Modeling the interaction of computer errors by four-valued contaminating logics

Roberto Ciuni[1], Thomas Macaulay Ferguson[2], and Damian Szmuc[3]

[1] Department FISPPA, Section of Philosophy, University of Padova
roberto.ciuni@unipd.it
[2] Cycorp and Saul Kripke Center, CUNY Graduate Center
tferguson@gradcenter.cuny.edu
[3] IIF-SADAF, CONICET and Department of Philosophy, University of Buenos Aires
szmucdamian@gmail.com

Abstract. Logics based on Weak Kleene Algebra (WKA) and related structures have been recently proposed as a tool for reasoning about *flaws* in computer programs [11]. The key element of this proposal is the presence, in WKA and related structures, of a non-classical truth-value that is "contaminating" in the sense that whenever the value is assigned to a formula ϕ, any complex formula in which ϕ appears is assigned that value as well. Under the interpretation by [11], the contaminating states 'represents' occurrence of a flaw. However, since different programs and machines can interact with (or be nested into) one another, we need to account for different kind of errors, and this calls for an evaluation of systems with *multiple* contaminating values. In this paper, we make first steps toward these evaluation systems by considering the logics HYB_1 and HYB_2 by [19], whose semantic interpretation accounts for two contaminating values beside classical values 0 and 1. In particular, we provide two main formal contributions. First, we give a characterization of their relations of (mutiple-conclusion) logical consequence—that is, necessary and sufficient conditions for a set Δ of formulas to logically follow from a set Γ of formulas in HYB_1 or HYB_2. Second, we provide sound and complete sequent calculi for the two logics.

1 Introduction

Applications of logic to the topic of reasoning about computer errors date back at least to [17]. There, *critical errors* affecting *sequential* computing are considered, and the main intuition concerning such errors is that the exact step of the computation in which the error occurs determine which string of information fails to be computed. For instance, if one is computing the value of $\phi \vee \psi$ and an error occurs while computing the value of ϕ, then the computation will stop without establishing a value [1]. By contrast, if the error occurs when computing the value of ψ *and after* computing the value of ϕ—assigning, say, 1—then the computation will prove successful and assign value 1 to $\phi \vee \psi$. This kind of error, in a nutshell, suggests the need for a non-commutative disjunction (and conjunction). [1] made a progress w.r.t. [17] by providing a reasoning tool for errors in *parallel* computing, and by proposing a four-valued logic for reasoning about the interaction of the two kinds of errors. In particular, the logic for reasoning on errors in parallel computing is the Strong Kleene Logic K_3, which is in turn interpreted on the so-called (Strong) Kleene Algebra from [16].

More recently, [11] has discussed another kind of errors, which is due to failure to declare a variable in programs that are based on $C++$. An interesting consequence by the discussion in [11] is that this kind of errors are not suitably represented by structures such as Strong Kleene Algebra, or by the matrix proposed in [17]. By contrast, such errors would be better represented

by Weak Kleene Algebra (WKA). This algebra comprises a non-classical value n beside 0 and 1; such a value, in turn, *contaminates* the other, in the sense that, if it is assigned to any input of a truth function, it will be assigned to the output of the truth function, independently from the value of the other inputs.

Two non-trivial logics are based on WKA, namely systems $\mathsf{K}_3^{\mathsf{w}}$ from [4] and PWK from [15]. [11] considers the former in his computational interpretation of the third value from WKA.

The kind of errors pointed at in [11] are novel w.r.t. [1, 17], but contrary to [1], [11] does not consider the interaction of different kinds of errors. This is limiting, since a host of different errors may display the contaminating behavior of value n, or a very close behavior—see Section 4 below. This seems to call for sublogics of $\mathsf{K}_3^{\mathsf{w}}$, where each contaminating value represent a different kind of error.

In this paper, we consider two families of computer errors that, in our view, can be represented by a contaminating value. These are *code errors* involving some virtual machine, and *physical errors* involving the operating systems—see Section 4 below. In order to provide reasoning tools that can, in principle, capture the interaction of these two kinds of errors, we introduce a four-valued algebra that is somehow inspired to WKA; we call this further structure 'hybrid algebra', since it hybridizes two different contaminating values. Also, we introduce two logics that are interpreted on the hybrid algebra, namely the four-valued systems HYB_1 and HYB_2. These in turn provides reasoning tools that can account for the interaction of the two kinds of errors above.

Given the role played by contaminating values in the systems we present here, we call $\mathsf{K}_3^{\mathsf{w}}$, PWK, HYB_1 and HYB_2 'contaminating logics'. We provide some semantic and proof-theoretical results for the two four-valued contaminating logics HYB_1 and HYB_2. In particular, we provide sound and complete sequent calculi, and a characterization of the two logics—that is, a way to establish HYB_1- and HYB_2-consequence relations on the ground of $\mathsf{K}_3^{\mathsf{w}}$- or PWK-consequence relations, or, alternatively, on the ground of classical consequence. In view of the results concerning sequent calculi, we focus on so-called multiple-conclusion consequence relations. We believe this secures a nice uniformity between the semantic and the proof-theoretic parts of the paper.

The paper proceeds as follows. After introducing some preliminaries in Section 2, in Section 3 we present the basic contaminating logics $\mathsf{K}_3^{\mathsf{w}}$ and PWK from [4] and [15], respectively. This will allow the reader to familiarize with contaminating logics and their envisaged applications to computer errors. In section 4, we discuss the interaction of different sources of computer errors, with each of the sources discussed operating at different levels. In order to capture such an interaction, we introduce the *hybrid algebra* **HYB** and two four-valued contaminating logics interpreted on it, namely systems HYB_1 and HYB_2. In Section 5, we provide sound and complete annotated sequent calculi for the two logics. Section 6 provides conclusions and discuss some research directions.

2 Preliminaries

Given a similarity type ν and a countably infinitely set $X = \{p, q, r, \dots\}$ of generators (the *propositional variables*), we define the *formula algebra* **Fml** over X of type ν as the absolutely free algebra defined on X, with Fml denoting the universe of **Fml**, and the members of Fml being *formulas*, which we denote by $\phi, \psi, \theta, \dots$. $\Gamma, \Delta, \Psi \dots$ denote *sets of formulas*. In this paper, **Fml** will be a formula algebra of type $(1, 2, 2)$, namely, of the type containing the connectives $\neg, \vee, \wedge$. Given this, we feel free to omit reference to the type ν in what follows.[4]

[4] Throughout the paper, we adopt the standard notation and basic definitions from *Abstract Algebraic Logic*, as presented *e.g.* in [13]. One important exception with regard to [13], however, concerns our definition of multiple-conclusion matrix consequence (see below).

We define a *logic* (of type ν) as a pair $\mathsf{S} = \langle \mathbf{Fml}, \vdash_{\mathsf{S}} \rangle$, with $\mathbf{Fml}$ a formula algebra (of type ν) and $\vdash_{\mathsf{S}} \subseteq \mathcal{P}(Fml) \times \mathcal{P}(Fml)$ a substitution invariant *multiple-conclusion consequence relation.*

We define a *matrix* as a pair $\mathcal{M} = \langle \mathbf{A}, \mathcal{D} \rangle$ with $\mathbf{A}$ an algebra (of some given type ν) with universe A and $\mathcal{D} \subset A$. $\mathcal{D}$ is called the *filter* of $\mathcal{M}$. Informally, we think of the members of A as *truth-values*, and of members of $\mathcal{D}$ as *designated truth values.*[5]

The following notion of a submatrix is relevant for our purposes:

Definition 1. *A matrix $\mathcal{M} = \langle \mathbf{A}, \mathcal{D} \rangle$ is a submatrix of a matrix $\mathcal{M}' = \langle \mathbf{A}', \mathcal{D}' \rangle$ ($\mathcal{M} \sqsubseteq \mathcal{M}'$) if and only if $\mathbf{A}$ is a subalgebra of $\mathbf{A}'$ and $\mathcal{D} = \mathcal{D}' \cap A$.*

In this paper, we focus on matrices that have the $\mathcal{M}_{\mathsf{CL}}$ of Classical Logic as a submatrix. In particular, Classical Logic CL can be defined as $\langle \mathbf{Fml}, \models_{\mathcal{M}_{\mathsf{CL}}} \rangle$, and $\mathcal{M}_{\mathsf{CL}}$ is defined as $\langle \mathbf{B}_2, \{1\} \rangle$, where $\mathbf{B}_2 = \langle \{0, 1\}, \neg, \vee, \wedge \rangle$ is the well-known two-element Boolean algebra of type $(1, 2, 2)$. The elements 0 and 1 of its universe are informally interpreted as 'false' and 'true', respectively, with 1 being the only designated value in $\mathcal{M}_{\mathsf{CL}}$.

