Proposition algebra with projective limits

Bergstra, J.A.; Ponse, A.

Publication date
2008
Document Version
Submitted manuscript

Proposition Algebra with Projective Limits

Jan A. Bergstra*
Alban Ponse

Section Software Engineering, Informatics Institute, University of Amsterdam
Email: {janb,alban}@science.uva.nl

Abstract

Sequential propositional logic deviates from ordinary propositional logic by taking into account that during the sequential evaluation of a proposition, atomic propositions may yield different Boolean values at repeated occurrences. We introduce 'free reactive valuations' to capture this dynamics of a proposition's environment. The resulting logic is phrased as an equationally specified algebra rather than in the form of proof rules, and is named 'proposition algebra'. It is strictly more general than Boolean algebra to the extent that the classical connectives fail to be expressively complete in the sequential case.

Proposition algebra is developed in a fashion similar to the process algebra ACP and the program algebra PGA, via an algebraic specification which has a meaningful initial algebra for which a range of courser congruences are considered important as well. In addition infinite objects (that is propositions, processes and programs respectively) are dealt with by means of an inverse limit construction which allows the transfer of knowledge concerning finite objects to facts about infinite ones while reducing all facts about infinite objects to an infinity of facts about finite ones in return.

1 Introduction

A proposition is a composition of atomic propositions made by means of one or more (proposition) composition mechanisms, usually called connectives. Atomic propositions are considered to represent facts about an environment (execution environment, execution architecture, operating context) that are used by the logical mechanism contained in the proposition which aggregates these facts for presentation to the proposition's user. Different occurrences of the same atomic propositions represent different queries (measurements, issued information requests) at different moments in time.

A valuation that may return different Boolean values for the same atomic proposition during the evaluation of a single proposition is called *reactive*. This is in contrast with a "static" valuation, which always returns the same value for the same atomic proposition.

*J.A. Bergstra acknowledges support from NWO (project Thread Algebra for Strategic Interleaving).

Reactive valuations are thus semantically at the opposite end of static valuations, and are *observation based* in the sense that they capture the identity of a proposition as a (possibly empty) series of queries followed by a Boolean value. In the context of reactive valuations, we further speak of *active propositions*.

Many classes of reactive valuations can be distinguished. Given a class K of reactive valuations, two propositions are K-equivalent if they evaluate to the same Boolean value for each valuation in K. Given a family of proposition connectives, K-equivalence need not be a congruence, and K-congruence is the largest congruence that is contained in K-equivalence. It is obvious that with larger K more propositions can be distinguished and the one we consider most distinguishing is named *free reactive valuation congruence*. It is this congruence that plays the role of an initial algebra for the proposition algebras developed in this paper. The axioms of proposition algebra specify free reactive valuation congruence in terms of the single ternary connective *conditional composition* (in computer science terminology: *if-then-else*) and constants for truth and falsity, and their soundness and completeness (for closed equations) is easily shown. Additional axioms are given for static valuation congruence, and for some reactive valuation congruences in between.

Sequential versions of the well-known binary connectives of proposition logic and negation can be expressed with conditional composition. We prove that these connectives have insufficient expressive power at this level of generality and that a ternary connective is needed (in fact this holds for any collection of binary connectives definable by conditional composition.)

Infinite active propositions are defined by means of an inverse limit construction which allows the transfer of knowledge concerning finite objects to facts about infinite ones while reducing all facts about infinite objects to an infinity of facts about finite ones in return. This construction was applied in giving standard semantics for the process algebra ACP (see [5] and for a more recent overview [1]). In doing so the design of proposition algebra is very similar to the thread algebra of [6] which is based on a similar ternary connective but features constants for termination and deadlock rather than for truth and falsity. Whereas thread algebra focuses on multi-threading and concurrency, proposition algebra, however, has a focus on sequential mechanisms. At the same time free reactive valuation congruence constitutes the preferred setting in the context of thread algebra and an extreme case for proposition algebra. The reason for this discrepancy lies in the fact that within thread algebra side effects of actions (atomic proposition queries) are intended and expected while in propositional algebra such side effects are considered rather more problematic though not excluded in principle.

The paper is structured as follows: in the next section we discuss some motivation for proposition algebra. In Section 3 we define the signature and equations of proposition algebra, and in Section 4 we formally define reactive valuations. In Section 5 we consider some observation based equivalences and congruences generated by reactive valuations, and in Section 6 some related complexity issues. Definable (binary) connectives and their expressiveness (functional incompleteness) are considered in Sections 7 and 8, respectively. In Section 9 we discuss a connection between three-valued logic and proposition algebra, which is continued in Appendix A. In Section 10 we introduce projection and projective limits for defining potentially infinite active propositions, and in Section 11 we discuss recursive specifications of such propositions. The paper is ended with some conclusions in Section 12.

2 Motivation for proposition algebra

Proposition algebra for active propositions CP is proposed as a preferred way of viewing the data type of propositions, at least in a context of sequential systems. Here are some arguments in favor of that thesis:

In a sequential program a test which is a conjunction of P and Q will be evaluated in a sequential fashion, beginning with P and not evaluating Q unless the evaluation of P led to a positive outcome. The sequential form of evaluation takes precedence over the axioms or rules of classical propositional calculus or Boolean algebra. For instance neither conjunction nor disjunction are commutative when evaluated sequentially in the presence of side-effects, errors or exceptions. The absence of these latter features is never claimed for imperative programming and thus some extension of ordinary two-valued logic is necessary to understand the basics of propositional logic as it occurs in the context of imperative programs. Three-, four- or more sophisticated many-valued logics may be used to explain the logic in this case (see, e.g., [4, 7, 16]), and the non-commutative, sequential reading of conjunction mentioned above can be traced back to McCarthy's seminal work on computation theory [22], in which a specific value for undefinedness (e.g., a divergent computation) is considered that in conjunction with falsity results in what was evaluated first. We return to this subject in Section 9.

Importing non-commutative conjunction to two valued proposition logic means that the sequential order of events is taken decisive, and that is what proposition algebra is meant to specify and analyze in the first place. As a simple example, consider the active proposition a pedestrian processes just before crossing a road with two-way traffic driving on the right:

$$\textit{look-left-and-check} \mathbin{\diagup\!\!\circ\!\!\wedge} \textit{look-right-and-check} \mathbin{\diagup\!\!\circ\!\!\wedge} \textit{look-left-and-check}. \tag{1}$$

Here $\mathbin{\diagup\!\!\circ\!\!\wedge}$ is *left-sequential conjunction*, which is as normal conjunction but the left argument is evaluated first and upon false, evaluation finishes with result false. Valuations associated with this example are (should be) reactive: also in the case that the leftmost occurrence of *look-left-and-check* evaluates to true, its second evaluation might very well evaluate to false. However, the order of events (or their amount) needs not to be taken decisive in all circumstances and one may still wish or require that in particular cases conjunction is idempotent or commutative. A most simple example is perhaps

$$a \mathbin{\diagup\!\!\circ\!\!\wedge} a = a$$

with a an atomic proposition, which is not valid in free reactive valuation semantics (and neither is the falsity of $a \mathbin{\diagup\!\!\circ\!\!\wedge} \neg a$). For this reason we distinguish two restricted forms of reactive valuation equivalence/congruence that both validate variations of this example, but still refine static valuation congruence. It is evident that many more such refinements can or even should be distinguished.

So, we take the point of departure that the very purpose of any action taken by a program under execution is to change the state of a system. If no change of state results with certainty the action can just as well be skipped. This holds for tests as well as for any action mainly performed because of its intended side-effects. The common intuition that the state is an external matter, not influenced by the evaluation of a test justifies ignoring side-effects of

tests and for that reason it justifies an exclusive focus on static valuations to a large extent, thereby rendering the issue of reactivity pointless as well. But there are some interesting cases where this intuition is not necessarily convincing. We mention three such issues, all of which also support the general idea of considering propositions under sequential evaluation:

1. It is common to accept that in a mathematical text an expression $1/x$ is admissible only after a test $x \neq 0$ has been performed. One might conceive this test as an action changing the state of mind of the reader thus influencing the evaluation of further assertions such as $x/x = 1$.

2. A well-known dogma on computer viruses introduced by Cohen in 1984 (in [13]) states that a computer cannot decide whether or not a program that it is running is itself a virus. The proof involves a test which is hypothetically enabled by a decision mechanism which is supposed to have been implemented on a modified instance of the machine under consideration. It seems fair to say that the property of a program being viral is not obviously independent of the state of the program. So here is a case where performing the test might (in principle at least) result in a different state from which the same test would lead to a different outcome.
   This matter has been analyzed in detail in [9, 3] with the conclusion that the reactive nature of valuations gives room for criticism of Cohen's original argument. In the didactic literature on computer security Cohen's viewpoint is often repeated and it can be found on many websites and in the introduction of many texts. But there is a remarkable lack of secondary literature on the matter; an exception is the discussion in [14] and the papers cited therein. In any case the common claim that this issue is just like the halting problem (and even more important in practice) is open for discussion.

3. The on-line halting problem is about execution environments which allow a running program to acquire information about its future halting or divergence. This information is supposed to be provided by means of a forecasting service. In [10] that feature is analyzed in detail in a setting of thread algebra and the impossibility of sound and complete forecasting of halting is established. In particular, calling a forecasting service may have side-effects which leads to different replies in future calls (see, e.g., [23]). This puts the impossibility of effectively deciding the halting problem at a level comparable to the impossibility of finding a convincing truth assignment for the liar paradox sentence in conventional two valued logic.

Our account of proposition algebra is based on the ternary operator *conditional composition* (or *if-then-else*). This operator has sequential evaluation as its natural semantics, and thus combines naturally with reactive valuation semantics. Furthermore, proposition algebra constitutes a simple setting for constructing infinite active propositions by means of an inverse limit construction. The resulting *projective limit model* can be judged as one that didactically precedes (prepares for) technically more involved versions for process algebra and thread algebra, and as such provides by itself a motivation for proposition algebra.

(CP1)	$x \lhd T \rhd y = x$
(CP2)	$x \lhd F \rhd y = y$
(CP3)	$T \lhd x \rhd F = x$
(CP4)	$x \lhd (y \lhd z \rhd u) \rhd v = (x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v)$

Table 1: The set CP of axioms for proposition algebra

3 Proposition algebra and active propositions

In this section we introduce the signature and equational axioms of proposition algebra. Let A be a finite set of atomic propositions, ranged over by $a, b, c, ...$ The elements of A serve as atomic (i.e., non-divisible) queries that will produce a Boolean reply value. Issuing $a \in A$ may have a side-effect that influences forthcoming replies for a or other $b \in A$.

