Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 162 (2006) 287–293
www.elsevier.com/locate/entcs

A Generalization of ACP Using Belnap's Logic

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

Programming Research Group, University of Amsterdam, The Netherlands

Abstract

An overview is given of ACP with conditional composition (i.e., *if-then-else*) over Belnap's four-valued logic. The interesting thing is that much of ACP can be analyzed using this logic. For example, both the choice operation $+$ and δ (deadlock) can be seen as instances of conditional composition, and the axiom $x + \delta = x$ follows from this perspective. Furthermore, parallel composition can be generalized to conditional parallel composition, which has sequential composition as an instance, next to common parallel composition, pure interleaving and synchronous ACP.
This article is an extended abstract of [12]. The full article contains all proofs and some examples on parallel scheduling in GACP.

Keywords: Many-Valued Logic, Belnap's Logic, Process Algebra, ACP, Conditional Composition, Choice

1 Introduction

In 1994, Jan Bergstra and co-workers experienced a revival in the specification of datatypes with divergence, errors and recovery or exception handling. This was triggered by languages such as VDM [8] and an upcoming interest in Java [6]. The first outcome was an article on a four-valued propositional logic by Bergstra, Bethke and Rodenburg [2]. Consequently it was felt that a combination with ACP [3] via a conditional composition construct (i.e., an *if-then-else* operator) was obvious, and a first paper involving Kleene's three-valued logic [9] was written [4], the idea being that in

$$\textit{if } \phi \textit{ then } P \textit{ else } Q$$

the condition ϕ may take Kleene's truth value *undefined*. This led to [5] in which the logic of [2] is combined with ACP, to papers in which other non-classical logics were used, and ultimately to the four-valued logic $\mathbf{C}_4$ for ACP with conditional

[1] Email: alban@science.uva.nl
[2] Email: mbz@science.uva.nl

doi:10.1016/j.entcs.2005.12.102

composition (in [11] baptized "the logic of ACP"). In [11] we show that this logic (with one, sequential connective) is equivalent to the natural extension of Kleene's three-valued logic with a fourth truth value (which has symmetric connectives). It was only two years ago that we found out that this latter logic is Belnap's four-valued logic [1], as we could have known from, e.g., [7].

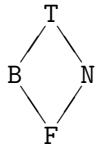
In this article we focus on process-algebraic conditional composition over Belnap's logic. A tricky corner in ACP is the combination of choice and deadlock. One often reads that the process $x+y$ makes a choice between x and y. However, this is not true for $a+\delta$, where a is an action and δ represents deadlock (indeed, a standard ACP axiom is $x+\delta=x$). Can choice in this case be seen as a *prescriptive* operation? In this article we show that it can: there is a straightforward correspondence with conditional composition over Belnap's logic, allowing one to explain the nature of choice in process algebra from a logical perspective. We generalize alternative composition $+$ to conditional composition $+_\phi$ so that

$$x+y = x +_{\mathtt{B}} y$$

where B (both) is the truth value which stands for *both true and false*. We describe our results only informally; all proofs can be found in [11,12].

2 Belnap's Logic and Conditional Composition

Belnap's logic $\mathbf{B}_4$ [1] has truth values B, T, F, and N, where B (both) represents both true and false, T and F are the values true and false, and N (none) represents undefinedness.[3] Negation is defined as an involution (satisfying $\neg\neg x = x$) by $\neg\mathtt{B}=\mathtt{B}$, $\neg\mathtt{T}=\mathtt{F}$, $\neg\mathtt{F}=\mathtt{T}$, and $\neg\mathtt{N}=\mathtt{N}$, and conjunction ($\wedge$) and disjunction ($\vee$) are the greatest lower bound and the least upper bound in the distributive lattice



called the *truth ordering* [7]. This characterization of the logic as a distributive lattice with involution leads directly to a finite and complete equational axiomatization [11].

Now we define an alternative logic $\mathbf{C}_4$ over these truth values that has only one, ternary operation $\_ \lhd \_ \rhd \_$ called *conditional composition*. This operation is defined by

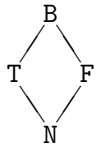
$$x \lhd \mathtt{T} \rhd y = x, \quad x \lhd \mathtt{F} \rhd y = y, \quad x \lhd \mathtt{N} \rhd y = \mathtt{N},$$

[3] Belnap motivated B as the result of conflicting outcomes of database queries, and N as the absence of answers.