A further relevant notion is that of a *valuation*:

Definition 2. *A valuation is a* homomorphism $v : \mathbf{Fml} \longrightarrow \mathbf{A}$ *from a formula algebra* $\mathbf{Fml}$ *into an algebra* $\mathbf{A}$ *of the same type.*

We denote by $\mathrm{Hom}_{\mathbf{Fml},\mathbf{A}}$ the set of valuations for $\mathbf{Fml}$ defined on $\mathbf{A}$. When $\mathbf{Fml}$ is clear by the context and we wish to focus on the matrix rather than on the algebra, we write $\mathrm{Hom}_{\mathcal{M}}$. For every $\mathcal{M} = \langle \mathbf{A}, \mathcal{D} \rangle$, we let $\mathrm{Hom}_{\mathcal{M}}(\Gamma)$ be the set $\{v \in \mathrm{Hom}_{\mathcal{M}} \mid v[\Gamma] \subseteq \mathcal{D}\}$ of the *models* of Γ based on $\mathcal{M}$.

Logical matrices, in turn, can be seen to give raise to substitution invariant multiple-conclusion consequence relations—the so-called *matrix consequence relation*—as the next definition illustrates:

Definition 3. *Given a matrix $\mathcal{M} = \langle \mathbf{A}, \mathcal{D} \rangle$, the relation $\models_{\mathcal{M}} \subseteq \mathcal{P}(Fml) \times \mathcal{P}(Fml)$ defined as follows:*

$$\Gamma \models_{\mathcal{M}} \Delta \;\Leftrightarrow\; \text{for every } v \in \mathrm{Hom}_{\mathcal{M}}, \nu[\Gamma] \subseteq \mathcal{D} \text{ implies } \nu(\psi) \in \mathcal{D} \text{ for some } \psi \in \Delta$$

is a multiple-conclusion matrix consequence *relation.*

We follow standard terminology and say that Δ is a *tautology* if and only if $\varnothing \models_{\mathcal{M}} \Delta$, and we say that Γ is *unsatisfiable* if and only if $\Gamma \models_{\mathcal{M}} \varnothing$—if Γ has no models. We write $\phi \models_{\mathcal{M}} \psi$ instead of $\{\phi\} \models_{\mathcal{M}} \{\psi\}$, and $\phi, \psi \models_{\mathcal{M}} \gamma, \delta$ instead of $\{\phi, \psi\} \models_{\mathcal{M}} \{\gamma, \delta\}$. We also use other notation, writing *e.g.* Γ, Δ for $\Gamma \cup \Delta$, or Γ, ϕ for $\Gamma \cup \{\phi\}$.[6] Finally, when $\models_{\mathcal{M}_{\mathsf{S}}}$ is the matrix consequence relation of a logic S, we refer to $\models_{\mathcal{M}_{\mathsf{S}}}$ as to S-consequence.

Notice that the notion of multiple-conclusion consequence from Definition 3 differs from the one given in [13] in that the former comes with a *disjunctive* reading of the right side of $\models_{\mathcal{M}}$, while [13, Definition 1.7] comes with a *conjunctive* reading of it—implying that *all* the formulas in the conclusion-set have to be satisfied. In [13], the author himself notices that his definition is not standard.

[5] Notice that, in using these notions, we do not assume or even try to stress that we do not allow the presence of matrices whose algebraic reduct is the trivial algebra. However, as will become clear shortly, in this paper our interest is in investigating logics induced by matrices having contaminating values which, in turn, extend the two-valued matrix inducing Classical Logic—i.e. the matrix whose algebraic reduct is the two-element Boolean algebra. We would like to thank an anonymous reviewer for urging us to clarify this issue.

[6] For this notation, see also [13, Chapter 1].

Since the disjunctive reading of the right side of $\models_{\mathcal{M}}$ fits the interpretation of two-sided sequents in sequent calculi, we believe that in the present paper Definition 3 proves more suitable than the one from [13]. In particular, a uniform reading seems more appropriate in view of the results on sequent calculi from Section 5.

3 Basic Contaminating Logics

Here, we introduce two logics that are based on the so-called Weak Kleene Algebra (WKA) from [16]. These are relevant for our purposes, since the WKA is a submatrix of the structures on which HYB_1 and HYB_2 are based.

Definition 4. *[Weak Kleene Algebra] The Weak Kleene Algebra (WKA) is the algebra* **WKA** *of type* $(1,2,2)$ *such that (1)* $\mathbf{WKA} = \langle \{0, n, 1\}, \neg, \vee, \wedge \rangle$ *and (2) has operations* $\neg, \vee, \wedge$ *behaving as per Table 1.*

Table 1.

	$\neg$
1	0
n	n
0	1

$\vee$	1	n	0
1	1	n	1
n	n	n	n
0	1	n	0

$\wedge$	1	n	0
1	1	n	0
n	n	n	n
0	0	n	0

Given its behavior w.r.t. the connectives, value n from Table 1 is usually said to be *contaminating* [6, 9] or *infectious* [11, 19]. Here, we prefer the first label. The following gives a straightforward and intuitive expression to this intuitive notion:

Observation 1 (Contamination) For all formulas ϕ in Fml and valuation $v \in \mathrm{Hom}_{\mathbf{Fml},\mathbf{WKA}}$:

$$v(\phi) = n \quad \text{iff} \quad v(p) = n \text{ for some } p \in var(\phi)$$

The LTR (left-to-right) direction is shared by all the most widespread three-valued logics; the RTL (right-to-left) direction is clear from Table 1, and it implies that ϕ takes value n if some $p \in var(\phi)$ has the value, and *no matter* what the value of q is for any $q \in var(\phi) \setminus \{p\}$.

WKA provides the *simplest* case of contamination, where a value n contaminates *all* the values in the universe A of the algebra in question. Another example of this is the four-valued matrix used to interpret the system $\mathsf{S}_{\mathsf{fde}}$ from [10].

Two distinct non-trivial systems can be defined on **WKA**:[7] The logic $\mathsf{K}_3^{\mathsf{w}}$ has been introduced in [4] in order to reason about Russell's paradox and related set-theoretic antinomies. [11] has later proposed $\mathsf{K}_3^{\mathsf{w}}$ and cognate formalisms as a tool to reason about the way $C++$ processes information (see below). The logic PWK has been first introduced in [15] in order to reason about meaningless expressions and is investigated by [5, 6, 8]. We discuss some background and motivations for these logics at the end of the present section. $\mathsf{K}_3^{\mathsf{w}}$ and PWK are defined as follows:

Definition 5. $\mathsf{K}_3^{\mathsf{w}} = \langle \mathbf{Fml}, \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \rangle$ *and* $\mathsf{PWK} = \langle \mathbf{Fml}, \models_{\mathcal{M}_{\mathsf{PWK}}} \rangle$*, where:*

$$\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}} = \langle \mathbf{WK}, \{1\} \rangle \qquad \mathcal{M}_{\mathsf{PWK}} = \langle \mathbf{WK}, \{n, 1\} \rangle$$

The following observation details some validities and the most notable failures of $\mathsf{K}_3^{\mathsf{w}}$ and PWK:

[7] We do not consider here the trivial systems resulting from $\mathcal{D} = A$ and $\mathcal{D} = \varnothing$.

Observation 2 *The following holds for* $\mathsf{K}_3^{\mathsf{w}}$*-consequence and* PWK*-consequence:*

$1a\ \varnothing \not\models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \phi$ *for every* $\phi \in Fml$	$1b\ \varnothing \models_{\mathcal{M}_{\mathsf{PWK}}} \phi$ *for* ϕ *a classical tautology*
$2a\ \beta \not\models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \alpha$ *for* α *a classical tautology*	$2b\ \beta \models_{\mathcal{M}_{\mathsf{PWK}}} \alpha$ *for* α *a classical tautology*
$3a\ \alpha, \neg\alpha \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \beta$	$3b\ \alpha, \neg\alpha \not\models_{\mathcal{M}_{\mathsf{PWK}}} \beta$
$4a\ \alpha \supset (\beta \wedge \neg\beta) \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \neg\alpha$	$4b\ \alpha \supset (\beta \wedge \neg\beta) \not\models_{\mathcal{M}_{\mathsf{PWK}}} \neg\alpha$
$5a\ \alpha, \alpha \supset \beta \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \beta$	$5b\ \alpha, \alpha \supset \beta \not\models_{\mathcal{M}_{\mathsf{PWK}}} \beta$

We refer the reader to [6, 11] for these failures and validities. Given the standard definitions of *paraconsistency* and *paracompleteness*,[8] an immediate consequence of Observation 2 is that $\mathsf{K}_3^{\mathsf{w}}$ is *paracomplete* and PWK is *paraconsistent*.