The signature of proposition algebra consists of the constants T and F (representing true and false), a constant a for each $a \in A$, and, following Hoare in [18], the ternary operator *conditional composition*

$$\_ \lhd \_ \rhd \_.$$

Terms are subject to equational logic and the axioms in Table 1. We further write CP for this set of axioms (for conditional propositions), and $\Sigma_{\mathrm{CP}}(A)$ for the signature introduced here.

An alternative name for the conditional composition $y \lhd x \rhd z$ is *if x then y else z*: the axioms CP1 and CP2 model that its central condition x is evaluated first, and depending on the reply either its leftmost or rightmost argument is evaluated. Axiom CP3 establishes that a term can be extended to a larger conditional composition by adding T as a leftmost argument and F as a rightmost one, and CP4 models the way a non-atomic central condition distributes over the outer arguments. We note that the expression

$$F \lhd x \rhd T$$

can be seen as defining the negation of x:

$$\mathrm{CP} \vdash z \lhd (F \lhd x \rhd T) \rhd y = (z \lhd F \rhd y) \lhd x \rhd (z \lhd T \rhd y) = y \lhd x \rhd z, \tag{2}$$

which illustrates that "*if* $\neg x$ *then* z *else* y" and "*if* x *then* y *else* z" are considered equal.

We introduce the abbreviation

$$x \circ y \quad \text{for} \quad y \lhd x \rhd y,$$

and we name this expression x *and then* y. It follows easily that $\circ$ is associative:

$$(x \circ y) \circ z = z \lhd (y \lhd x \rhd y) \rhd z = (z \lhd y \rhd z) \lhd x \rhd (z \lhd y \rhd z) = x \circ (y \circ z).$$

We take the and-then operator $\circ$ to bind stronger than conditional composition. In a later stage we will formally add negation, the "and then" connective $\circ$, and some other binary connectives to proposition algebra (i.e., add their function symbols to $\Sigma_{\mathrm{CP}}(A)$ and their defining equations to CP).

Closed terms over $\Sigma_{\mathrm{CP}}(A)$ are called *active propositions* or simply *propositions*, and we write $\mathbb{P}$ for the set of active propositions over A with typical elements $P, Q, R, ...$

Definition 1. *A proposition P is a **basic form** if*

$$P ::= T \mid F \mid P_1 \lhd a \rhd P_2$$

with $a \in A$, and P_1 and P_2 basic forms.

So, basic forms can be seen as binary trees of which the leaves are labeled with either T or F, and the internal nodes with an atomic proposition.

Lemma 1. *Each proposition in $\mathbb{P}$ can be proved equal to one in basic form by the axioms in Table 1.*

Proof. By structural induction on the form that proposition P may take. If $P = T$ or $P = F$ we are done.

If $P = a$, then $\mathrm{CP} \vdash P = T \lhd a \rhd F$.

For the case $P = Q \lhd R \rhd S$ we may assume without loss of generality that Q, R, S are basic forms. If $R = T$ or $R = F$ the result follows immediately; if $R = R_1 \lhd a \rhd R_2$, then, by structural induction,

$$\mathrm{CP} \vdash Q \lhd R \rhd S = Q \lhd (R_1 \lhd a \rhd R_2) \rhd S = (Q \lhd R_1 \rhd S) \lhd a \rhd (Q \lhd R_2 \rhd S) = Q' \lhd a \rhd S'$$

for basic forms Q', S'. □

By structural induction it follows that basic forms constitute a convenient representation:

Proposition 1. *For basic forms, provable equality and syntactic equality coincide.*

4 Reactive valuations

In this section we formally define reactive valuations. Let $\mathbb{B} = \{T, F\}$ be the set of Booleans. The signature $\Sigma_{ReVal}(A)$ of reactive valuations extends $\mathbb{B}$ with a sort RV and for each $a \in A$ a function

$$y_a : RV \to \mathbb{B}$$

called the *yield* of a, and a function

$$\tfrac{\partial}{\partial a} : RV \to RV$$

called the a-derivative. Furthermore, RV has two constants

$$T_{RV} \quad \text{and} \quad F_{RV},$$

which represent the valuations that assign to each atomic proposition the value T respectively F.

A structure $\mathbb{A}$ over $\Sigma_{ReVal}(A)$ is a *reactive valuation algebra* (RVA) if for all $a \in A$ it satisfies the axioms

$$\begin{aligned}
&y_a(T_{RV}) = T,\\
&y_a(F_{RV}) = F,\\
&\tfrac{\partial}{\partial a}(T_{RV}) = T_{RV},\\
&\tfrac{\partial}{\partial a}(F_{RV}) = F_{RV}.
\end{aligned}$$

Given a reactive valuation algebra $\mathbb{A}$ with element H and a proposition P we define simultaneously the evaluation of P over H, written

$$P/H,$$

and a generalized notion of an a-derivative for proposition Q, notation $\frac{\partial}{\partial Q}(H)$ by the following case distinctions:

$$\begin{aligned}
T/H &= T,\\
F/H &= F,\\
a/H &= y_a(H),\\
(P \triangleleft Q \triangleright R)/H &= \begin{cases} P/\frac{\partial}{\partial Q}(H) & \text{if } Q/H = T,\\ R/\frac{\partial}{\partial Q}(H) & \text{if } Q/H = F, \end{cases}
\end{aligned}$$

and

$$\begin{aligned}
\tfrac{\partial}{\partial T}(H) &= H,\\
\tfrac{\partial}{\partial F}(H) &= H,\\
\tfrac{\partial}{\partial (P \triangleleft Q \triangleright R)}(H) &= \begin{cases} \frac{\partial}{\partial P}(\frac{\partial}{\partial Q}(H)) & \text{if } Q/H = T,\\ \frac{\partial}{\partial R}(\frac{\partial}{\partial Q}(H)) & \text{if } Q/H = F. \end{cases}
\end{aligned}$$

Some explanation: whenever in a conditional composition the central condition is an atomic proposition, say c, then a valuation H distributes over the outer arguments as $\frac{\partial}{\partial c}(H)$, thus

$$(P \triangleleft c \triangleright Q)/H = \begin{cases} P/\frac{\partial}{\partial c}(H) & \text{if } y_c(H) = T,\\ Q/\frac{\partial}{\partial c}(H) & \text{if } y_c(H) = F. \end{cases} \tag{3}$$

If in a conditional composition the central condition is not atomic, valuation decomposes

further according to the equations above, e.g.,

$$(a \lhd (b \lhd c \rhd d) \rhd e)/H = \begin{cases} a/\frac{\partial}{\partial(b \lhd c \rhd d)}(H) & \text{if } (b \lhd c \rhd d)/H = T, \\ e/\frac{\partial}{\partial(b \lhd c \rhd d)}(H) & \text{if } (b \lhd c \rhd d)/H = F, \end{cases}$$
$$= \begin{cases} a/\frac{\partial}{\partial b}(\frac{\partial}{\partial c}(H)) & \text{if } y_c(H) = T \text{ and } y_b(\frac{\partial}{\partial c}(H) = T, \\ a/\frac{\partial}{\partial d}(\frac{\partial}{\partial c}(H)) & \text{if } y_c(H) = F \text{ and } y_d(\frac{\partial}{\partial c}(H) = T, \\ e/\frac{\partial}{\partial b}(\frac{\partial}{\partial c}(H)) & \text{if } y_c(H) = T \text{ and } y_b(\frac{\partial}{\partial c}(H) = F, \\ e/\frac{\partial}{\partial d}(\frac{\partial}{\partial c}(H)) & \text{if } y_c(H) = F \text{ and } y_d(\frac{\partial}{\partial c}(H) = F. \end{cases} \quad (4)$$

We compare the last example with

$$((a \lhd b \rhd e) \lhd c \rhd (a \lhd d \rhd e))/H, \quad (5)$$

which is a particular instance of (3) above. For the case $y_c(H) = T$ we find from (3) that

$$(5) = (a \lhd b \rhd e)/\tfrac{\partial}{\partial c}(H) = \begin{cases} a/\frac{\partial}{\partial b}(\frac{\partial}{\partial c}(H)) & \text{if } y_b(\frac{\partial}{\partial c}(H) = T, \\ e/\frac{\partial}{\partial b}(\frac{\partial}{\partial c}(H)) & \text{if } y_b(\frac{\partial}{\partial c}(H) = F, \end{cases}$$

and for the case $y_c(H) = F$ we find the other two right-hand sides of (4), thus providing a prototypical example of the soundness of axiom CP4 of CP. Without (further) proof we state the following result.

Theorem 1 (Soundness)**.** *If for propositions P and Q, $\mathrm{CP} \vdash P = Q$, then for all free reactive valuations H, $P/H = Q/H$.*

5 Reactive valuation varieties

We discuss some specific equivalences and congruences generated by reactive valuations. The set of RVAs that satisfy a certain collection of equations over $\Sigma_{ReVal}(A)$ is called a *reactive valuation variety*. We distinguish the following varieties:

1. The variety of RVAs with *free reactive* valuations: no further RVA-equations than those defined in Section 4.
2. The variety of RVAs with *repetition proof* valuations: all RVAs that satisfy the equations

   $$y_a(x) = y_a(\tfrac{\partial}{\partial a}(x))$$

   for all $a \in A$. Note that there are $|A|$ such equations.
3. The variety of RVAs with *contractive* valuations: all RVAs that satisfy the equations

   $$\tfrac{\partial}{\partial a}(\tfrac{\partial}{\partial a}(x)) = \tfrac{\partial}{\partial a}(x)$$

   for all $a \in A$, and those for repetition proof valuations. Thus this is a *subvariety* of the one defined in 2 above.

4. The variety of RVAs with *static* valuations: all RVAs that satisfy the equations

$$y_a(\tfrac{\partial}{\partial b}x) = y_a(x)$$

for all $a, b \in A$.

Definition 2. *Let K be a variety of reactive valuation algebras over A. Then propositions P and Q are K-**equivalent**, notation:*

$$P \equiv_K Q.$$

*if for all $\mathbb{A} \in K$ and $H \in \mathbb{A}$, $P/H = Q/H$. Propositions P and Q are K-**congruent**, notation*

$$P =_K Q$$

if $=_K$ is the largest congruence contained in $\equiv_K$.

