation, making left-sequential conjunction, as well as the associated right-sequential and dual operations, definable:

$$x \owedge y = y \lhd x \rhd \mathrm{F}.$$

This article starts from the observation that with a propositional logic over C and D, the more involved primitives of ACP, i.e., choice and inaction, can be characterized via

[8] In [3], left-sequential and symmetric (or parallel) conjunction both occur in a single logic.

[9] Also in [6], a strict correspondence is established between the truth value M and a process constant μ representing chaos (which can be added to ACP).

conditional composition ($+_{\mathrm{C}}$ and $+_{\mathrm{D}}$, respectively). This justifies the introduction of a four-valued propositional logic over $\{\mathrm{C}, \mathrm{T}, \mathrm{F}, \mathrm{D}\}$ with conditional composition, and the idea to call this logic "the logic of ACP". We studied this logic in detail, and showed that it can either be viewed as a sequential one ($\mathbb{L}_4$, using $\_ \lhd \_ \rhd \_$ as the basic operation), or as a symmetric one ($\mathbb{K}_4$, based on $\wedge$ and $\neg$).

A final word about the truth value C. We may include proposition letters in our logic and choose to interpret these only as T, F, or D (thus excluding C from interpretation). Motivation for this choice is that C typically models a situation that is beyond any means of analysis (as, e.g., the order of interleaving in a concurrent process) or control, and hence a situation that cannot be referenced to by a user-defined proposition. Of course, all our completeness results are preserved when proposition letters are added to one of the logics discussed, and interpretation may follow the consideration raised here.

References

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

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

[3] J.A. Bergstra, I. Bethke, and P.H. Rodenburg. A propositional logic with 4 values: true, false, divergent and meaningless. *Journal of Applied and Non-Classical Logics*, 5(2):199–218, 1995.

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

[5] J.A. Bergstra and A. Ponse. Bochvar-McCarthy logic and process algebra. *Notre Dame Journal of Formal Logic*, 39(4):464–484, 1998.

[6] J.A. Bergstra and A. Ponse. Process algebra with four-valued logic. *Journal of Applied Non-Classical Logics*, 10(1):27–53, 2000.

[7] J.A. Bergstra and A. Ponse. Process algebra with five-valued logic. In C.S. Calude and M.J. Dinneen, editors, *Combinatorics, Computation and Logic*, volume 21(3) of *Australian Computer Science Communications*, pages 128–143. Springer-Verlag, 1999.

[8] J.A. Bergstra and A. Ponse. Process Algebra and Conditional Composition. Information Processing Letters 80(1):41–49, 2001

[9] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier Science, 2001.

[10] G. Birkhoff. On the structure of abstract algebras. In *Proceedings of the Cambridge Philosophical Society 31(4)*, pages 433–454, 1935.

[11] S. Blamey. Partial logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume III: Alternatives to Classical Logic*, pages 1–70. Reidel, Dordrecht, 1986.

[12] E.W. Dijkstra. Cooperating sequential processes. In F. Genuys, editor, *Programming Languages*, pages 43–112, Academic Press, New York, 1968.

[13] 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.

[14] W.J. Fokkink. *Introduction to Process Algebra.* Texts in Theoretical Computer Science. Springer-Verlag, 2000.

[15] 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, 1987.

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

[17] S.C. Kleene. On a notation for ordinal numbers. *Journal of Symbolic Logic*, 3:150–155, 1938.

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

[19] 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.

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