$\mathsf{K}_3^{\mathsf{w}}$ shares the above failures and validities with the related Strong Kleene Logic K_3 from [16], while PWK shares the above failures and validities with the related LP by [18].

The contaminating behavior of n contributes to some further failure, which are distinctive of $\mathsf{K}_3^{\mathsf{w}}$, PWK, and their sublogics:

$\phi \not\models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \phi \vee \psi$	*Failure of Addition*
$\phi \wedge \psi \not\models_{\mathcal{M}_{\mathsf{PWK}}} \phi$	*Failure of Simplification*

In particular, we have $v(\phi \vee \psi) = n$ in any valuation v such that $v(\phi) = 1$ and $v(\psi) = n$. Since $n \notin \mathcal{D}_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}}$, this implies failure of disjunctive addition in $\mathsf{K}_3^{\mathsf{w}}$. Also, $v(\phi \wedge \psi) = n$ in any valuation v such that $v(\phi) = 0$ and $v(\psi) = n$. Since $\mathcal{D}_{\mathcal{M}_{\mathsf{PWK}}} = \{n, 1\}$, this implies failure of conjunctive simplification.

By contrast, the following *local* versions of these properties hold:

$\phi \vee \psi \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \phi \vee \neg\phi$	*Local Excluded Middle*
$\phi \wedge \neg\phi \models_{\mathcal{M}_{\mathsf{PWK}}} \phi \wedge \psi$	*Local Explosion*

[8] provides sound and complete sequent calculi for $\mathsf{K}_3^{\mathsf{w}}$ and PWK. [5] provides a sound and complete Hilbert-style axiomatization of PWK. [7] proves that $\mathsf{K}_3^{\mathsf{w}}$-consequence and PWK-consequence can be determined by classical consequence via two different variable-inclusion requirements. In particular:

Proposition 1 ([7], Theorem 3.4 and Theorem 4.3). $\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}$*- and* $\mathcal{M}_{\mathsf{PWK}}$*-consequence can be characterized as follows:*

$$\Gamma \models_{\mathcal{M}_{\mathsf{K}_3^{\mathsf{w}}}} \Delta \ \Leftrightarrow Var(\Delta') \subseteq Var(\Gamma) \text{ for some } \Delta' \subseteq \Delta \text{ s.t. } \Gamma \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$$

$$\Gamma \models_{\mathcal{M}_{\mathsf{PWK}}} \Delta \Leftrightarrow Var(\Gamma') \subseteq Var(\Delta) \text{ for some } \Gamma' \subseteq \Gamma \text{ s.t. } \Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta$$

This result provides a *characterization* of $\mathsf{K}_3^{\mathsf{w}}$ and PWK—that is, they specify necessary and sufficient conditions for a set Δ to follow from a set Γ of formulas in $\mathsf{K}_3^{\mathsf{w}}$ and PWK, respectively.

[8] A system is *paraconsistent* if it falsifies Ex Falso Quodlibet $\phi, \neg\phi \models \beta$, and is *paracomplete* if it falsifies Excluded Middle $\varnothing \models \phi \vee \neg\phi$.

Discussion. PWK has been introduced by [15] in order to capture the impact of meaningless expressions on reasoning. One point worth noting is that [15] supports that some formulas should be valid even if there are occasions in which they are meaningless—in our matrix-based setting, this equates with designating n alongside 1. [15] defends this by arguing that the *validity* of a formula should be judged solely on the basis of its *meaningful* instances.

The idea motivating introduction of $\mathsf{K}_3^{\mathsf{w}}$ in [4] is that statements such as Russell's paradox would be *meaningless*. Under this interpretation, the third value n in **WKA** represents this further semantical category, and any $v(p) = n$ is a valuation where p is deemed meaningless.

Notice that this 'meaninglessness' interpretation fares well with the fact that n is a contaminating element in **WKA**: if p is meaningless, then it is not possible to process any information involving it—that is, it is not possible to process any formula ϕ in which p occurs.

[11] provides a computational application of this conceptual and formal apparatus, with a particular attention to programming language $C++$. In this language, if some syntactical object p is to be used as a Boolean variable, the interpreter must be informed that p is to be used in this way. When the program is run, an instruction is made to allocate sufficient memory for p to take a value. To *declare* the Boolean variable p is to allocate the necessary resources. If a variable is *undeclared*, then it is *meaningless*: even if a formula is well-formed, if its atomic variables have not yet been declared, it is no more serviceable than an ill-formed string of symbols. In particular, the algorithm from Figure 1 exemplifies how a $C++$-based program would react when fed with an undeclared variable.

```
procedure DECLARATION(y)
    boolean p ← 1
    x ← (p or q)
end procedure
```

Fig. 1. Algorithm with Undeclared Variables

The fact that undeclared variables in a formula prevents the entire formula to be processed matches the contaminating behavior of the third value n from **WKA**; also, a formula that cannot be processed cannot be assigned values 0 or 1 as well, and this fits with $n \notin \mathcal{D}$. The algorithm in Figure 1 shows that addition is bound to fail exactly as it does in $\mathsf{K}_3^{\mathsf{w}}$ [11, p. 352]. All this confirms [11] in suggesting that $\mathsf{K}_3^{\mathsf{w}}$ is a suitable tool to reason about the way $C++$ processes information.

Figure 1 also suggests that undeclared variables also bring a kind of computer error: if they are involved, a $C++$-based program becomes unable to process relevant information along the lines of classical logic (or the logic of choice on which the program is based).

Application of logic to computer errors is not new. [17] proposes non-commutative disjunction and conjunction in order to reason about errors in sequential computing. Crucial to this proposal is a matrix-based semantics involving a third truth value beside 1 and 0. [1] applies the system by [17] to *critical errors* from sequential computing—that is, errors that make the computation stop—and the Strong Kleene Logic from [16] to *non-critical errors* from parallel computing—that is, errors that can be somehow remedied. Also, [1] proposes a four-valued sublogic of both McCarthy's and Kleene's systems, which allows for reasoning on both kinds of errors.

Errors due to undeclared variables differ from those considered by [1] and [17], insofar as they represent computation stops that are due to a syntactic failure. This, and the logical features of errors due to undeclared variables, justify the application of a different system such as $\mathsf{K}_3^{\mathsf{w}}$ or some relevant subsystem.

4 Four-valued contaminating logics

Failure to assign a value to an *undeclared* variable in $C++$ is an error in code, and hence on the level of *software*. Errors of this kind may cause a process to halt. Beside these kinds of errors, we have errors on the level of *hardware*. An instance of these are the physical errors that are caused when an environment attempts to retrieve a value from a physical address that is corrupt.

Physical errors that cause some failure at the level of the operating system also affects the virtual machines in the system. By contrast, errors causing failure at the level of a virtual machine need not propagate to all the environments in the system. Both errors may happen, but they will affect the entire system in different ways. This calls for an *interaction* of different types of errors, and this interaction seems to be hierarchical in a sense, since the level at which an error takes place contributes to determine how much of the environment is affected by that error. We believe that this interaction is suitably captured by applying *two* contaminating values similar to the value n from Table 1, and we briefly discuss why.

Errors in code and physical errors. Consider errors in code. The triggering of the syntactic error at the local level—that is, within the virtual machine—may cause the *environment within which the executable was run* to halt prematurely. This calls for some truth value that displays a contaminating behavior in the style of n from Table 1, since the situation we have described represents the capacity of an error to affect any string of information or environment *in which* the error takes place. At the same time, however, this error happens within the scope of a virtual machine, which in turn insulates the operating system from such local errors. This is better represented by a value that is just *partially* contaminating, that is a value that contaminates *some*, but not *all* other values. Value n from Table 1 cannot capture this, since it contaminates all other values in the universe of **WKA**. Thus, we need to adjust n to fit our current purpose.

Going to physical errors at the level of the hardware, if the *operating system* attempts to retrieve a value *on behalf of* a virtual machine from a bad address, the error that causes the operating system to fail will bring down the virtual machine alongside it. This calls for a contaminating value that affects *all* other values, including possibly partially contaminating values like the one discussed above.

In a nutshell, the logical representation of the interaction of the two kinds of errors above requires *two* values that are contaminating in some sense—in particular, we need one value to be contaminating in a weaker sense than n, and the other to be contaminating in exactly the same sense of n.