So, by the varieties defined thus far we distinguish four types of K-equivalence and K-congruence: free reactive, repetition proof, contractive, and static. We use the following abbreviations for these cases:

$$K = fr,\ rp,\ cr,\ st,$$

respectively. A convenient auxiliary notation in comparing these equivalences and congruences concerns the valuation of strings: given a RVA, say $\mathbb{A}$, a reactive valuation $H \in \mathbb{A}$ can be associated with a function $H_f : A^+ \to \mathbb{B}$ by defining

$$H_f(a) = y_a(H) \quad \text{and} \quad H_f(a\sigma) = (\tfrac{\partial}{\partial a}(H))_f(\sigma).$$

Proposition 2. *The inclusions $\equiv_{fr} \subseteq \equiv_{rp} \subseteq \equiv_{cr} \subseteq \equiv_{st}$, and $=_K \subseteq \equiv_K$ for $K \in \{fr,\ rp,\ cr\}$ are all proper.*

Proof. First, $a \equiv_{rp} a \lhd a \rhd F$, but $\equiv_{fr}$ does not hold in this case as is witnessed by an RVA with element H with $H_f(a) = T$ and $H_f(aa) = F$ (yielding $a/H = T$ and $(a \lhd a \rhd F)/H = F$).

Then, $b \lhd a \rhd F \equiv_{cr} b \lhd (a \lhd a \rhd F) \rhd F$, but $\equiv_{rp}$ does not hold by the RVA with element H with $H_f(a) = H_f(ab) = T$ and $H_f(aab) = F$.

Furthermore, $a \equiv_{st} a \lhd b \rhd a$ (distinguish all possible cases), but $\equiv_{cr}$ does not hold as is witnessed by the RVA with element H with $H_f(a) = H_f(b) = T$ and $H_f(ba) = F$ (yielding $a/H = T$ and $(a \lhd b \rhd a)/H = F$).

Finally, for $K \in \{fr,\ rp,\ cr\}$, $T \equiv_K T \lhd a \rhd T$, but $b \lhd T \rhd T \not\equiv_K b \lhd (T \lhd a \rhd T) \rhd T$ as is witnessed by the RVA with element H with $H_f(a) = H_f(b) = T$ and $H_f(ab) = F$. □

We continue by showing that CP axiomatizes free reactive valuation congruence:

Theorem 2 (Completeness)**.** *If $P =_{fr} Q$ then* $\mathrm{CP} \vdash P = Q$.

Proof. By soundness and Proposition 1 we may assume that P and Q are basic forms. For basic forms, $=_{fr}$ equality implies syntactic equality. □

Claim 1. *Repetition proof valuation congruence $=_{rp}$ is axiomatized by the axioms in* CP *(see Table 1) and these axiom schemes ($a \in A$):*

$$\text{(CPrp1)} \qquad (x \lhd a \rhd y) \lhd a \rhd z = (x \lhd a \rhd x) \lhd a \rhd z,$$
$$\text{(CPrp2)} \qquad x \lhd a \rhd (y \lhd a \rhd z) = x \lhd a \rhd (z \lhd a \rhd z).$$

Proof sketch. Soundness: recall repetition proof valuation equivalence is defined by

$$y_a(x) = y_a(\tfrac{\partial}{\partial a}(x))$$

for all $a \in A$. We check axiom CPrp2:

$$\begin{aligned}(P \lhd a \rhd (Q \lhd a \rhd R))/H &= \begin{cases} P/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = T, \\ (Q \lhd a \rhd R)/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = F, \end{cases} \\ &= \begin{cases} P/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = T, \\ R/\frac{\partial}{\partial a}(\frac{\partial}{\partial a}H)) & \text{if } y_a(H) = F = y_a(\frac{\partial}{\partial a}(H), \end{cases} \\ &= (P \lhd a \rhd (R \lhd a \rhd R))/H.\end{aligned}$$

Completeness: we omit a precise proof, but we note that the axiom schemes CPrp1 and CPrp2 exactly imply the definition of repetition proof valuation equivalence (for $y_a(H) = T$ and $y_a(H) = F$, respectively). □

Claim 2. *Contractive valuation congruence $=_{cr}$ is axiomatized by the axioms in* CP *(see Table 1) and these axiom schemes ($a \in A$):*

$$\text{(CPcr1)} \qquad (x \lhd a \rhd y) \lhd a \rhd z = x \lhd a \rhd z,$$
$$\text{(CPcr2)} \qquad x \lhd a \rhd (y \lhd a \rhd z) = x \lhd a \rhd z.$$

These schemes contract for each $a \in A$ respectively the T-case and the F-case, and immediately imply CPrp1 and CPrp2.

Proof sketch. Soundness follows from the definition of contractive valuation equivalence, i.e.,

$$\tfrac{\partial}{\partial a}(\tfrac{\partial}{\partial a}(x)) = \tfrac{\partial}{\partial a}(x) \qquad \text{and} \qquad y_a(x) = y_a(\tfrac{\partial}{\partial a}(x))$$

for all $a \in A$. We check axiom scheme CPcr1:

$$\begin{aligned}((P \lhd a \rhd Q) \lhd a \rhd R)/H &= \begin{cases} (P \lhd a \rhd Q)/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = T, \\ R/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = F, \end{cases} \\ &= \begin{cases} P/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = T = y_a(\frac{\partial}{\partial a}(H)), \\ R/\frac{\partial}{\partial a}(H) & \text{if } y_a(H) = F, \end{cases} \\ &= (P \lhd a \rhd R)/H.\end{aligned}$$

Completeness: again we omit a precise proof, but we note that the axiom schemes CPcr1 and CPcr2 exactly imply the definition of contractive valuation equivalence (for $y_a(H) = T$ and $y_a(H) = F$, respectively). □

Theorem 3 (Hoare [18])**.** *Static valuation congruence $=_{st}$ is axiomatized by the axioms in* CP *(see Table 1) and these axioms:*

$$\begin{aligned} &\text{(CPstat)} && (x \lhd y \rhd z) \lhd u \rhd v = (x \lhd u \rhd v) \lhd y \rhd (z \lhd u \rhd v), \\ &\text{(CPcontr)} && (x \lhd y \rhd z) \lhd y \rhd u = x \lhd y \rhd u. \end{aligned}$$

We write CP_{st} *for this set of axioms.*

Note that the symmetric variants of the axioms CPstat and CPcontr, say

$$\begin{aligned} &\text{(CPstat}') && x \lhd y \rhd (z \lhd u \rhd v) = (x \lhd y \rhd z) \lhd u \rhd (x \lhd y \rhd v), \\ &\text{(CPcontr}') && x \lhd y \rhd (z \lhd y \rhd u) = x \lhd y \rhd u, \end{aligned}$$

easily follow with identity (2), i.e., $y \lhd x \rhd z = z \lhd (F \lhd x \rhd T) \rhd y$, which is even valid in reactive valuation congruence. Thus, the axiomatization of static valuation congruence is obtained from CP by adding the axiom CPstat that prescribes for a nested conditional composition how the order of the first and a second central condition can be changed, and a generalization of the CPcr-axioms that prescribes contraction for terms (instead of atoms). Moreover, in CP_{st} it can be derived that

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

thus any 'and-then' prefix can be added to (or left out from) a proposition while preserving static valuation congruence, in particular $x \lhd x \rhd x = x \circ x = x$.

Proof of Theorem 3. Soundness follows from the definition of static valuation congruence, i.e., all RVAs that satisfy the equations

$$y_a(\tfrac{\partial}{\partial b} x) = y_a(x)$$

for all $a, b \in A$. These equations imply that for all P, Q and H,

$$P/H = P/\tfrac{\partial}{\partial Q}(H).$$

As a consequence, the validity of axioms CPstat and CPcontr follows from simple case distinctions. Furthermore, Hoare showed in [18] that CP_{st} is complete for static valuation congruence. □

As an aside, we note that the axioms CPcontr and CPcontr$'$ immediately imply CPcr1 and CPcr2, and conversely, that for y ranging over closed terms, these axioms are derivable

from CP+CPstat+CPcr1+CPcr2 (by induction on basic forms), which proves completeness for closed equations of this particular group of axioms. For an idea of a direct proof, observe that under static valuation congruence each proposition can be rewritten into the following special type of basic form: assume the atomic propositions in A are ordered $a_1, ..., a_n$, and consider the full binary tree with at level i only occurrences of atom a_i (there are 2^{i-1} such occurrences), and at level $n+1$ only leaves that are either T or F (there are 2^n such leaves). Then each series of leaves represents one of the possible equivalence classes, and the axioms in CP+CPstat+CPcr1+CPcr2 are sufficient to rewrite each proposition into such a special basic form. Hence, completeness for closed equations follows.

6 Satisfiability, homogeneity, and complexity

In this section we briefly consider some complexity issues. Given a variety K of reactive valuation algebras, a proposition is *satisfiable* with respect to K if for some non-trivial $\mathbb{A} \in K$ (T_{RV} and F_{RV} are different) there exists a reactive valuation $H \in \mathbb{A}$ such that

$$P/H = T.$$

We write

$$\mathbf{SAT}_K(P)$$

if P is satisfiable. We say that P is *falsifiable* with respect to K, notation

$$\mathbf{FAS}_K(P),$$

if and only if $\mathbf{SAT}_K(F \triangleleft P \triangleright T)$. This is the case if and only if a valuation $H \in \mathbb{A} \in K$ exists with $P/H = F$. We leave out K in $\mathbf{SAT}_K$ and $\mathbf{FAS}_K$ if it is clear from the context which variety K is meant.

Let $U \subseteq A$ and $\mathbb{A}$ be some reactive valuation algebra. Then $\mathbb{A}$ is *U-homogeneous* if for all $a, b \in U$, the following equation is satisfied in $\mathbb{A}$:

$$y_a(x) = y_a(\tfrac{\partial}{\partial b}(x)).$$

A variety K of RVAs is *U-homogeneous* if each $\mathbb{A} \in K$ is U-homogeneous.

Let P be a partition of A. Then some RVA $\mathbb{A}$ is *P-homogeneous* if it is *U-homogeneous* for each part U in P, and a variety K of RVAs is *P-homogeneous* if it is U-homogeneous for each part U in P. We notice the following facts without proof.

- The collection of P-homogeneous RVAs is a variety, axiomatized by the union of the equations for each part U in P.
- The variety of RVAs with static valuations is the variety of P-homogeneous RVAs with P the trivial partition $P(A) = A$.
- The variety of RVAs with repetition proof valuations is found by taking the (other trivial) partition $P(A) = \bigcup_{a \in A} \{a\}$.