(C1)	$x \lhd (u \lhd v \rhd w) \rhd y = (x \lhd u \rhd y) \lhd v \rhd (x \lhd w \rhd y)$
(C2)	$(x \lhd w \rhd y) \lhd v \rhd (x' \lhd w \rhd y') = (x \lhd v \rhd x') \lhd w \rhd (y \lhd v \rhd y')$
(C3)	$(x \lhd w \rhd y) \lhd w \rhd z = x \lhd w \rhd (y \lhd w \rhd z)$
(C4)	$\mathtt{T} \lhd x \rhd \mathtt{F} = x$
(C5)	$x \lhd \mathtt{T} \rhd y = x$
(C6)	$x \lhd \mathtt{F} \rhd y = y$
(C7)	$x \lhd \mathtt{N} \rhd y = \mathtt{N}$
(C8)	$x \lhd \mathtt{B} \rhd y = y \lhd \mathtt{B} \rhd x$
(C9)	$x \lhd \mathtt{B} \rhd \mathtt{N} = x$
(C10)	$\mathtt{B} \lhd \mathtt{B} \rhd x = \mathtt{B}$

Table 1
$\mathbf{C}_4$ axioms

and $x \lhd \mathtt{B} \rhd y = x \sqcup y$, where $\sqcup$ is the least upper bound of x and y in the lattice



called the *information (*or *knowledge) ordering* [2,7]. Conditional composition has an operational, sequential reading: in $x \lhd y \rhd z$, first y is evaluated, and depending on the outcome, possibly x and/or z. In Table 1 we give a complete set of axioms for $\mathbf{C}_4$.

The logics $\mathbf{B}_4$ and $\mathbf{C}_4$ have exactly the same expressiveness, that is, their operations can be defined in each other: using $\mathtt{B}, \mathtt{T}, \mathtt{F}$ we have

$$\neg x = \mathtt{F} \lhd x \rhd \mathtt{T}, \quad x \wedge y = (y \lhd x \rhd \mathtt{F}) \lhd \mathtt{B} \rhd (x \lhd y \rhd \mathtt{F}),$$

and, vice versa, using $\mathtt{N}$,

$$x \lhd y \rhd z = (x \wedge y) \vee (z \wedge \neg y) \vee (x \wedge z \wedge \mathtt{N}) \vee (y \wedge \neg y \wedge \mathtt{N}).$$

Hence the two logics can be considered "the same", but with a different functional basis. We show that the logics are truth-functionally complete for monotone functions with respect to the information ordering. Let f be a $(k+1)$-ary monotone function and write $\bar{x}, y$ for $(k+1)$-tuples. By monotonicity of f,

$$f(\bar{x}, y) = f(\bar{x}, \mathtt{N}) \sqcup (f(\bar{x}, \mathtt{T}) \lhd y \rhd f(\bar{x}, \mathtt{F})) \sqcup ((\mathtt{N} \lhd y \rhd f(\bar{x}, \mathtt{B})) \lhd y \rhd \mathtt{N}).$$

(G1)	$x +_{\phi \lhd \psi \rhd \chi} y = (x +_\phi y) +_\psi (x +_\chi y)$
(G2)	$(x +_\psi y) +_\phi (x' +_\psi y') = (x +_\phi x') +_\psi (y +_\phi y')$
(G3)	$x +_\phi (y +_\phi z) = (x +_\phi y) +_\phi z$
(G4)	$(x +_\phi y)z = xz +_\phi yz$
(G5)	$(xy)z = x(yz)$
(G6)	$x +_{\mathtt{T}} y = x$
(G7)	$x +_{\mathtt{F}} y = y$
(G8)	$x +_{\mathtt{N}} y = \delta$
(†)	$x +_\phi y = x +_\psi y$ if $\mathbf{C}_4 \vdash \phi = \psi$

Table 2
GBPA$_\delta$ axioms

By induction on k, the function f is expressible (because $f(\bar{x}, a)$ is, for all truth values a).

3 Conditional Composition in Process Algebra

We now look at GBPA$_\delta$, a generalization of BPA$_\delta$ (Basic Process Algebra with deadlock, which includes sequential composition $\cdot$, alternative composition $+$ and the constant δ for deadlock). In GBPA$_\delta$ alternative composition is parametrized with $\mathbf{C}_4$ terms ϕ, hence obtaining the operation $+_\phi$ called conditional composition, and $x +_\phi y$ is read as *if* ϕ *then* x *else* y. The axiom system GBPA$_\delta$ consists of the axioms in Table 2. Deadlock corresponds to $+_{\mathtt{N}}$ by axiom G8, and alternative composition can be seen as the instance $+_{\mathtt{B}}$, as will be shown.