Reasoning about the interaction of the two errors. In this section we propose the two logics HYB_1 and HYB_2 as tools that can be used for reasoning about the interaction of the kind of errors that we have been discussed above. Both HYB_1 and HYB_2display the two different types of contaminating values that we see fit in capturing the two different kinds of errors that we have discussed above. Also, HYB_1 and HYB_2 comprise a designated and an undesignated contaminating value.

Whether one or more contaminating values should be designated or not is, in our view, a *pragmatic* issue, determined by an end user's interest. For instance, an end user may be concerned with the stability of the code itself and *not* in the stability of the physical memory. Thus, one might be justified in modeling this global error via a designated value. Take the concrete case of a large ontology with an integrated theorem prover, for example. Here, one might wish for certain theorems to be derivable, in spite of the potential for hardware errors. In this case, practical concerns make lead the ontology's developers to *discount* this type of situation from consideration when judging validity, just as Halldén *elects* to discount meaninglessness. Also,

when one is testing code, some tiers of errors are important to acknowledge while others are not. Simply put, whether one's code leads to a software error is part of a developer's concern; the fact that a particular piece of hardware upon which the software runs crashes due to faulty RAM is not.

We acknowledge that the examples above do not bring conclusive evidence for designating a contaminating values, but we also believe that they provide reasons for it, which would deserve further discussion and testing. We postpone a detailed discussion of this issue to a further paper. In this paper, we take this provisional reasons as strong enough to support the elaboration of four-valued logics with both designated and undesignated values.

Thus, in the next sections we extend our previous considerations to build appropriate semantic tools to model such settings. We do this by appealing to the idea of a linear order of contaminating values, such that the greater contaminating values contaminate the smaller ones and, of course, the non-contaminating values.

4.1 An algebra for the interaction of different computer errors

First, we introduce a structure that can represent the interaction between computer errors that we have envisaged above:

Definition 6. *[Hybrid Algebra] The Hybrid Algebra (HA) is the algebra* **HYB** *of type* $(1, 2, 2)$ *such that (1)* $\mathbf{HYB} = \langle \{0, n_1, n_2, 1\}, \neg, \vee, \wedge \rangle$ *and (2) has operations* $\neg, \vee, \wedge$ *behaving as per Table 2.*

Table 2.

	$\neg$
1	0
n_1	n_1
n_2	n_2
0	1

$\vee$	1	n_1	n_2	0
1	1	n_1	n_2	1
n_1	n_1	n_1	n_2	n_1
n_2	n_2	n_2	n_2	n_2
0	1	n_1	n_2	0

$\wedge$	1	n_1	n_2	0
1	1	n_1	n_2	0
n_1	n_1	n_1	n_2	n_1
n_2	n_2	n_2	n_2	n_2
0	0	n_1	n_2	0

Values n_1 and n_2 from Table 2 enjoys a sort of contaminating behavior in the style of n, but notice that the behavior of n_1 does not satisfy the conditions sorted out by Observation 1, contrary to n_2 does not. In order to qualify their different behaviors, we adjust the notion of contamination from Observation 1 and we define a full-fledged, general notion of contamination:

Definition 7. *An algebra* $\mathbf{A}$ *of type* ν *has a contaminating element* k *if and only if there is a non-empty* $A' \subseteq A$*, with* $A' \neq \{k\}$*, such that for every* m*-ary* $g \in \nu$ *and every* $\{a_1, \ldots, a_m\} \subseteq A'$*:*

$$\text{if } k \in \{a_1, \ldots, a_m\} \text{ then } g^{\mathbf{A}}(a_1, \ldots, a_m) = k$$

Both n_1 and n_2 satisfy Definition 7. Since n_2 contaminates every other value, we will say that n_2 is *absolutely* contaminating. By contrast, we will say that n_1 is just *partially* contaminating, since it contaminates all values *but* n_2.

Discussion of the two contaminating values. Given its *partially contaminating* behavior, value n_1 fits our description of how errors in code work. Indeed, n_1 does not trump any other value, and this seems to fit the fact that errors in code do not necessarily affect *any* environment, while they do prevent computation to proceed in the virtual machine where they occur.

By contrast, given its *absolutely contaminating* behavior, value n_2 fits our description of how physical errors work at the level of the operating system. Again, the former trumps any other value, and this seems fit the fact that a physical errors occurring in the operating system affects any environment.

4.2 Logics based on HYB

Two logics induced by logical matrices built using the **HYB** algebra, the Systems HYB_1 and HYB_2 are two non-trivial logics definable on **HYB**. We give sound and complete sequent calculi in Section 5. In what follows, we familiarize with the two systems, and provide characterizations in the style of Theorem 3.4 and Theorem 4.3 from [7]. The two logics are defined as follows:

Definition 8. $\mathsf{HYB}_1 = \langle \mathbf{Fml}, \models_{\mathcal{M}_{\mathsf{HYB}_1}} \rangle$ *and* $\mathsf{HYB}_2 = \langle \mathbf{Fml}, \models_{\mathcal{M}_{\mathsf{HYB}_2}} \rangle$*, where:*

$$\mathcal{M}_{\mathsf{HYB}_1} = \langle \mathbf{HYB}, \{n_1, 1\} \rangle \qquad \mathcal{M}_{\mathsf{HYB}_2} = \langle \mathbf{HYB}, \{n_2, 1\} \rangle$$

Each of HYB_1 and HYB_2 shares all the failures of $\mathsf{K}_3^{\mathsf{w}}$ and PWK, since the former are subsystems of the latter. Additionally, the following distinguish the two logics HYB_1 and HYB_2 from $\mathsf{K}_3^{\mathsf{w}}$ and PWK:

$$\phi \vee \psi \vDash_{\mathcal{M}_{\mathsf{HYB}_1}} \phi \vee \neg\phi \qquad \phi \wedge \neg\phi \nvDash_{\mathcal{M}_{\mathsf{HYB}_1}} \phi \wedge \psi$$
$$\phi \vee \psi \nvDash_{\mathcal{M}_{\mathsf{HYB}_2}} \phi \vee \neg\phi \qquad \phi \wedge \neg\phi \vDash_{\mathcal{M}_{\mathsf{HYB}_2}} \phi \wedge \psi$$

As for Local Excluded Middle, any valuation v such that $v(\psi) = v(\phi \vee \psi) = n_2$ and $v(\phi) = n_1$ is such that $v(\phi \vee \psi) \in \mathcal{D}_{\mathcal{M}_{\mathsf{HYB}_2}}$ and $v(\phi \vee \neg\phi) \notin \mathcal{D}_{\mathcal{M}_{\mathsf{HYB}_2}}$. Also, for every valuation v such that $v(\phi \vee \psi) \in \{n_1, 1\}$, we have $v(\phi \vee \neg\phi) \in \{n_1, 1\}$. Since $\mathcal{D}_{\mathcal{M}_{\mathsf{HYB}_1}} = \{n_1, 1\}$, the rule has no countermodel in $\mathcal{M}_{\mathsf{HYB}_1}$. As for Local Explosion, any valuation v where $v(\phi \wedge \neg\phi) = n_1$ and $v(\psi) = n_2$ provides a countermodel to the rule in HYB_1; for every valuation v where $v(\phi) = v(\phi \wedge \neg\phi) = n_2$, we have $v(\phi \wedge \psi) = n_2$ by contamination. Since $\mathcal{D}_{\mathcal{M}_{\mathsf{HYB}_2}} = \{n_2, 1\}$, the rule has no countermodel in $\mathcal{M}_{\mathsf{HYB}_2}$.

The following lemma plays a crucial role in proving Theorem 2 from this section and Theorem 6 from Section 5:

Lemma 1. *The consequence relations* $\models_{\mathcal{M}_{\mathsf{HYB}_1}}$ *and* $\models_{\mathcal{M}_{hybh}}$ *are* dual*, that is:*

$$\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_1}} \Delta \Leftrightarrow \Delta^{\neg} \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Gamma^{\neg}$$

where, for every $\Gamma \subseteq Fml$*,* $\Gamma^{\neg} = \{\neg\phi \in Fml \mid \phi \in \Gamma\}$*.*

Discussion of the interaction of n_1 and n_2 in HYB_1 and HYB_2. Values n_1 and n_2 may represent, as we have discussed above, code errors in a virtual machine, and physical errors in the operating system. Given what we have proposed about the pragmatic nature of designation of a contaminating value, two combinations are possible: code errors are taken as unthreatening and physical errors as fatal, or vice versa. The two options correspond to taking HYB_1 and HYB_2 as one's logic of choice, respectively. Under this informal reading, $\phi \vee \psi \models_{\mathcal{M}_{\mathsf{HYB}_1}} \phi \vee \neg\phi$ can be seen as a way of expressing that, if both ϕ and ψ are safe from fatal errors at the software level, then any of the involved formulas can be assigned a value—which implies that either ϕ or its negation will receive a designated value, given the behavior of $\neg$. By contrast, $\phi \vee \psi \nvDash_{\mathcal{M}_{\mathsf{HYB}_2}} \phi \vee \neg\phi$ implies that, if some supposedly unthreatening error occurs in processing either of ϕ or ψ, nothing excludes that the other piece of information is not involved in some error in code, which is less contaminating but fatal, under this specific interpretation.