Partitions give rise to equivalences and congruences on propositions in an obvious manner. Given a partition P of A, two propositions are P-equivalent if their value is equal on all valuations in all P-homogeneous RVAs. P-congruence is the largest congruence contained in P-equivalence. If P' refines P then P'-equivalence implies P-equivalence and P'-congruence implies P-congruence.

If A has cardinality n, it is plausible to formulate algorithmic versions of P-satisfiability as follows (similar definitions can be given for P-equivalence and P-congruence):

- P-satisfiability is in P (that is: polynomial time computable) if there is an algorithm ϕ which for a proposition Q of length ℓ_Q and a family E of equations in $\Sigma_{ReVal}(A)$ with size ℓ_E decides P-satisfiability in time polynomial in n, ℓ_Q and ℓ_E.
- P-satisfiability is in NP (that is: non-deterministic polynomial time computable) if there is a non-deterministic algorithm ϕ which for a proposition Q of length ℓ_Q and a family E of equations in $\Sigma_{ReVal}(A)$ with size ℓ_E decides P-satisfiability in time polynomial in n, ℓ_Q and ℓ_E.

Without giving the relatively simple proofs we mention these facts:

- For the trivial partition P with a single part containing all atoms, P-satisfiability is NP-complete.
- For the other trivial partition having singleton parts only, P-satisfiability is in P.
- For each partition P, P-satisfiability is in NP.

This leaves at least the following questions open:

- Is P-congruence reducible in P-time to P-equivalence?
- Is P-equivalence reducible in P-time to P-satisfiability?
- For which partitions P-satisfiability can be proved in P?

It follows easily that satisfiability under reactive valuation congruence is in P, because in this case **SAT** and **FAS** can be defined simultaneously in an inductive manner: let $a \in A$ and write $\neg\mathbf{SAT}(P)$ to express that $\mathbf{SAT}(P)$ does not hold, and similar for **FAS**, then

$\mathbf{SAT}(T)$,	$\neg\mathbf{FAS}(T)$,
$\neg\mathbf{SAT}(F)$,	$\mathbf{FAS}(F)$,
$\mathbf{SAT}(a)$,	$\mathbf{FAS}(a)$,

$$\neg T = F \tag{7}$$
$$\neg F = T \tag{8}$$
$$\neg\neg x = x \tag{9}$$
$$\neg(x \lhd y \rhd z) = \neg x \lhd y \rhd \neg z \tag{10}$$
$$x \lhd \neg y \rhd z = z \lhd y \rhd x \tag{11}$$

Table 2: Some immediate consequences of the set of axioms CP and equation (6)

and

$$\mathbf{SAT}(P \lhd Q \rhd R) \quad \text{if} \quad \begin{cases} \mathbf{SAT}(Q) \text{ and } \mathbf{SAT}(P), \\ \text{or} \\ \mathbf{FAS}(Q) \text{ and } \mathbf{SAT}(R), \end{cases}$$

$$\mathbf{FAS}(P \lhd Q \rhd R) \quad \text{if} \quad \begin{cases} \mathbf{SAT}(Q) \text{ and } \mathbf{FAS}(P), \\ \text{or} \\ \mathbf{FAS}(Q) \text{ and } \mathbf{FAS}(R). \end{cases}$$

Hence, for reactive valuation congruence both $\mathbf{SAT}(P)$ and $\mathbf{FAS}(P)$ are computable in polynomial time.

7 Adding negation and definable connectives

In this section we formally add negation and various definable connectives to CP. As stated earlier (see identity (2)), negation $\neg x$ can be defined as follows:

$$\neg x = F \lhd x \rhd T \tag{6}$$

The derivable identities in Table 2 play a role in the derivable connectives that we discuss below. They can be derived as follows:

(7) follows from $\neg T = F \lhd T \rhd T = F$,

(8) follows in a similar way,

(9) follows from $\neg\neg x = F \lhd (F \lhd x \rhd T) \rhd T = (F \lhd F \rhd T) \lhd x \rhd (F \lhd T \rhd T) = T \lhd x \rhd F = x$,

(10) follows in a similar way,

(11) follows from $x \lhd \neg y \rhd z = (z \lhd F \rhd x) \lhd \neg y \rhd (z \lhd T \rhd x) = z \lhd (F \lhd \neg y \rhd T) \rhd x = z \lhd y \rhd x$.

$x \mathbin{\overset{\circ}{\wedge}} y = y \lhd x \rhd F$	$x \circ\!\!\rightarrow y = y \lhd x \rhd T$
$x \mathbin{\wedge_\circ} y = x \lhd y \rhd F$	$x \rightarrow\!\!\circ\, y = T \lhd y \rhd \neg x$
$x \mathbin{\overset{\circ}{\vee}} y = T \lhd x \rhd y$	$x \circ\!\!\leftrightarrow y = y \lhd x \rhd \neg y$
$x \mathbin{\vee^\circ} y = T \lhd y \rhd x$	$x \leftrightarrow\!\!\circ\, y = x \lhd y \rhd \neg x$

Table 3: Defining equations for derived connectives

A definable (binary) connective already introduced is the *and then* operator $\circ$ with defining equation $x \circ y = y \lhd x \rhd y$. Furthermore, following [4] we write

$$\overset{\circ}{\wedge}$$

for *left-sequential* conjunction, i.e., a conjunction that first evaluates its lefthand argument and only after that is found T carries on with evaluating its second argument (the small circle indicates which argument is evaluated first). Similar notations are used for other sequential connectives. We provide defining equations for a number of derived connectives in Table 3.

The operators $\overset{\circ}{\wedge}$ and left-sequential disjunction $\overset{\circ}{\vee}$ are associative and the dual of each other, and so are their right-sequential counterparts. For $\overset{\circ}{\wedge}$ a proof of this is as follows:

$$\begin{aligned}(x \mathbin{\overset{\circ}{\wedge}} y) \mathbin{\overset{\circ}{\wedge}} z &= z \lhd (y \lhd x \rhd F) \rhd F \\ &= (z \lhd y \rhd F) \lhd x \rhd (z \lhd F \rhd F) \\ &= (y \mathbin{\overset{\circ}{\wedge}} z) \lhd x \rhd F \\ &= x \mathbin{\overset{\circ}{\wedge}} (y \mathbin{\overset{\circ}{\wedge}} z),\end{aligned}$$

and

$$\begin{aligned}\neg(x \mathbin{\overset{\circ}{\wedge}} y) &= F \lhd (y \lhd x \rhd F) \rhd T \\ &= (F \lhd y \rhd T) \lhd x \rhd (F \lhd F \rhd T) \\ &= \neg y \lhd x \rhd T \\ &= T \lhd \neg x \rhd \neg y \\ &= \neg x \mathbin{\overset{\circ}{\vee}} \neg y.\end{aligned}$$

Furthermore, note that $T \mathbin{\overset{\circ}{\wedge}} x = x$ and $x \mathbin{\overset{\circ}{\wedge}} T = x$, and $F \mathbin{\overset{\circ}{\vee}} x = x$ and $x \mathbin{\overset{\circ}{\vee}} F = x$.

Of course, *distributivity*, as in $(x \mathbin{\overset{\circ}{\wedge}} y) \mathbin{\overset{\circ}{\vee}} z = (x \mathbin{\overset{\circ}{\vee}} z) \mathbin{\overset{\circ}{\wedge}} (y \mathbin{\overset{\circ}{\vee}} z)$ is not valid in free reactive valuation congruence: it changes the order of evaluation and in the right-hand expression z can be evaluated twice. It is also obvious that both sequential versions of *absorption*, one of which reads

$$x = x \mathbin{\wedge_\circ} (x \mathbin{\vee^\circ} y),$$

are not valid. Furthermore, it is not difficult to prove in CP that $\circ\!\!\leftrightarrow$ and $\leftrightarrow\!\!\circ$ (i.e., the two sequential versions of bi-implication defined in Table 3) are also associative, and that $\circ\!\!\rightarrow$

and $\multimap$ are not associative, but satisfy the sequential versions of the common definition of implication:

$$x \circ\!\!\rightarrow y = \neg x \mathbin{\overset{\circ}{\vee}} y \quad \text{and} \quad x \rightarrow\!\!\circ\, y = \neg x \mathbin{\overset{\circ}{\vee}} y.$$

From now on we assume that $\mathbb{P}$ stands for the set of propositions defined over the extension of $\Sigma_{\mathrm{CP}}(A)$ with the "and then" operator $\circ$, negation and all derived connectives introduced in this section, and we adopt their defining equations. Of course, it remains the case that each proposition in $\mathbb{P}$ has a unique basic form (cf. Lemma 1).

Concerning the example of an active proposition sketched in Example (1) in Section 2:

$$\textit{look-left-and-check} \mathbin{\overset{\circ}{\wedge}} \textit{look-right-and-check} \mathbin{\overset{\circ}{\wedge}} \textit{look-left-and-check}$$

indeed precisely models part of the processing of a pedestrian planning to cross a road with two-way traffic driving on the right.

We end this section with a short comment on static valuation congruence. In this semantics, all of $\overset{\circ}{\wedge}$, $\overset{\circ}{\vee}$, $\wedge_\circ$ and $\overset{\circ}{\vee}$ are commutative and idempotent. For example, we derive with axiom CPstat that

$$x \mathbin{\overset{\circ}{\wedge}} y = y \lhd x \rhd F = (T \lhd y \rhd F) \lhd x \rhd F = (T \lhd x \rhd F) \lhd y \rhd (F \lhd x \rhd F) = x \lhd y \rhd F = y \mathbin{\overset{\circ}{\wedge}} x,$$

and with axiom CPcontr and its symmetric counterpart CPcontr$'$,

$$x \mathbin{\overset{\circ}{\vee}} x = T \lhd x \rhd x = T \lhd x \rhd (T \lhd x \rhd F) = T \lhd x \rhd F = x,$$
$$x \mathbin{\overset{\circ}{\wedge}} x = x \lhd x \rhd F = (T \lhd x \rhd x) \lhd x \rhd F = T \lhd x \rhd F = x.$$

As a consequence, the sequential notation of these connectives is not meaningful in this case, and distributivity and absorption hold in static valuation congruence.

8 Expressiveness

In this section we show that taking active propositions and reactive valuations as a point of departure implies that a ternary connective is needed for functional completeness.