Next, we give an operational semantics for *process-closed* terms, that is, for process terms that do not contain process variables. Let W be the set of valuations for $\mathbf{C}_4$ terms, let A be the nonempty set of action symbols that comes as a parameter of the axiom system, and let $A \times W$ be the set of transition labels. The transition rules are given in the upper part of Table 4, where a transition with label a, w models the execution of action a under valuation w.

We define (strong) bisimulation as usual. Process-closed terms are bisimilar ($\leftrightarroweq$) if they are related by a bisimulation. Since bisimilar terms have matching action steps for every valuation, we allow (user-defined) propositions in the logic, the evaluation of which may not be constant throughout the execution of a process. The transition rules are in the *panth* format [13], from which it follows that bisimilarity is a congruence. Furthermore, the GBPA$_\delta$ axioms are sound and complete [11,12].

Our claim that $+$ equals $+_{\mathtt{B}}$ is substantiated by showing that all axioms of BPA$_\delta$

are derivable in GBPA_δ. Commutativity of alternative composition is derived by

$$x +_{\mathtt{B}} y = (y +_{\mathtt{F}} x) +_{\mathtt{B}} (y +_{\mathtt{T}} x) = y +_{\mathtt{F} \lhd \mathtt{B} \rhd \mathtt{T}} x = y +_{\mathtt{B}} x$$

using axioms G6, G7, G1, C8, and C4. Associativity is an instance of G3. Idempotence:

$$x +_{\mathtt{B}} x = (x +_{\mathtt{T}} y) +_{\mathtt{B}} (x +_{\mathtt{T}} y) = x +_{\mathtt{T} \lhd \mathtt{B} \rhd \mathtt{T}} y = x$$

using G6, G1, and $\mathbf{C}_4 \vdash x \lhd \mathtt{B} \rhd x = x$. Right-distributivity of sequential composition over alternative composition is an instance of G4. The axiom $x + \delta = x$ can be derived by

$$x +_{\mathtt{B}} \delta = (x +_{\mathtt{T}} y) +_{\mathtt{B}} (x +_{\mathtt{N}} y) = x +_{\mathtt{T} \lhd \mathtt{B} \rhd \mathtt{N}} y = x$$

using G6, G8, G1, and C9. Finally, the axiom $\delta x = \delta$ can be derived using G8 and G4.

We find that process-algebraic and logical conditional composition are quite similar, as becomes apparent when one compares their axioms. The process-algebraic counterpart of the information ordering ($\leq$) is the summand inclusion ordering $\subseteq$ defined by $x \subseteq y \Leftrightarrow x + y = y$. So alternative composition can be said to be the counterpart of $\lhd \mathtt{B} \rhd$, while δ corresponds to $\mathtt{N}$. The following result implies that $\mathbf{C}_4$ characterizes choice and deadlock:

Proposition 3.1 *For $i = 1, 2$, let p_i be an open process term in which no action symbols occur and the only operation is conditional composition. Let t_i be the $\mathbf{C}_4$ term which is obtained from p_i by interpreting $+_\phi$ as $\lhd \phi \rhd$ and δ as $\mathtt{N}$, and in which the process variables also represent propositions. Then $\mathrm{GBPA}_\delta/\leftrightarrows \models p_1 \subseteq p_2$ iff $\mathbf{C}_4 \models t_1 \leq t_2$, and hence $\mathrm{GBPA}_\delta/\leftrightarrows \models p_1 = p_2$ iff $\mathbf{C}_4 \models t_1 = t_2$.*

4 Generalized Parallel Composition

GACP (Generalized ACP) is parametrized with a nonempty set A of action symbols, and a commutative and associative function $| : A \times A \to A \cup \{\delta\}$ which defines which actions communicate. It extends GBPA_δ with a generalization ${}_\phi\|_\psi$ of parallel composition, where the condition ϕ covers the choice between interleaving and synchronization, and ψ may determine the order of execution. Furthermore, it has an auxiliary generalized left merge ${}_\phi\mathbb{L}_\psi$ and generalized communication merge ${}_\phi|_\psi$, and the encapsulation operation ∂_H, which renames actions from $H \subseteq A$ to δ. The axioms are those of GBPA_δ with four straightforward axioms for encapsulation (omitted here) and those in Table 3.