Although we have a preference for the option that supports HYB_1 over HYB_2—we feel that physical errors at the level of the operating system can be hardly seen as unthreatening—we believe that it is worth exploring both options.

4.3 Characterizating logical consequence in HYB_1

The following is a characterization result for HYB_1:

Theorem 1.

$$\Gamma \models_{\mathsf{HYB}_1} \Delta \text{ iff } \Gamma \models_{\mathsf{PWK}} \Delta' \text{ for at least a non-empty } \Delta' \subseteq \Delta \text{ s.t. } var(\Delta') \subseteq var(\Gamma).$$

This fits with the way HYB_1 conceives of the hierarchy of errors represented by n_1 and n_2: since the fatal errors are those represented by the most contaminating value, any information that is included in the premises will be safe from fatal errors, since the premises itself must be, if they are to be designated.

Theorem 1 in turn explains why the PWK-valid inference from $\phi \wedge \neg\phi$ to $\phi \wedge \psi$ fails in HYB_1, while the PWK-valid inference from $\phi \vee \psi$ to $\phi \vee \neg\phi$ is valid in HYB_1. The former violates the variable-inclusion requirement from Theorem 1, while the latter complies with it.

4.4 Characterizing logical consequence in HYB_2

With the above notions and facts at hand, we are ready to provide the characterization result for HYB_2:

Theorem 2.

$$\Gamma \models_{\mathsf{HYB}_2} \Delta \text{ iff } \Gamma' \models_{\mathsf{K}_3^{\mathsf{w}}} \Delta \text{ for at least a non-empty } \Gamma' \subseteq \Gamma \text{ s.t. } var(\Gamma') \subseteq var(\Delta).$$

This fits with the way HYB_2 conceives of the hierarchy of errors represented by n_1 and n_2: since the fatal errors are those represented by the least contaminating value, if information from part of the premise is included in the conclusion, then the latter will be safe from fatal errors, since otherwise the premises would not be.

Theorem 2 explains $\phi \vee \psi \not\models_{\mathsf{HYB}_2} \phi \vee \neg\phi$. Indeed, although the inference is $\mathsf{K}_3^{\mathsf{w}}$-valid, there is no guarantee that the variables of $\phi \vee \psi$ are all contained in those of ϕ—notice that $\phi \vee \psi$ is, in turn, the only non-empty subset of $\phi \vee \psi$.

The following corollary will be helpful in proving Lemma 2 from Section 5. It is a straightforward consequence of Proposition 1, Theorem 1 and Theorem 2:

Corollary 1. *$\mathcal{M}_{\mathsf{HYB}_1}$-consequence and $\mathcal{M}_{\mathsf{HYB}_2}$-consequence can be characterized as follows:*

$$\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_1}} \Delta \Leftrightarrow Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma) \text{ for some } \Gamma' \subseteq \Gamma, \Delta' \subseteq \Delta \text{ s.t. } \Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$$

$$\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta \Leftrightarrow Var(\Delta') \subseteq Var(\Gamma') \subseteq Var(\Delta) \text{ for some } \Gamma' \subseteq \Gamma, \Delta' \subseteq \Delta \text{ s.t. } \Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$$

4.5 Discussion of Theorem 1 and Theorem 2

Sublogics like HYB_1 and HYB_2 are attracting increasing attention [3, 19], and they are natural way to generalize the three-valued contaminating setting from WKLs to more than one contaminating value. However, very little is known about these logics to this day. The two theorems from the present section make a significant progress in our knowledge of such logics, and we believe that this explains their relevance.

Additionally, we believe that our results make a significant progress w.r.t. [12, Observation 1], that also provides a clear direction for a general characterization methods for logics endowed

with many contaminating values. First, [12, Observation 1] is concerned with single-conclusion consequence relations, while our results can suggest a method that would apply to the more general multiple-conclusion case. Second, and more important, [12, Observation 1] concerns logics where contaminating values are not designated, while Theorem 2 provides an insight that is relevant also for logics that comprise one (or more) designated contaminating values. Although the insight from [12, Observation 1] easily extends to HYB_1, it is not clear if it extends naturally to HYB_2. Thus, the present results offer an insight that is more general than the insight offered by [12, Observation 1].

5 Sequent Calculi

We go now to the sequent calculi for HYB_1 and HYB_2. More precisely, we provide sound and complete calculi of *annotated* sequents for the two four-valued logics. An *annotated* sequent is an object of the form $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$ where $\Gamma, \Gamma', \Delta, \Delta'$ are sets of formulas of the language. In annotated sequent calculi, additional rules are provided in order to capture the interaction among formulas within squared brackets, outside square brackets, and the interaction of formulas within square brackets and formulas outside the brackets.

Our results extend the ones from [8] for $\mathsf{K}_3^{\mathsf{w}}$ and PWK. As in [8], each of our calculi places *restrictions* on several rules—more precisely, the rules need some variable inclusion condition to be satisfied in order to be applicable. We will specify the relevant restrictions when needed.

Below, we present the rules for the two annotated calculi, and we briefly discuss the differences between them and many-sided sequent calculi in the style of [2] . After that, we go to soundness and completeness of the two calculi.

5.1 Rules

Both systems include the following three rules, where for every $\Gamma \subseteq Fml$, Γ^* is any modification of Γ by permuting elements, absorbing redundancies, or duplicating formulas:

$$\frac{}{\varnothing, [\![p]\!] \Rightarrow \varnothing, [\![p]\!]}\ [\text{Axiom}]$$

$$\frac{\Gamma, [\![\Xi]\!] \Rightarrow \Delta, [\![\Theta]\!]}{\Gamma^*, [\![\Xi^*]\!] \Rightarrow \Delta^*, [\![\Theta^*]\!]}\ [\text{Structural}]$$

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, \Xi, [\![\Gamma']\!] \Rightarrow \Delta, \Theta, [\![\Delta']\!]}\ [\text{Weak}]$$

[Axiom] secures the validity of those classical axioms in which a propositional variable is within the scope of a square bracket in each sequent. [Structural] grants standard structural rules, but Weakening, *within any of the four slots*. [Weak] differs from the Weakening for non-annotated calculus in that we can only allow Weakening *outside* the scope of the bracket. The following "push" rules below meet the need to shift formulas from outside the scope of a square bracket to within its scope. It is with these rules that variable-inclusion restrictions come into play:

$$\frac{\Gamma, \phi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma', \phi]\!] \Rightarrow \Delta, [\![\Delta']\!]}\ [\text{PushL}] \qquad \frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \psi, [\![\Delta']\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \psi]\!]}\ [\text{PushR}]$$

***Restrictions for* PushL *and* PushR .** In the HYB_1 calculus, [PushL] requires the restriction $Var(\phi) \subseteq Var(\Delta')$ and [PushR] requires $Var(\psi) \subseteq Var(\Gamma \cup \Gamma')$. In the HYB_2 calculus, the two rules require $Var(\phi) \subseteq Var(\Delta \cup \Delta')$ and $Var(\psi) \subseteq Var(\Gamma')$, respectively.

Negation rules come with a pair of right rules and a pair of left rules, since we need to distinguish the case where we are introducing the sign within the scope of a square bracket from that where we are introducing the sign out of such a scope. The right rules:

$$\frac{\Gamma, [\![\Gamma', \phi]\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \neg\phi]\!]}\,[\neg\mathrm{R}_1] \qquad \frac{\Gamma, \phi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \neg\phi, [\![\Delta']\!]}\,[\neg\mathrm{R}_2]$$

The left rules:

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \psi]\!]}{\Gamma, [\![\Gamma', \neg\psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\neg\mathrm{L}_1] \qquad \frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \psi, [\![\Delta']\!]}{\Gamma, \neg\psi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\neg\mathrm{L}_2]$$

***Restrictions for* $\neg\mathbf{R}_1$,$\neg\mathbf{R}_2$, *and* $\neg\mathbf{L}_1$,$\neg\mathbf{R}_2$.** In the HYB_1 calculus, $[\neg\mathrm{R}_1]$ and $[\neg\mathrm{R}_2]$ require $Var(\phi) \subseteq Var(\Gamma \cup \Gamma')$; in the HYB_2 calculus, $[\neg R_1]$ requires that $Var(\phi) \subseteq Var(\Gamma')$, and $[\neg R_1]$ has no proviso. In both calculi, $[\neg\mathrm{L}_1]$ requires that $Var(\psi) \subseteq Var(\Delta')$ and $[\neg\mathrm{L}_2]$ has no proviso.