We define the *depth* of a proposition $P \in \mathbb{P}$, notation $\mathit{depth}(P)$, as follows ($a \in A$):

$$\mathit{depth}(T) = 0, \tag{12}$$
$$\mathit{depth}(F) = 0, \tag{13}$$
$$\mathit{depth}(x \lhd a \rhd y) = 1 + \max(\mathit{depth}(x), \mathit{depth}(y)). \tag{14}$$

By structural induction on basic forms, it follows that $\mathit{depth}(P)$ equals the maximum number of atomic propositions that can be used in an evaluation of P. By Lemma 1 and Proposition 1 it follows that for propositions P and Q,

$$\mathrm{CP} \vdash P = Q \quad \Rightarrow \quad \mathit{depth}(P) = \mathit{depth}(Q). \tag{15}$$

We will use these properties to prove the mentioned result.

Definition 3. *Let t range over $\Sigma_{\mathrm{CP}}(A)$ terms and let X be an infinite set of variables. Then t is an* ***open basic form*** *if*

$$t ::= T \mid F \mid t_1 \lhd x \rhd t_2$$

with $x \in X$, and t_1 and t_2 open basic forms.

A binary operator defined by $f(x, y) = t$ is ***properly binary*** *if there exists an open basic form u such that $\mathrm{CP} \vdash t = u$ and u contains both variables x and y.*

Note that for open basic forms, provable equality in CP and syntactic equality coincide (cf. Proposition 1). As an example of Definition 3,

$$f(x, y) = F \lhd (T \lhd x \rhd T) \rhd y$$

is not properly binary because $f(x, y) = F \lhd x \rhd F$, while $g(x, y) = F \lhd (T \lhd x \rhd F) \rhd y$ is properly binary because $g(x, y) = F \lhd x \rhd (T \lhd y \rhd F)$. Also, all derived connectives discussed in Section 7 and the "and then" operator $\circ$ are properly binary, as can be easily checked. Definition 3 immediately implies that for any properly binary operator f and all $a, b \in A$,

$$\mathit{depth}(f(a, b)) \geq 2.$$

Note that a properly binary operator is not definable as a unary or nullary operator. We state the following fact without proof:

Lemma 2. *Each unary operator that preserves the depth of its argument is definable as a properly binary operator with one of its arguments T or F, and there are exactly four such unary operators:*

$$f(x) = \begin{cases} x & (\text{or } T \lhd x \rhd F, \text{ thus } x \mathbin{\diamondsuit\!\!\wedge} T \text{ or } T \mathbin{\diamondsuit\!\!\wedge} x), \\ F \lhd x \rhd T & (\text{thus } \neg x \text{ or } x \circ\!\!\leftrightarrow F), \\ F \lhd x \rhd F & (\text{thus } x \circ F \text{ or } x \mathbin{\diamondsuit\!\!\wedge} F), \\ T \lhd x \rhd T & (\text{thus } x \circ T \text{ or } x \mathbin{\diamondsuit\!\!\vee} T). \end{cases}$$

We call these unary operators ***flat unary operators****.*

We now come to the main result of this section:

Theorem 4. *In* CP*, conditional composition can not be expressed in terms of any collection of binary operators each of which are themselves expressible over $\Sigma_{\mathrm{CP}}(A)$.*

Proof. By Lemmas 1 and 2 it suffices to restrict to properly binary operators. Given two such operators f_1 and f_2, it easily follows that for all $a, b, c \in A$,

$$\mathit{depth}(f_1(a, f_2(b, c))) \geq 3,$$
$$\mathit{depth}(f_2(f_1(a, b), c)) \geq 3,$$

and similar for all symmetric variants f_i' (i.e., $f_i'(x, y) = f_i(y, x)$).

Consider $P = b \lhd a \rhd c$ with a, b and c different atoms. Because all of a, b and c can be used in a valuation of P, a minimal expression of P with properly binary operators requires the application of at least two such operators, so has depth of at least 3. Since $\mathit{depth}(P) = \mathit{depth}((T \lhd b \rhd F) \lhd a \rhd (T \lhd c \rhd F)) = 2$, it follows by equation (15) that P and hence conditional composition can not be expressed using definable binary operators only. □

We end this section with a comment on "connectives" in the setting of reactive valuation semantics. A *flat connective* is a properly binary operator f with defining equation

$$f(x, y) = t(x, y)$$

for $t(x, y)$ a term over $\Sigma_{\mathrm{CP}}(A)$ such that for $B \in \{T, F\}$ the open basic forms of all of

$$t(x, B) \quad \text{and} \quad t(B, x)$$

have *at most* one occurrence of x and contain no other variables (so, these basic open forms are either T or F, or equal one of the four flat unary operators $f(x)$ mentioned in Lemma 2). The idea here is that for all possible evaluations of $f(a, b)$ with $a, b \in A$, each of a and b is evaluated at most once and one of them is evaluated at least once. It follows immediately that $\mathit{depth}(f(a, b)) = 2$. The derived connectives defined in Table 3 and the "and then" operator $\circ$ are all flat.

Obviously, compositions of flat connectives need not be flat, for example $f(x, y) = x \mathbin{\diagup\!\!\!\wedge} (y \mathbin{\diagup\!\!\!\wedge} x)$ which yields $\mathit{depth}(f(a, b)) = 3$. More interestingly, there exist properly definable connectives $f(x, y)$ with $\mathit{depth}(f(a, b)) = 2$ that can not be defined as a composition of flat connectives. An example of such a connective is

$$x \,\square\, y = x \lhd y \rhd y.$$

We here only sketch a proof of the non-definability mentioned. Suppose the converse is true. Let $B_i \in \{T, F\}$ and assume that flat connectives are defined by open basic forms $t_j(x, y)$. It follows by an argument on depth that $x \,\square\, y$ must be definable by a term of the form

$$t_3(t_1(x, B_1), t_2(y, B_2))$$

with $t_1(x, B_1)$ and $t_2(y, B_2)$ both flat unary operators. But all such forms again yield a flat connective, and thus a contradiction is obtained.

Including $\square$, there exist 64 different,[1] non-flat properly definable operators $f(x, y)$ in CP with the property that $\mathit{depth}(f(a, b)) = 2$. This follows from the fact that the definitions of these must be provably equal to the form

$$f(x) \lhd y \rhd g(y) \quad \text{or} \quad g(y) \lhd y \rhd f(x)$$

with f and g one the four flat unary operators mentioned in Lemma 2, or a symmetric variant thereof (exchange x and y). None of these connecives is definable as a composition of flat connectives (cf. the preceding proof sketch) and all of these are not associative.

[1] $\square_1$ and $\square_2$ are 'different' if for $a, b \in A$ with $a \neq b$, $a \mathbin{\square_1} b \neq_{fr} a \mathbin{\square_2} b$.

It easily follows that also 64 different flat connectives can be defined in CP: first, the defining $\Sigma_{\mathrm{CP}}(A)$-term of a flat connective can always be rewritten to its open basic form, so that the central condition is a single variable, say y. Using the four flat unary operators $f(x)$ mentioned in Lemma 2, each of

$$f(x) \lhd y \rhd T \quad \text{and} \quad f(x) \lhd y \rhd F$$

defines 4 different operators, and swapping the outer arguments yields 8 more. The form

$$f(x) \lhd y \rhd g(x)$$

with also $g(x)$ a flat unary operator defines another 16 flat connectives. Including all symmetric variants (exchange x and y), this adds up to 64 different flat connectives. Many of these are non-associative (for example, left-sequential implication $\circ\!\!\rightarrow$ as defined in Table 3).

9 Three-valued logic and reactive valuations

In this short section we demonstrate that propositional reasoning with a truth value *undefined* can be captured in (two-valued) proposition algebra with reactive valuations. We define a three-valued propositional logic $\mathbb{L}$ with next to T and F a truth value $*$ that represents the value *undefined* in the spirit of McCarthy's seminal work [22].

Let $\mathbb{C} = \{T, F, *\}$. Propositions in $\mathbb{L}$ are the constants in $\mathbb{C}$ and the set A of atomic propositions, and conditional compositions of these. Furthermore, all identities provable in CP extended with the axiom

$$x \lhd * \rhd y = *$$

are valid in $\mathbb{L}$, for example $F \lhd (a \lhd b \rhd F) \rhd * = (F \lhd a \rhd *) \lhd b \rhd *$. Let $H : A \to \mathbb{C}$ be a function. We define the evaluation of a proposition $\phi \in \mathbb{L}$ according to H, notation

$$\phi \between_3 H,$$

as follows, where we further call H an $\mathbb{L}$-*valuation*:

1. For all $C \in \mathbb{C}$ and $\mathbb{L}$-valuations H, $C \between_3 H = C$,
2. Each $\mathbb{L}$-valuation H distributes over conditional composition, thus

$$(P \lhd Q \rhd R) \between_3 H = (P \between_3 H) \lhd (Q \between_3 H) \rhd (R \between_3 H).$$

As an example, $(F \lhd (a \lhd b \rhd F) \rhd *) \between_3 H = F$ if $H(a) = H(b) = T$, and $*$ in all other cases.

We now show how two-valued reactive valuations can be used to capture the three-valued semantics of $\mathbb{L}$. Assume ψ is a proposition in $\mathbb{L}$ that is $*$-free and define

$$\psi' = \mathit{init} \circ \psi \circ \mathit{corr},$$

where *init* and *corr* are fresh atomic propositions (or, at least, fresh wrt. ψ).

Given an $\mathbb{L}$-valuation H for ψ we shall define a reactive valuation H' in the two-valued setting of proposition algebra that satisfies

$$\psi'/H' = T \iff \psi /_3 H \in \{T, F\},$$

and ψ is undefined according to H if, and only if, $\psi'/H' = F$. The modified valuation H' of H is defined as the reactive valuation that

(*i*) replies T to *init* and has the side effect that the next *corr* query gets reply T,

(*ii*) replies T or F to an atom in ψ whenever H does so, and

(*iii*) replies F to an atom in ψ if H replies $*$ with the side-effect that the first reply to *corr* is F, in this way overruling earlier side effects that concern the evaluation of *corr* (the reply T would also be correct, the crucial point here is the mentioned side-effect).

Observe that if $\psi'/H' = T$ then $\psi /_3 H = \psi/H'$. We conclude that propositional reasoning in the three-valued "error logic" $\mathbb{L}$ can be captured in the setting of two-valued reactive valuation semantics, which hence provides another motivation for proposition algebra.