The operation ${}_{\mathtt{T}}\|_{\mathtt{B}}$ restricts to interleaving (free merge), while ${}_{\mathtt{F}}\|_\diamond$ for $\diamond \in \{\mathtt{B}, \mathtt{T}, \mathtt{F}\}$ defines synchronous merge, and ${}_{\mathtt{T}}\|_{\mathtt{T}}$ represents sequential composition. Some typical identities:

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

The transition rules are presented in Table 4. Bisimilarity is a congruence, and all axioms are sound in the model thus obtained. The parallel composition

(GM1)	$x \mathbin{\|_\phi} y = (x \mathbin{\lfloor\!\lfloor_\psi} y +_\psi y \mathbin{\lfloor\!\lfloor_{\neg\psi}} x) +_\phi (x \mathbin{\lfloor_\phi} y +_\psi y \mathbin{\lfloor_{\neg\psi}} x)$
(GM2)	$a \mathbin{\lfloor\!\lfloor_\psi} x = ax$
(GM3)	$ax \mathbin{\lfloor\!\lfloor_\psi} y = a(x \mathbin{\|_\phi} y)$
(GM4)	$(x +_\phi y) \mathbin{\lfloor\!\lfloor_\chi} z = x \mathbin{\lfloor\!\lfloor_\chi} z +_\phi y \mathbin{\lfloor\!\lfloor_\chi} z$
(GM5)	$a \mathbin{\lfloor_\phi}_\psi b = a \mid b$
(GM6)	$a \mathbin{\lfloor_\phi}_\psi bx = (a \mid b)x$
(GM7)	$ax \mathbin{\lfloor_\phi}_\psi b = (a \mid b)x$
(GM8)	$ax \mathbin{\lfloor_\phi}_\psi by = (a \mid b)(x \mathbin{\|_\phi} y)$
(GM9)	$(x +_\phi y) \mathbin{\lfloor_\psi}_\chi z = x \mathbin{\lfloor_\psi}_\chi z +_\phi y \mathbin{\lfloor_\psi}_\chi z$
(GM10)	$z \mathbin{\lfloor_\phi}_\psi (x +_\chi y) = z \mathbin{\lfloor_\phi}_\psi x +_\chi z \mathbin{\lfloor_\phi}_\psi y$

Table 3
Axioms for generalized merge; a, b range over A

operations can be eliminated from process-closed terms, so GACP is complete for these terms.

5 Conclusion

We have argued that conditional composition over Belnap's logic characterizes choice and deadlock in ACP from a logical perspective. We further remark that conditional composition can be used to define sequential connectives such as McCarthy's directed $\wedge$ [10], left sequential conjunction $\wedge\!\!\!\!\diagup$ [2], and Fitting's $\overrightarrow{\wedge}$ [7]. E.g., $x \mathbin{\wedge\!\!\!\!\diagup} y = y \lhd x \rhd \mathtt{F}$. Finally we remark that the generalized merge can be used to model parallel scheduling, see [11,12] for examples.

References

[1] N.D. Belnap. A Useful Four-Valued Logic. In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8-37, D. Reidel, 1977.

[2] 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.

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

[4] J.A. Bergstra and A. Ponse. Kleene's three-valued logic and process algebra. *Information Processing Letters*, 67(2):95-103, 1998.

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

[6] G. Bracha, et al. *The Java Language Specification* (2nd edition). Addison Wesley, 2000.

[7] M.C. Fitting. Kleene's three valued logics and their children. *Fundamenta Informaticae*, 20:113-131, 1994.

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

$$\frac{x \xrightarrow{a,w} x'/\surd,\ w(\phi) \in \{\mathtt{B},\mathtt{T}\}}{x +_\phi y \xrightarrow{a,w} x'/\surd} \qquad \frac{x \xrightarrow{a,w} x'/\surd,\ w(\phi) \in \{\mathtt{B},\mathtt{F}\}}{y +_\phi x \xrightarrow{a,w} x'/\surd}$$

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

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

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

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

$$\frac{x \xrightarrow{a,w} x'/\surd,\ a \notin H}{\partial_H(x) \xrightarrow{a,w} \partial_H(x'/\surd)}$$

Table 4
Transition rules. $a, b, c \in A$; $w \in W$; $x'/\surd$ and $y'/\surd$ range over $P \cup \{\surd\}$, where P is the set of process-closed terms; and $x \,{}_\phi\|_\psi \surd = \surd \,{}_\phi\|_\psi\, x = x$, $\surd \,{}_\phi\|_\psi \surd = \partial_H(\surd) = \surd$

[8] C.B. Jones. *Systematic Software Development using VDM* (2nd edition). Prentice-Hall International, Englewood Cliffs, 1990.

[9] S.C. Kleene. On a notation for ordinal numbers. *J. of Symbolic Logic*, 3:150-155, 1938.

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

[11] A. Ponse and M.B. van der Zwaag. The logic of ACP. Report SEN-R0207, CWI, 2002.

[12] A. Ponse and M.B. van der Zwaag. A generalization of ACP using Belnap's logic. Submitted to *Journal of Logic and Algebraic Programming*.

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