Conjunction rules also come in pairs:

$$\frac{\Gamma, [\![\Gamma', \phi, \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma', \phi \wedge \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\wedge\mathrm{L}_1] \qquad \frac{\Gamma, \phi, \psi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, \phi \wedge \psi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\wedge\mathrm{L}_2]$$

Rules $[\wedge\mathrm{L}_1]$ and $[\wedge\mathrm{L}_2]$ require no provisos in either HYB_1 or HYB_2. However, the following mixed rule requires a variable-inclusion restriction:

$$\frac{\Gamma, \phi, [\![\Gamma', \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma', \phi \wedge \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\wedge\mathrm{L}^*]$$

In HYB_1, the rule is admissible provided that $Var(\phi) \subseteq Var(\Delta')$, while in HYB_2, $Var(\phi) \subseteq Var(\Delta \cup \Delta')$ is required. For the right rules, we consider the case in which both conjuncts are outside of the scope of $[\![-]\!]$ and the case in which both are within its scope. Note, again, that we can appeal to [PushR] in order to cover mixed cases.

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \phi]\!] \qquad \Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \psi]\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \phi \wedge \psi]\!]}\,[\wedge\mathrm{R}_1]$$

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \phi, [\![\Delta']\!] \qquad \Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \psi, [\![\Delta']\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \phi \wedge \psi, [\![\Delta']\!]}\,[\wedge\mathrm{R}_2]$$

Again, neither $[\wedge\mathrm{R}_1]$ nor $[\wedge\mathrm{R}_2]$ requires a proviso in the two logics, but one could define an admissible rule that requires that $Var(\phi) \subseteq Var(\Gamma \cup \Gamma')$ in HYB_1 and $Var(\phi) \subseteq Var(\Gamma')$ in HYB_2:

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \phi, [\![\Delta']\!] \qquad \Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \psi]\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \phi \wedge \psi]\!]}\,[\wedge\mathrm{R}^*]$$

Disjunction rules are as follows:

$$\frac{\Gamma, [\![\Gamma', \phi]\!] \Rightarrow \Delta, [\![\Delta']\!] \qquad \Gamma, [\![\Gamma', \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, [\![\Gamma', \phi \vee \psi]\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\vee\mathrm{L}_1]$$

$$\frac{\Gamma, \phi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!] \qquad \Gamma, \psi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}{\Gamma, \phi \vee \psi, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]}\,[\vee\mathrm{L}_2]$$

Neither [$\vee$L$_1$] nor [$\vee$L$_2$] require provisos. Again, for the right rules, we consider the case in which both disjuncts are outside of the scope of $[\![-]\!]$ and the case in which both are within its scope. Note, again, that we can appeal to [PushR] in order to cover mixed cases.

$$\frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \phi, \psi]\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta', \phi \vee \psi]\!]}\ [\vee \mathrm{R}_1] \qquad \frac{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \phi, \psi, [\![\Delta']\!]}{\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, \phi \vee \psi, [\![\Delta']\!]}\ [\vee \mathrm{R}_2]$$

5.2 Soundness and completeness

Now we state *soundness* and *completeness* of HYB_1 is *sound* and *complete* with respect to $\mathcal{M}_{\mathsf{HYB}_2}$ (Theorem 3 and Theorem 4), and HYB_2 is *sound* and *complete* with respect to $\mathcal{M}_{\mathsf{HYB}_1}$ (Theorem 5 and Theorem 6).

Theorem 3 (Soundness of HYB_1). *If $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$ is provable in HYB_1, then $\Gamma \cup \Gamma' \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta \cup \Delta'$.*

In what follows, when we talk about 'the two-sided sequent calculi for PWK and $\mathsf{K}_3^{\mathsf{w}}$', we will be referring to the calculi from [8], which are presented there as fragments of Gentzen's sequent calculus for Classical Logic (indeed, as fragments where some of the operational rules were restricted with *variable inclusion* requirements). This is important for understanding the following lemma, which helps prove the completeness of HYB_1 with respect to $\mathcal{M}_{\mathsf{HYB}_2}$.

Lemma 2. *If $\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta$ such that $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, $Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma)$ and $\Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$, then $\Gamma' \Rightarrow \Delta'$ is provable in the calculus for PWK.*

Definition 9. *In the HYB_1 calculus, a PWK rule that applies only to formulas within brackets is a "bracketed rule".*

Theorem 4 (Completeness of HYB_1). *If $\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta$ such that $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, $Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma)$ and $\Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$, then $\Gamma', [\![\Gamma'']\!] \Rightarrow \Delta', [\![\Delta'']\!]$ is provable in HYB_1, where $\Gamma = \Gamma' \cup \Gamma''$ and $\Delta = \Delta' \cup \Delta''$.*

By similar means, we arrive at the corresponding results for HYB_2.

Theorem 5 (Soundness of HYB_2). *If $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$ is provable in HYB_2, then $\Gamma \cup \Gamma' \models_{\mathcal{M}_{\mathsf{HYB}_1}} \Delta \cup \Delta'$.*

Theorem 6 (Completeness of HYB_2). *If $\Gamma \models_{\mathcal{M}_{\mathsf{HYB}_1}} \Delta$ such that $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, $Var(\Delta') \subseteq Var(\Gamma') \subseteq Var(\Delta)$ and $\Gamma' \models_{\mathcal{M}_{\mathsf{CL}}} \Delta'$, then $\Gamma', [\![\Gamma'']\!] \Rightarrow \Delta', [\![\Delta'']\!]$ is provable in HYB_2, where $\Gamma = \Gamma' \cup \Gamma''$ and $\Delta = \Delta' \cup \Delta''$.*

5.3 Annotated and many-sided calculi.

Our calculi for HYB_1 and HYB_2 are *decorated*, since we use a bracketing device in each of the antecedent and succedent to track variable-inclusion properties. It is of crucial importance not to confuse this 'decoration' with the labelling deployed by the so-called many-sided sequent calculi, and not to mistake our calculi for four-sided sequent calculi. We briefly explain why.

Given a set $A = \{a_1, a_2, \ldots, a_n\}$ whose members are interpreted as truth values and where $a_1 = 0$ and $a_2 = 1$, many-side sequent calculi allow for sequents of the form $\Gamma_1 \mid \cdots \mid \Gamma_n$. The standard informal meaning of such a sequent is: 'for some i between 1 and n, and some ϕ in Γ_i, ϕ has value a_i'. In a nutshell, each "side" of a sequent plays the role of a distinct truth-value.

This suffices to understand the different between the calculi we are presenting here and many-sided sequent calculi. Contrary to the latter, the bracketing in our calculi for HYB_1 and HYB_2 is not supposed to capture (and needs not be understood as capturing) any semantic feature underlying the logic; by contrast, the device we deploy is syntactic in nature, since it tracks variable-inclusion properties.

We believe this difference is relevant for our proposal for at least two reasons. First, many-sided sequent calculi are often criticized, in light of the standard interpretation above, as smuggling semantics into proof theory, which is usually supposed not to be desirable. Our bracketing device, by contrast, does not imply any mixing of semantics and proof theory and is, therefore, free from such a criticism.

Second, the construction of a many-sided sequent calculus is a quite trivial business today, thanks to existing tools such as `MUltseq`—described, *e.g.*, in [14])—that can construct sound and complete many-sided sequent calculi for any finitely-valued logic. Since our calculi do not belong, in fact, to the family of many-sided sequent calculi, they do not fall prey of such a triviality.