Of course, one might prefer to reason directly in $\mathbb{L}$. For this purpose we add the truth value $*$ to the signature $\Sigma_{\mathrm{CP}}(A)$ (negation and the derived connectives as defined in Section 7 can also be added without any problem) and we extend CP with the following two axioms:

$$\text{(CP5)} \qquad x \lhd y \rhd (z \lhd u \rhd (v \lhd y \rhd w)) = x \lhd y \rhd (z \lhd u \rhd w),$$

$$\text{(CP6)} \qquad x \lhd * \rhd y = *.$$

We write CPu for the axioms CP1–CP6, and we state here a completeness result. A further discussion of $\mathbb{L}$ and CPu, and a proof of this result can be found in Appendix A.

Theorem 5. *The axioms of* CPu *characterize* $\mathbb{L}$*: for three-valued propositions* P *and* Q*,* $\mathrm{CPu} \vdash P = Q$ *if, and only if,* $\mathbb{L} \models P = Q$*, i.e., for all* $\mathbb{L}$*-valuations* H*,* $P /_3 H = Q /_3 H$*.*

10 Projections and the projective limit model

In this section we introduce the *projective limit model* $\mathbb{A}^\infty$ for defining potentially infinite active propositions. Let $\mathbb{N}^+$ denote $\mathbb{N} \setminus \{0\}$. We first define a so-called *projection* operator

$$\pi : \mathbb{N}^+ \times \mathbb{P} \to \mathbb{P},$$

which will be used to finitely approximate every proposition in $\mathbb{P}$. We further write

$$\pi_n(P)$$

instead of $\pi(n, P)$. The defining equations for the π_n-operators are these ($n \in \mathbb{N}^+$):

$$\pi_n(T) = T, \tag{16}$$

$$\pi_n(F) = F, \tag{17}$$

$$\pi_1(x \lhd a \rhd y) = a, \tag{18}$$

$$\pi_{n+1}(x \lhd a \rhd y) = \pi_n(x) \lhd a \rhd \pi_n(y), \tag{19}$$

for all $a \in A$. We write PR for this set of equations.

By Lemma 1 projections are well-defined on $\mathbb{P}$. We state without proof that CP + PR is a conservative extension of CP and mention the following derivable identities in CP + PR for $a \in A$ and $n, k \in \mathbb{N}^+$:

$$\begin{aligned} \pi_n(a) &= \pi_n(T \lhd a \rhd F) = a, \\ \pi_{n+1}(a \circ x) &= a \circ \pi_n(x), \\ \pi_n(\pi_k(x)) &= \pi_{\min(n,k)}(x). \end{aligned}$$

Obviously, for each proposition P there exists $n \in \mathbb{N}^+$ such that for all $j \in \mathbb{N}$,

$$\pi_{n+j}(P) = P.$$

This can be related to the depth of a proposition (see equations (12)–(14)). It is easily follows that for all $P \in \mathbb{P}$ and $n \in \mathbb{N}^+$,

- $depth(\pi_n(P)) \leq n$, and
- if $depth(P) \geq 1$, then $depth(P)$ equals the smallest n that satisfies $\pi_n(P) = P$.

The following lemma establishes how the arguments of the projection of a conditional composition can be restricted to certain projections, in particular

$$\pi_n(P \lhd Q \rhd R) = \pi_n(\pi_n(P) \lhd \pi_n(Q) \rhd \pi_n(R)), \tag{20}$$

which is a property that we will use in the definition of our projective limit model.

Lemma 3. *For all $P, Q, R \in \mathbb{P}$ and all $n \in \mathbb{N}^+$, $k, \ell, m \in \mathbb{N}$,*

$$\pi_n(P \lhd Q \rhd R) = \pi_n(\pi_{n+k}(P) \lhd \pi_{n+\ell}(Q) \rhd \pi_{n+m}(R)).$$

Proof. By Lemma 1 we may assume that Q is a basic form. We apply structural induction on Q.

If $Q = T$ or $Q = F$ we are done.

If $Q = Q_1 \lhd a \rhd Q_2$ then we proceed by induction on n. The case $n = 1$ is trivial, and

$$\begin{aligned} &\pi_{n+1}(P \lhd Q \rhd R) \\ &= \pi_{n+1}(P \lhd (Q_1 \lhd a \rhd Q_2) \rhd R) \\ &= \pi_{n+1}((P \lhd Q_1 \rhd R) \lhd a \rhd (P \lhd Q_2 \rhd R)) \\ &= \pi_n(P \lhd Q_1 \rhd R) \lhd a \rhd \pi_n(P \lhd Q_2 \rhd R) \\ &\overset{IH}{=} \pi_n(\pi_{n+k}(P) \lhd \pi_{n+\ell}(Q_1) \rhd \pi_{n+m}(R)) \lhd a \rhd \pi_n(\pi_{n+k}(P) \lhd \pi_{n+\ell}(Q_2) \rhd \pi_{n+m}(R)) \\ &= \pi_n(\pi_{n+k+1}(P) \lhd \pi_{n+\ell}(Q_1) \rhd \pi_{n+m+1}(R)) \lhd a \rhd \pi_n(\pi_{n+k+1}(P) \lhd \pi_{n+\ell}(Q_2) \rhd \pi_{n+m+1}(R)) \\ &= \pi_{n+1}((\pi_{n+k+1}(P) \lhd \pi_{n+\ell}(Q_1) \rhd \pi_{n+m+1}(R)) \lhd a \rhd (\pi_{n+k+1}(P) \lhd \pi_{n+\ell}(Q_2) \rhd \pi_{n+m+1}(R))) \\ &= \pi_{n+1}(\pi_{n+k+1}(P) \lhd (\pi_{n+\ell}(Q_1) \lhd a \rhd \pi_{n+\ell}(Q_2)) \rhd \pi_{n+m+1}(R)) \\ &= \pi_{n+1}(\pi_{n+1+k}(P) \lhd \pi_{n+1+\ell}(Q) \rhd \pi_{n+1+m}(R)). \end{aligned}$$

□

The *projective limit model* $\mathbb{A}^\infty$ is defined as follows:

- The domain of $\mathbb{A}^\infty$ is the set of *projective sequences* $(P_n)_{n\in\mathbb{N}^+}$: these are all sequences with the property that all P_n are in $\mathbb{P}$ and satisfy

  $$\pi_n(P_{n+1}) = P_n,$$

  so that they can be seen as successive projections of the same infinite proposition (observe that $\pi_n(P_n) = P_n$). We write $\mathbb{P}^\infty$ for this domain, and we further write $(P_n)_n$ instead of $(P_n)_{n\in\mathbb{N}^+}$.
- Equivalence of projective sequences in $\mathbb{A}^\infty$ is defined component-wise, thus $(P_n)_n = (Q_n)_n$ if for all n, $P_n = Q_n$.
- The constants T and F are interpreted in $\mathbb{A}^\infty$ as the projective sequences that consist solely of these respective constants.
- An atomic proposition a is interpreted in $\mathbb{A}^\infty$ as the projective sequence $(a,\ a,\ a, ...)$.
- Projection in $\mathbb{A}^\infty$ is defined component-wise, thus $\pi_k((P_n)_n) = (\pi_k(P_n))_n$.
- Conditional composition in $\mathbb{A}^\infty$ is defined using projections:

  $$(P_n)_n \triangleleft (Q_n)_n \triangleright (R_n)_n = (\pi_n(P_n \triangleleft Q_n \triangleright R_n))_n.$$

  The projections are needed if the depth of a component $P_n \triangleleft Q_n \triangleright R_n$ exceeds n. equation (20) implies that this definition indeed yields a projective sequence:

  $$\begin{aligned}\pi_n(\pi_{n+1}(P_{n+1} \triangleleft Q_{n+1} \triangleright R_{n+1})) &= \pi_n(P_{n+1} \triangleleft Q_{n+1} \triangleright R_{n+1}) \\ &= \pi_n(\pi_n(P_{n+1}) \triangleleft \pi_n(Q_{n+1}) \triangleright \pi_n(R_{n+1})) \\ &= \pi_n(P_n \triangleleft Q_n \triangleright R_n).\end{aligned}$$

The following result can be proved straightforwardly:

Theorem 6. $\mathbb{A}^\infty \models \mathrm{CP} + \mathrm{PR}$.

Furthermore, the following characterization of the interpretation of propositions in $\mathbb{P}$ follows easily by structural induction and Equation 20.

Proposition 3. *The interpretation of a proposition $P \in \mathbb{P}$ in $\mathbb{A}^\infty$ is provably equal to the projective sequence $(P_n)_n$ defined by*

$$P_n = \pi_n(P).$$

For example, the proposition $a \owedge b$ is in $\mathbb{A}^\infty$ represented by $(a,\ a \owedge b,\ a \owedge b,\ ...)$.

We extend the notion of depth to a projective sequence $(P_n)_n$ in the expected way: if $\mathit{depth}(P_1) = 0$, then $\mathit{depth}((P_n)_n) = 0$, otherwise it is defined as the smallest k for which $\pi_k((P_n)_n) = (P_n)_n$, or infinite if such k does not exist. Indeed, the projective limit model $\mathbb{A}^\infty$ contains elements that are not the interpretation of finite propositions in $\mathbb{P}$ (in other words, elements of infinite depth). In the next section we discuss some examples.

11 Recursive specifications

In this section we discuss *recursive specifications* over $\Sigma_{\mathrm{CP}}(A)$, which provide an alternative and simple way to define active propositions in $\mathbb{A}^\infty$. We first restrict to a simple class of recursive specifications: given $\ell > 0$, a set

$$E = \{X_i = t_i \mid i = 1, ..., \ell\}$$

of equations is a *linear specification* over $\Sigma_{\mathrm{CP}}(A)$ if

$$t_i ::= T \mid F \mid X_j \lhd a_i \rhd X_k$$

for $i, j, k \in \{1, ..., \ell\}$ and $a_i \in A$. A *solution* for E in $\mathbb{A}^\infty$ is a series of propositions

$$(P_{1,n})_n, ..., (P_{\ell,n})_n$$

such that $(P_{i,n})_n$ solves the equation for X_i. In $\mathbb{A}^\infty$, solutions for linear specifications exist. This follows from the property that for each $m \in \mathbb{N}^+$, $\pi_m(X_i)$ can be computed as a proposition in $\mathbb{P}$ by replacing variables X_j by t_j sufficiently often. For example, if

$$E = \{X_1 = X_3 \lhd a \rhd X_2,\ X_2 = b \circ X_1,\ X_3 = T\}$$

we find $\pi_m(X_3) = \pi_m(T) = T$ for all $m \in \mathbb{N}^+$, and

$$\begin{aligned} \pi_1(X_2) &= \pi_1(b \circ X_1) & \qquad \pi_{m+1}(X_2) &= \pi_{m+1}(b \circ X_1) \\ &= b, & &= b \circ \pi_m(X_1), \\ \pi_1(X_1) &= \pi_1(X_3 \lhd a \rhd X_2) & \qquad \pi_{m+1}(X_1) &= \pi_{m+1}(X_3 \lhd a \rhd X_2) \\ &= a, & &= T \lhd a \rhd \pi_m(X_2), \end{aligned}$$

and we can in this way construct a projective sequence per variable. We state without proof that for a linear specification $E = \{X_i = t_i \mid i = 1, ..., \ell\}$ such sequences model *unique* solutions in $\mathbb{A}^\infty$,[2] and we write

$$\langle X_i | E \rangle$$

for the solution of X_i as defined in E. In order to reason about linearly specified propositions, we add these constants to the signature Σ_{CP}, which consequently satisfy the equations

$$\langle X_i | E \rangle = \langle t_i | E \rangle$$

where $\langle t_i | E \rangle$ is defined by replacing each X_j in t_i by $\langle X_j | E \rangle$. The proof principle introducing these identities is called the *Recursive Definition Principle* (RDP), and for linear specifications RDP is valid in the projective limit model $\mathbb{A}^\infty$.[3] As illustrated above, all solutions

[2] $\mathbb{A}^\infty$ can be turned into a metric space by defining $d((P_n)_n, (Q_n)_n) = 2^{-n}$ for n the least value with $P_n \neq Q_n$. The existence of unique solutions for linear specifications then follows from Banach's fixed point theorem; a comparable and detailed account of this fact can be found in [25].

[3] A nice and comparable account of the validity of RDP in the projective limit model for ACP is given in [1]. In that text book, a sharp distinction is made between RDP—stating that certain recursive specifications have *at least* a solution per variable—and the Recursive Specification Principle (RSP), stating that they have *at most* one solution per variable. The uniqueness of solutions per variable then follows by establishing the validity of both RDP and RSP.

satisfy

$$\langle X_i|E\rangle = (\pi_n(\langle X_i|E\rangle))_n.$$

Some examples of propositions defined by recursive specifications are these:

- For $E = \{X_1 = X_2 \triangleleft a \triangleright X_3,\ X_2 = T,\ X_3 = F\}$ we find

  $$\langle X_1|E\rangle = (a,\ a,\ a,\ ...)$$

  which in the projective limit model represents the atomic proposition a. Indeed, by RDP we find $\langle X_1|E\rangle = \langle X_2|E\rangle \triangleleft a \triangleright \langle X_3|E\rangle = T \triangleleft a \triangleright F = a$.

- For $E = \{X_1 = X_2 \triangleleft a \triangleright X_3,\ X_2 = T,\ X_3 = T\}$ we find

  $$\langle X_1|E\rangle = (a,\ a \circ T,\ a \circ T,\ a \circ T,\ ...)$$

  which in the projective limit model represents $a \circ T$. By RDP we find $\langle X_1|E\rangle = a \circ T$.

- For $E = \{X_1 = X_3 \triangleleft a \triangleright X_2,\ X_2 = b \circ X_1,\ X_3 = T\}$ as discussed above, we find

  $$\langle X_1|E\rangle = (a,\ T \triangleleft a \triangleright b,\ T \triangleleft a \triangleright b \circ a,\ T \triangleleft a \triangleright b \circ (T \triangleleft a \triangleright b),\ ...)$$

  which in the projective limit model represents an *infinite* proposition, that is, one that satisfies

  $$\pi_i(\langle X_1|E\rangle) = \pi_j(\langle X_1|E\rangle) \quad \Rightarrow \quad i = j,$$

  and thus has infinite depth. By RDP we find $\langle X_1|E\rangle = T \triangleleft a \triangleright b \circ \langle X_1|E\rangle$. We note that the infinite proposition $\langle X_1|E\rangle$ can be characterized as *while* $\neg a$ *do* b.

An example of a projective sequence that can not be defined by a linear specification, but that can be defined by the *infinite* linear specification $I = \{X_i = t_i \mid i \in \mathbb{N}^+\}$ with

$$t_i = \begin{cases} a \circ X_{i+1} & \text{if } i \text{ is prime,} \\ b \circ X_{i+1} & \text{otherwise,} \end{cases}$$

is $\langle X_1|I\rangle$, satisfying

$$\langle X_1|I\rangle = (b,\ b \circ a,\ b \circ a \circ a,\ b \circ a \circ a \circ b,\ b \circ a \circ a \circ b \circ a,\ ...).$$

Other examples of projective sequences that can not be defined by a finite linear specification are $\langle X_j|I\rangle$ for any $j > 1$.

Returning to Example (1) of an active proposition sketched in Section 2, we can be more explicit now: the recursively defined active proposition $\langle X_1|E\rangle$ with E containing

$$X_1 = X_2 \triangleleft \textit{green-light} \triangleright X_1,$$
$$X_2 = X_3 \triangleleft (\textit{look-left-and-check} \mathbin{\overset{\circ}{\wedge}} \textit{look-right-and-check} \mathbin{\overset{\circ}{\wedge}} \textit{look-left-and-check}) \triangleright X_1,$$
$$X_3 = ...$$

models in a straightforward way a slightly larger part of the processing of a pedestrian planning to cross a road with two-way traffic driving on the right.

12 Conclusions

Proposition algebra in the form of CP for active propositions with conditional composition and either or not enriched with negation and sequential connectives, is proposed as an abstract data type. Free reactive valuations provide the natural semantics for CP and these are semantically at the opposite end of static valuations. It is shown that taking conditional composition and reactive valuations as a point of departure implies that a ternary connective is needed for functional completeness; binary connectives are not sufficient. Furthermore, CP admits a meaningful and non-trivial extension to projective limits, and this constitutes the most simple case of an inverse limit construction that we can think of. We see this as the key contribution of this paper.

The potential role of proposition algebra is only touched upon by some examples. It remains a challenge to find convincing examples that require reactive valuations, and to find earlier accounts of this type of semantics for proposition logic. The basic idea of proposition algebra with its active propositions and reactive valuations can be seen as a combination of the following two ideas:

- Consider atomic propositions as events (queries) that can have a side effect in a sequential system, and take McCarthy's sequential evaluation as described in [22] to 2-valued proposition logic; this motivates reactive valuations as those that define evaluation or computation as a sequential phenomenon.
- In the resulting setting, Hoare's conditional composition as introduced in [18] is more natural than the sequential, non-commutative versions of conjunction and disjunction, and (as it appears) more expressive: a ternary connective is needed anyhow.

For conditional composition we have chosen for the notation

$$\_ \lhd \_ \rhd \_$$

from Hoare [18] in spite of the fact that our theme is technically closer to thread algebra [6] where a different notation is used, because its most well-known semantics is static valuation semantics (which is simply conventional propositional logic) for which the notation in [18] was introduced.[4] To some extent, thread algebra and proposition logic in the style of [18] are models of the same signature. A much more involved use of conditional composition can be found in [24], where the propositional fragment of Belnap's four-valued logic [2] is characterized using only conditional composition and his four constants representing these truth values.

We end with a few notes on related matters.

1. On McCarthy's conditional expressions [22]. In quite a few papers the 'lazy evaluation' semantics proposed in McCarthy's work is discussed, or taken as a point of departure. We mention a few of these in reverse chronological order:

[4] This notation was used by Hoare in his 1985 book on CSP [19] and by Hoare *et al.* in the well-known 1987 paper *Laws of Programming* [17] for expressions $P \lhd b \rhd Q$ with P and Q programs and b a Boolean expression without mention of [18] that appeared in 1985.

Hähnle states in his paper *Many-valued logic, partiality, and abstraction in formal specification languages* [16] that

> *"sequential conjunction [...] represents the idea that if the truth value can be determined after evaluation of the first argument, then the result is computed without looking at the second argument. Many programming languages contain operators that exhibit this kind of behavior".*

Furthermore, Konikowska describes in [21] a model of so-called McCarthy algebras in terms of three-valued logic, while restricting to the well-known symmetric binary connectives, and provides sound axiomatizations and representation results. This is achieved by admitting only T and F as constants in a McCarthy algebra, and distinguishing an element a as in one of four possible classes ('positive' if $a \vee x = a$, 'negative' if $a \wedge x = a$, 'defined' if $a \wedge \neg a = F$, and 'strictly undefined' if $a = \neg a$).

Finally, Bloom and Tindell discuss in their paper *Varieties of "if-then-else"* [11] various modelings of conditional composition, both with and without a truth value *undefined*, while restricting to the "redundancy law"

$$(x \lhd y \rhd z) \lhd y \rhd u = x \lhd y \rhd u,$$

a law that we called CPcontr in Section 5 and that generalizes the axiomatization of contractive valuation congruence defined in that section to an extent in which only the difference beteen T, F and *undefined* plays a decisive role.[5]

As far as we can see, none of the papers mentioned here even suggests the idea of *reactive valuation semantics*.

2. Concerning projections and the projective limit model $\mathbb{A}^\infty$ we mention that in much current research and exposition, projections are defined also for depth 0 (see, e.g., [6, 25] for thread algebra, and [15] for process algebra). However, CP does not have a natural candidate for $\pi_0(P)$ and therefore we stick to the original approach as described in [5] (and overviewed in [1]) that starts from projections with depth 1.

3. Reactive valuations were in a different form employed in accounts of process algebra with propositions: in terms of operational semantics, this involves transitions

   $$P \stackrel{a,w}{\rightarrow} Q$$

   for process expressions P and Q with a an action and w ranging over a class of valuations. In particular this approach deals with process expressions that contain propositions in the form of guarded commands, such as $\phi :\rightarrow P$ that has a transition

   $$(\phi :\rightarrow P) \stackrel{a,w}{\rightarrow} Q$$

   if $P \stackrel{a,w}{\rightarrow} Q$ and $w(\phi) = T$. For more information about this approach, see, e.g., [7, 8].

[5] At the end of Section 9 and continued in Appendix A we discuss a simple approach to conditional composition (or *if-then-else*) in a setting with the truth value *undefined*, and we consider various laws for 'contraction' or 'redundancy'.

References

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

[2] N.D. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstein (eds.), *Modern Uses of Multiple-Valued Logic*, D. Reidel, pages 8–37, 1977.