6 Conclusions

In this paper, we have discussed the interaction of computer errors that come from different sources and, especially, takes place at different levels in the system. Some of these errors are suitably represented by values that are contaminating in a sense closely resembling the third value from Weak Kleene Algebra **WKA**. The paper discusses this structure together with the two non-trivial logics can be interpreted on it. These are the systems PWK from [15] and $\mathsf{K}_3^{\mathsf{w}}$ from [4]. In particular, $\mathsf{K}_3^{\mathsf{w}}$ and cognate formalisms have been given a computational interpretation in [11], where the logic is used in order to reason about those failures in $C++$-based programs that are due to the presence of undeclared variables (Section 3). Since computer errors may have a variety of different sources, and differ in their effects on the environment, we discuss the interaction of two different kinds of computer errors, namely those which occur at the level of some virtual machine, and those which occur at the level of the operating system (Section 4). In order to capture the interaction of these two kinds of errors, we introduce the four-valued algebra **HYB**, and two logics based on that: the systems HYB_1 and HYB_2. We provide characterization results for the two logics—that is, we provide necessary and sufficient conditions for two sets Γ and Δ of formulas to be in the relation of HYB_1- or HYB_2-consequence. Sound and complete sequent calculi for the systems in question are presented (Section 5). Before closing, we discuss some directions for future research.

First, we plan to devote future work to an investigation into the matter of designation (or not) of contaminating truth-values (see Section 4 for the issue). This is a very important point. Indeed, he current state of the art in applied computer science frequently encounters programs running in a cascade virtual machines nested in one another. Interest of the user and specific application may lead to *discount* some errors and consider them uninteresting and unthreatening. In this case, one might want to designate the relevant contaminating truth value, since this represent the ability of the computation to go on, the error notwithstanding. We wish to cast this general framework against the background of concrete scenarios of nested computer errors.

Another interesting issue concerns the *proof complexity* of HYB_1 and HYB_2. One way to look at the trade-off between the calculi that we have described and the method of many-sided sequent calculi is that our presentation has limited the number of additional rules at the cost of a possibly exponential increase of the search space. In particular, verifying that $\Gamma \Rightarrow \Delta$

is provable seems to require a back-and-forth procedure grabbing subsets of Γ and Δ with appropriate variable-inclusion properties until landing on $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ for which we can confirm that $\Gamma' \Rightarrow \Delta'$ is classically provable. This seems to indicate a worst-case complexity of verifying provability of a sequent as being in EXPTIME. We plan to devote future work to the investigation of proof complexity and related issues.

References

1. A. Avron and B. Konikowska. Proof systems for reasoning about computation errors. *Studia Logica*, 91(2):273–293, 2009.
2. M. Baaz, C. Fermüller, and R. Zach. Elimination of cuts in first-order many-valued logic. *Journal of Information Processing and Cybernetics*, 29:333–355, 1994.
3. E. Barrio, F. Pailos, and D. Szmus. A cartography of logics of formal inconsistency and truth. *manuscript*, 2016.
4. D. Bochvar. On a three-valued calculus and its application in the analysis of the paradoxes of the extended functional calculus. *Matematicheskii Sbornik*, 4:287–308, 1938.
5. S. Bonzio, J. Gil-Ferez, F. Paoli, and L. Peruzzi. On Paraconsistent Weak Kleene Logic: Axiomatization and Algebraic Analysis. *Studia Logica*, 105(2):253–297, 2017.
6. R. Ciuni and M. Carrara. Characterizing logical consequence in paraconsistent weak Kleene. In L. Felline, A. Ledda, F. Paoli, and E. Rossanese, editors, *New Developments in Logic and the Philosophy of Science*, pages 165–176. College Publications, London, 2016.
7. R. Ciuni and M. Carrara. Semantical analysis of weak kleene logic. *under submission*, ms.
8. M. E. Coniglio and M.I. Corbalan. Sequent calculi for the classical fragment of Bochvar and Halldén's nonsense logic. In D. Kesner and Petrucio, V., editors, *Proceedings of the 7th LSFA Workshop*, Electronic Proceedings in Computer Science, pages 125–136, 2012.
9. F. Correia. Weak necessity on Weak Kleene Matrices. *Advances in Modal Logic 4*, 2004.
10. H. Deutsch. Relevant analytic entailment. *The Relevance Logic Newsletter*, 2:26–44, 1977.
11. T. M. Ferguson. A computational interpretation of conceptivism. *Journal of Applied Non-Classical Logic*, 24(4):333–367, 2014.
12. T. M. Ferguson. Logics of nonsense and Parry systems. *Journal of Philosophical Logic*, 44(1):65–80, 2015.
13. J.M. Font. *Abstract Algebraic Logic*. College Publications, London, 2016.
14. A. J. Gil and G. Salzer. `MUltseq`: A generic prover for sequents and equations. In *Collegium Logicum: Annals of the Kurt-Gödel-Society*, volume 4, pages 229–233, Vienna, 2001. Kurt-Gödel-Society.
15. S. Halldén. *The Logic of Nonsense*. Lundequista Bokhandeln, Uppsala, Sweden, 1949.
16. S. Kleene. *Introduction to Metamathematics*. North Holland, Amsterdam, 1952.
17. J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 33–70. North-Holland Publishing Company, Amsterdam, 1963.
18. G. Priest. *In Contradiction*. Oxford University Press, Oxford, 2nd edition, 2006.
19. D. Szmuc. Defining LFIs and LFUs in extensions of infectious logics. *Journal of Applied Non-Classical Logic*, 26(4):286–314, 2016.
20. R. Wójcicki. Logical matrices strongly adequate for structural sentential calculi. *Bulletin de l'Academie Polonaise des Sciences, Série des Sciences Mathématiques, Astronomiques et Physiques*, 17:333–335, 1969.

Appendix

Proof of Lemma 1: We start with the LTR direction. Suppose that $\Gamma \vDash_{\mathcal{M}_{\mathsf{HYB}_1}} \Delta$. This means that, if $v(\psi) = \{0, n_2\}$ for every $\psi \in \Delta$, then $v(\phi) = \{0, n_2\}$ for some $\phi \in \Gamma$ and every $v \in \mathrm{Hom}_{\mathbf{Fml},\mathbf{HYB}}$. Given the behavior of n_2 w.r.t. negation, this implies that, if $v(\theta) = \{1, n_2\}$ for every $\theta \in \Delta^{\neg}$, then $v(\zeta) = \{1, n_2\}$ for some $\zeta \in \Gamma^{\neg}$ and every $v \in \mathrm{Hom}_{\mathbf{Fml},\mathbf{HYB}}$. Since

$\mathcal{D}_{\mathsf{HYB}_2} = \{1, n_2\}$, this implies $\Delta^{\neg} \models_{\mathcal{M}_{\mathsf{HYB}_2}} \Gamma^{\neg}$. The RTL direction is proved along the very same lines. ■

Proof of Theorem 1: We start with the LTR direction. We first prove that *if* $\Gamma \models_{\mathsf{HYB}_1} \Delta$, *then* $\Gamma' \models_{\mathsf{HYB}_1} \Delta$ for at least a non-empty $\Delta' \subseteq \Delta$ such that $var(\Delta') \subseteq var(\Gamma)$. Assume the antecedent as the initial hypothesis, and suppose that $\Gamma \not\models_{\mathsf{HYB}_1} \Delta'$ for every $\Delta \subseteq \Delta'$ such that $var(\Delta') \subseteq var(\Gamma)$. This implies that there is valuation $v \in \mathrm{Hom}_{\mathbf{Fml},\mathbf{HYB}}$ such that $v(\psi) \in \{n_2, 0\}$ for every $\psi \in \Delta'$ and yet $v(\phi) \in \{1, n_1\}$ for every $\phi \in \Gamma$. By the contaminating behavior of n_2 from Table 2 and $var(\Delta') \subseteq var(\Gamma)$, this implies $v(\psi) = 0$ for every $\psi \in \Delta'$. More in general, we have $v(p) \neq n_2$ for every $p \in var(\Gamma)$, and by this, we have $v(q) \neq n_2$ for every $q \in var(\bigcup_{\Delta' \in \mathbf{G}_{\Delta,\Gamma}})$. This implies that $v(\phi) = \{n_1, 1\}$ for every $\phi \in \Gamma$. v can be extended to a valuation $v' \in \mathrm{Hom}_{\mathbf{Fml},\mathbf{HYB}}$ such that $v'(p) = v(p)$ if $p \in var(\Gamma)$, and $v'(p) = n_2$ otherwise. This implies that $v'(\phi) \in \{1, n_1\}$ for every $\phi \in \Gamma$, $v'(\theta) = n_2$ for every $\theta \in \Delta \setminus \bigcup_{\Delta' \in \mathbf{G}_{\Delta,\Gamma}}$, and $v(\psi) = 0$ for every $\psi \in \Delta$. But this in turn contradicts the initial hypothesis, given the definition of HYB_1-consequence. Thus, we have that, *if* $\Gamma \models_{\mathsf{HYB}_1} \Delta$, *then* $\Gamma \models_{\mathsf{HYB}_2} \Delta'$ for at least a non-empty $\Delta' \subseteq \Delta$ such that $var(\Delta') \subseteq var(\Gamma)$. Since HYB_2 is a sublogic of PWK, we conclude that $\Gamma \models_{\mathsf{HYB}_1} \Delta$ implies $\Gamma \models_{\mathsf{PWK}} \Delta'$ for at least a non-empty $\Delta' \subseteq \Delta$ such that $var(\Delta') \subseteq var(\Gamma)$.