[3] J.A. Bergstra, I. Bethke, and A. Ponse. Thread algebra and risk assessment services. In C. Dimitracopoulos et. al. (eds.), *Proceedings Logic Colloquium 2005*, Cambridge Lecture Notes in Logic 28, pages 1–17, 2007.

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

[5] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(13):109–137, 1984.

[6] J.A. Bergstra and C.A. Middelburg. Thread algebra for strategic interleaving. *Formal Aspects of Computing*, 19(4):445–474, 2007.

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

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

[9] J.A. Bergstra and A. Ponse. A bypass of Cohen's impossibility result. In P.M.A. Sloot et. al. eds. *Advances in Grid Computing-EGC 2005*, LNCS 3470, pp. 1097–1106, Springer-Verlag, 2005. (Also available as Electronic report PRG0501 at `www.science.uva.nl/research/prog/publications.html`.)

[10] J.A. Bergstra and A. Ponse. Execution architectures for program algebra. *Journal of Applied Logic*, 5(1):170–192, 2007.

[11] S.L. Bloom and R. Tindell. Varieties of "if-then-else". *SIAM Journal of Computing*, 12(4):677–707, 1983

[12] D.A. Bochvar. On a three-valued logical calculus and its application to the analysis of the paradoxes of the classical extended functional calculus. (in Russian, the title was also translated as *On a 3-valued logical calculus and its application to the analysis of contradictions*). *Matematicheskii Sbornik*, 4(46):287–308, 1938. Translated from the Russian by M. Bergmann in *History and Philosophy of Logic*, 2:87–112, 1981. [6]

[13] F. Cohen. Computer viruses - theory and experiments. `http://vx.netlux.org/lib/afc01.html`, 1984. Version including some corrections and references: *Computers& Security*, 6(1): 22–35, 1987.

[6] Version of this paper including a consistency proof (translated from the Russian by M. Bergmann): D.A. Bochvar. On the consistency of a three-valued logical calculus. *Topoi*, 3(1):3–12, 1984.

[14] F. Cohen. Reply to 'Comment on "A Framework for Modelling Trojans and Computer Virus Infection"' by E. Mäkinen. *The Computer Journal*, 44(4):326–327, 2001.

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

[16] R. Hähnle. Many-valued logic, partiality, and abstraction in formal specification languages. *Logic Journal of IGPL*, 13(4):415–433, 2005.

[17] I.J. Hayes, He Jifeng, C.A.R. Hoare, C.C. Morgan, A.W. Roscoe, J.W. Sanders, I.H. Sorensen, J.M. Spivey, B.A. Sufrin. Laws of programming. *Communications of the ACM*, 3(8):672–686, 1987.

[18] C.A.R. Hoare. A couple of novelties in the propositional calculus. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 31(2):173-178, 1985.

[19] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

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

[21] B. Konikowska. McCarthy algebras: a model of McCarthy's logical calculus. *Fundamenta Informaticae*, 26(2):167–203, 1996.

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

[23] A. Ponse and M.B. van der Zwaag. Risk assessment for one-counter threads. Electronic report PRG0608, Programming Research Group, University of Amsterdam, October 2006, available at `www.science.uva.nl/research/prog/publications.html`.
Final version to appear in *Theory of Computing Systems*.

[24] A. Ponse and M.B. van der Zwaag. Belnap's logic and conditional composition. *Theoretical Computer Science*, 388(1-3):319–336, 2007.

[25] T.D. Vu. Denotational semantics for thread algebra. *Journal of Logic and Algebraic Programming*, 74, 94–111, 2008.

A An axiomatization of the three-valued logic $\mathbb{L}$

In Section 9 we defined a three-valued logic $\mathbb{L}$ and so-called $\mathbb{L}$-valuations, and the set of axioms CPu (i.e., axioms CP1–CP6) listed in Table 4. Before we turn to the completeness of CPu with respect to $\mathbb{L}$-valuations, we list some characteristics of $\mathbb{L}$ in terms of the sequential connectives and negation defined in Section 7:

1. The identities $T \mathbin{\owedge} x = x$, $F \mathbin{\owedge} x = F$ and $* \mathbin{\owedge} x = *$ are all valid in $\mathbb{L}$,
2. The associativity of $\owedge$ is valid in $\mathbb{L}$,

(CP1)	$x \lhd T \rhd y = x$
(CP2)	$x \lhd F \rhd y = y$
(CP3)	$T \lhd x \rhd F = x$
(CP4)	$x \lhd (y \lhd z \rhd u) \rhd v = (x \lhd y \rhd v) \lhd z \rhd (x \lhd u \rhd v)$
(CP5)	$x \lhd y \rhd (z \lhd u \rhd (v \lhd y \rhd w)) = x \lhd y \rhd (z \lhd u \rhd w)$
(CP6)	$x \lhd * \rhd y = *$

Table 4: The set CPu of axioms for three-valued logic $\mathbb{L}$

3. The identity $x \mathbin{\underset{\circ}{\wedge}} y \mathbin{\underset{\circ}{\wedge}} x = x \mathbin{\underset{\circ}{\wedge}} y$ is valid in $\mathbb{L}$,
4. Negation is defined in $\mathbb{L}$ by $\neg * = *$.

By the first characteristic the connective $\mathbin{\underset{\circ}{\wedge}}$ is not commutative in $\mathbb{L}$: $F \mathbin{\underset{\circ}{\wedge}} * = F$ while $* \mathbin{\underset{\circ}{\wedge}} F = *$. Furthermore, characteristic 3 implies with $y = T$ the idempotence of $\mathbin{\underset{\circ}{\wedge}}$. These four characteristics immediately reveal that $\mathbb{L}$ can be regarded as the sequential version of both Kleene's three-valued logic [20] (published in 1938) and Bochvar's three-valued logic [12] (published in Russian in 1938).

Axiom CP5 defines how the central condition y may recur in an expression. This axiom yields in combination with CP (the axioms CP1–CP4) some interesting consequences. First, CP5 has three symmetric variants, which all follow easily with identity (2), i.e., $y \lhd x \rhd z = z \lhd (F \lhd x \rhd T) \rhd y$ $(= z \lhd \neg x \rhd y)$:

$$x \lhd y \rhd ((z \lhd y \rhd u) \lhd v \rhd w) = x \lhd y \rhd (u \lhd v \rhd w), \tag{21}$$
$$(x \lhd y \rhd (z \lhd u \rhd v)) \lhd u \rhd w = (x \lhd y \rhd z) \lhd u \rhd w, \tag{22}$$
$$((x \lhd y \rhd z) \lhd u \rhd v) \lhd y \rhd w = (x \lhd u \rhd v) \lhd y \rhd w. \tag{23}$$

Furthermore, the axioms CP1–CP5 define various laws for contraction:

$$x \lhd y \rhd (v \lhd y \rhd w) = x \lhd y \rhd w, \tag{24}$$
$$x \lhd y \rhd (T \lhd u \rhd y) = x \lhd y \rhd u, \tag{25}$$
$$(x \lhd y \rhd z) \lhd y \rhd u = x \lhd y \rhd u, \tag{26}$$
$$(x \lhd y \rhd F) \lhd x \rhd z = y \lhd x \rhd z, \tag{27}$$

and

$$x \lhd y \rhd y = x \lhd y \rhd F, \tag{28}$$
$$x \lhd x \rhd y = T \lhd x \rhd y. \tag{29}$$

The latter two equations immediately imply the following very simple contraction laws:

$$x \lhd x \rhd x = x \lhd x \rhd F = T \lhd x \rhd x = x.$$

Finally, note that equation (27) implies the third characteristic of $\mathbb{L}$ mentioned above:

$$x \mathbin{\underset{\circ}{\wedge}} y \mathbin{\underset{\circ}{\wedge}} x = x \mathbin{\underset{\circ}{\wedge}} (y \mathbin{\underset{\circ}{\wedge}} x) = (x \lhd y \rhd F) \lhd x \rhd F = y \lhd x \rhd F = x \mathbin{\underset{\circ}{\wedge}} y.$$

In the remainder of this appendix we discuss our completeness result for $\mathbb{L}$, and we start by defining a useful class of propositions.

Definition 4. *In the signature of* CPu*,* ***basic forms*** *are defined by*

$$P ::= * \mid T \mid F \mid P_1 \lhd a \rhd P_2$$

with $a \in A$ and P_1, P_2 basic forms.

A proposition P is a ***non-replicating basic form*** *if P is a basic form in which no atomic proposition occurs twice.*

Proposition 4. *In CPu, each proposition is provably equal to a non-replicating basic form. For non-replicating basic forms, provable equivalence and syntactic equivalence coincide.*

Proof. First, each proposition is provably equal to a basic form (by structural induction).

Then, each basic form is provably equal to a non-replicating one. This also follows by structural induction. Modulo symmetry, the crucial case is

$$P = Q \lhd a \rhd R$$

with $R = R_1 \lhd b \rhd R_2$, $b \neq a$ and a occurs in R:

$$\begin{aligned}
P &= Q \lhd a \rhd (R_1 \lhd b \rhd R_2) \\
&= Q \lhd a \rhd ((T \lhd a \rhd R_1) \lhd b \rhd (T \lhd a \rhd R_2)) && \text{(by (21) and axiom CP5)} \\
&\overset{IH}{=} Q \lhd a \rhd ((T \lhd a \rhd R_1') \lhd b \rhd (T \lhd a \rhd R_2')) && \text{with } R_1' \text{ and } R_2' \ a\text{-free} \\
&= Q \lhd a \rhd (R_1' \lhd b \rhd R_2') && \text{(again by (21) and CP5).}
\end{aligned}$$

Finally, it is immediately clear that for non-replicating basic forms, provable and syntactic equivalence coincide. □

We end with the following result:

Theorem (Theorem 5 in Section 9)**.**
The axioms of CPu *characterize $\mathbb{L}$: for three-valued propositions P and Q,* CPu $\vdash P = Q$ *if, and only if, $\mathbb{L} \models P = Q$, i.e., for all $\mathbb{L}$-valuations H, $P /_3 H = Q /_3 H$.*

Proof. Soundness follows immediately (by some simple case distinctions).

Now assume $\mathbb{L} \models P = Q$. By soundness and Proposition 4, we may further assume that both P and Q are non-replicating basic forms. By $P /_3 H = Q /_3 H$ for all $\mathbb{L}$-evaluations H, P and Q must be syntactically equal, and thus by Proposition 4, CPu $\vdash P = Q$. □

We do not consider the question whether CPU's axioms CP1–CP6 are independent, nor the question whether CPu is ω-complete.