As for the RTL direction, assume as the initial hypothesis that $\Gamma \models_{\mathsf{PWK}} \Delta'$ for at least a non-empty $\Delta' \subseteq \Delta$ such that $var(\Delta') \subseteq var(\Gamma)$. To establish $\Gamma \models_{\mathsf{HYB}_1} \Delta'$, fix any valuation $U \in \mathrm{Hom}$ such that $v(\phi) \in \{1, n_1\}$ for every $\phi \in \Gamma'$. Our goal is to show that $v(\psi) \in \{1, n_1\}$ for some $\psi \in \Delta$. We consider two cases:
Case 1): $v(\phi) = n_1$ for some $\phi \in \Gamma$. Fix some formula $\theta \in \Gamma$ such that $v(\theta) = n_1$. By the contaminating behavior of n_1 from Table 2, there is a $q \in var(\theta)$ such that $v(q) = n_1$. Remember that $var(\Delta') \subseteq var(\Gamma)$, and suppose that $q \in var(\Gamma) \cap var(\Delta')$. Since $var(\Delta') \subseteq var(\Gamma)$ and $v(p) \neq n_2$ for every $p \in var(\Gamma)$, we have $v(q) \neq n_2$ for every $q \in var(\Delta')$. Suppose now that $v(\phi) \in \{1, n_1\}$ for every $\phi \in \Gamma$ and $v(\psi) = 0$ for every $\psi \in \Delta'$.This implies that there is a valuation $V \in \mathrm{Hom}$ such that $v(\phi) \in \{1, n_\}$ for every $\phi \in \Gamma$ and $v(\psi) = 0$ for every $\psi \in \Delta'$. But this contradicts the initial hypothesis that $\Gamma \models_{\mathsf{PWK}} \Delta'$.
Case 2): $v(\phi) \neq n_1$ for every $\phi \in \Gamma'$. This implies that $v(\phi) = 1$ for every $\phi \in \Gamma$, and, by the contaminating behavior of n_1, n_2 from Table 2, $v(p) = 1$ for every $p \in var(\Gamma)$. From this and $\Gamma \models_{\mathsf{CL}} \Delta'$ (which follows from the initial hypothesis $\Gamma \models_{\mathsf{PWK}} \Delta'$), we have that $v(\psi) = 1$ for some $\psi \in \Delta$, as desired.

Since these two cases are jointly exhaustive, we conclude $\Gamma \models_{\mathsf{HYB}_1} \Delta'$. From this and the Definition of $\models_{\mathsf{HYB}_1}$, it follows that $\Gamma \models_{\mathsf{HYB}_1} \Delta$. ■

Proof of Theorem 2: By Theorem 1 and Lemma 1. ■

Proof of Theorem 3: Any initial sequent $\varnothing, [\![p]\!] \Rightarrow \varnothing, [\![p]\!]$ has the form $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$ in which Γ and Δ are empty and $\Gamma' = \Delta' = \{p\}$. In this case, the sequent enjoys the property that:[9]

1. $Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma \cup \Gamma')$
2. $\Gamma' \subseteq \Gamma \cup \Gamma'$ and $\Delta' \subseteq \Delta \cup \Delta'$
3. The sequent $\Gamma' \Rightarrow \Delta'$ is derivable in **LK**

It can be easily checked that that this property is preserved under each of the foregoing rules. The case of the Exchange and Contraction rules, and Weakening (outside the scope of the square brackets) can be noted to preserve this property, since they correspond to properties that are

[9] As usual, this label denotes the standard sequent calculus for Classical Logic CL.

valid in every Tarskian logic and HYB_1 is a Tarskian logic, as every matrix logic is—see [20]. We notice that this property is preserved by the other rules as follows. Moreover, this can also be checked to apply straightforwardly to the "push" rules and the operational rules (in- and outside the square brackets). Hence, any derivable sequent enjoys the above tripartite property.

Now, we know that $\Xi \vDash_{\mathcal{M}_{\mathsf{HYB}_2}} \Theta$ if and only if there exists a $\Xi' \subseteq \Xi$ and a $\Theta' \subseteq \Theta$ such that $Var(\Xi') \subseteq Var(\Theta') \subseteq Var(\Xi)$ and $\Xi' \vDash_{\mathcal{M}_{\mathsf{CL}}} \Theta'$. Because of soundness of **LK** (a presentation of which is described in [8]), the above tripartite property entails validity in $\mathcal{M}_{\mathsf{HYB}_2}$. Soundness of HYB_2 with respect to $\mathcal{M}_{\mathsf{HYB}_1}$ is proved by similar reasoning. ■

Proof of Lemma 2: Assume $\Gamma \vDash_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta$. Then by Corollary 1 for $\mathcal{M}_{\mathsf{HYB}_2}$, we know that there are $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$, with $Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma)$ and $\Gamma' \vDash_{\mathcal{M}_{\mathsf{CL}}} \Delta'$. By completeness of **LK**, this implies that $\Gamma' \Rightarrow \Delta'$ is provable in **LK**. We also know that $Var(\Gamma') \subseteq Var(\Delta')$. Hence, by [8, Lemma 21], these two observations jointly imply that $\Gamma' \Rightarrow \Delta'$ is provable in the sequent calculus for PWK. ■

Proof of Theorem 4: Assume that $\Gamma \vDash_{\mathcal{M}_{\mathsf{HYB}_2}} \Delta$. Then, by Lemma 2, there is a PWK proof of $\Gamma' \Rightarrow \Delta'$. Call this proof, *i.e.* a rooted binary tree, Π. We can design an algorithm to transform a PWK proof of this sequent into an HYB_1 proof of $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$.

First, replace every node $\Xi \Rightarrow \Theta$ of Π by a node $\varnothing, [\![\Xi]\!] \Rightarrow \varnothing, [\![\Theta]\!]$. Then, place below each leaf, or axiom node, one instance of [Weak], such that from an axiom $\varnothing, [\![p]\!] \Rightarrow \varnothing, [\![p]\!]$ we infer in one step the sequent $\Gamma, [\![p]\!] \Rightarrow \Delta, [\![p]\!]$. After that, for each non-axiom node place Γ to the left of the square brackets in the antecedent and Δ to the left of the square brackets in the succedent. In the resulting proof, each PWK rule is applied within the scope of the square brackets. Moreover, we can check that every application of a PWK rule corresponds to a "bracketed rule" in HYB_1 that respects the corresponding provisos.

Actually, since Weakening is not fully admissible within the scope of square brackets, something must be said about this case. Suppose in an H proof of $\Gamma' \Rightarrow \Gamma'$ there is an *ineliminable* application of Weakening that allows to go from a node $\Xi \Rightarrow \Theta$ to a node $\Xi, \Xi' \Rightarrow \Theta, \Theta'$—whence we can legitimately call Ξ' and Θ' the active (sets of) formulas in this step. Then the current algorithm can be further specified by saying that if Π is a proof which has no ineliminable application of Weakening, then we proceed as previously stated. However, if Π *has* an ineliminable application of Weakening, then we enlarge every node (outside the square brackets) with Γ and Ξ', and Δ and Θ', in their respective sides. Finally, when the Π requires the corresponding application of Weakening, we mimic this in HYB_1 applying the [PushL] and [PushR] rules to Ξ' and Θ', as needed.

This renders a rooted binary tree Π^* with $\Gamma, [\![\Gamma']\!] \Rightarrow \Delta, [\![\Delta']\!]$ as its terminal sequent. We then proceed to apply the rules [PushL], [PushR] followed by elimination of duplicate formulas in Γ' and Δ'. We end up with a HYB_1 proof ending with $\Gamma'', [\![\Gamma']\!] \Rightarrow \Delta'', [\![\Delta']\!]$, for which $\Gamma'' \cup \Gamma' = \Gamma$ and $\Delta'' \cup \Delta' = \Delta$ and $Var(\Gamma') \subseteq Var(\Delta') \subseteq Var(\Gamma'' \cup \Gamma') = \Gamma$. ■

Proof of Theorem 6: By Theorem 4 and Lemma 1. ■