New Formalized Results on the Meta-Theory of a Paraconsistent Logic

Anders Schlichtkrull
DTU Compute - Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Richard Petersens Plads, Building 324,
DK-2800 Kongens Lyngby, Denmark
andschl@dtu.dk

Abstract

Classical logics are explosive, meaning that everything follows from a contradiction. Paraconsistent logics are logics that are not explosive. This paper presents the meta-theory of a paraconsistent infinite-valued logic, in particular new results showing that while the question of validity for a given formula can be reduced to a consideration of only finitely many truth values, this does not mean that the logic collapses to a finite-valued logic. All definitions and theorems are formalized in the Isabelle/HOL proof assistant.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Logic and verification; Theory of computation → Higher order logic; Theory of computation → Logic and databases

Keywords and phrases Paraconsistent logic, Many-valued logic, Formalization, Isabelle proof assistant, Paraconsistency

Digital Object Identifier 10.4230/LIPIcs.TYPES.2018.5

Related Version An earlier version of the present paper appears as a chapter in my PhD thesis (`http://matryoshka.gforge.inria.fr/pubs/schlichtkrull_phd_thesis.pdf`) [15].

Acknowledgements Jørgen Villadsen, Jasmin Christian Blanchette and John Bruntse Larsen provided valuable feedback on the paper and the formalization. Thanks to Freek Wiedijk for discussions. Thanks to the anonymous reviewers for many constructive comments.

1 Introduction

Classical logics are by design *explosive* – everything follows from a contradiction. This is mostly uncontroversial, but it seems problematic for certain kinds of reasoning. In paraconsistent logics, everything does not follow from a contradiction. Non-classical logics should also enjoy the benefits of formalization, and therefore this paper presents a formalization of a paraconsistent infinite-valued propositional logic.

The entry on paraconsistent logic in the Stanford Encyclopedia of Philosophy [13] thoroughly motivates paraconsistent logics by arguing that some domains do contain inconsistencies, but this should not make meaningful reasoning impossible. An example from computer science is that in large knowledge bases an inconsistency can easily occur if just one data point is entered wrong. A reasoning system based on such a database needs a meaningful way to deal with the inconsistency. Many other examples are mentioned from philosophy, linguistics, automated reasoning and mathematics. A recent book [1] looks at paraconsistency in the domain of engineering. There is no one paraconsistent logic to rule them all – there are many logics which can be used in different contexts. The encyclopedia gives a taxonomy of paraconsistent logics consisting of discussive logics, non-adjunctive systems, preservationism, adaptive logics, logics of formal inconsistency, relevant logics and many-valued logics.

Table 1 This table shows where the results of this paper have been conjectured and where their formal and informal proofs have previously been presented.

Results in	Conjecture	Formal proof	Informal proof
Section 4	Jensen and Villadsen [10]	Villadsen and me [23, 24]	the present paper
Section 5	Jensen and Villadsen [10]	the present paper	the present paper
Section 6	Villadsen and me [23, 24]	the present paper	the present paper

The logic considered here is the propositional fragment of a paraconsistent infinite-valued higher-order logic by Villadsen [19, 21, 20, 22] and more recently Jensen and Villadsen in an extended abstract [10]. The propositional logic, here called $\mathbb{V}$, has a semantics with the two classical truth values and countably infinitely many non-classical truth values. When U is a subset of the non-classical truth values, $\mathbb{V}_U$ is the logic defined the same as $\mathbb{V}$ except for the restriction that its non-classical truth values are only those in U. This does not require any change of the semantics of $\mathbb{V}$'s logical operators because they are defined in a way such that when their domain is restricted then their range is similarly restricted. In $\mathbb{V}_U$ with a finite U, one can find out whether a formula p is valid by enumerating enough interpretations that they cover all possible assignments of the propositional symbols in p. This approach does not work in $\mathbb{V}$ since there are infinitely many such interpretations. This paper shows that it is enough to consider the models in $\mathbb{V}_U$ for a finite U, but that the size of U depends on the formula considered.

The contents of this paper are as follows:

- Section 2 defines and formalizes $\mathbb{V}$. It gives an example of paraconsistency in the logic.
- Section 3 defines and formalizes $\mathbb{V}_U$.
- Section 4 proves and formalizes that for any formula p there is a finite U such that if p is valid in $\mathbb{V}_U$, it is also valid in $\mathbb{V}$. This allows the question of validity in $\mathbb{V}$ to be solved by a finite enumeration of interpretations.
- Section 5 proves and formalizes the new result that if $|U| = |W|$, then $\mathbb{V}_U$ and $\mathbb{V}_W$ consider the same formulas valid.
- Section 6 shows the new result that, to answer the question of validity in $\mathbb{V}$, one cannot fix a finite valued $\mathbb{V}_U$ once and for all because there exists a formula $\pi_{|U|}$ that is valid in this logic but not in $\mathbb{V}$. In other words, despite the result in Section 4, $\mathbb{V}$ is a different logic than any finite valued $\mathbb{V}_U$.

The formalization in Sections 2 and 3 was previously presented in a book chapter [23] and a paper [24] by Villadsen and myself. The result in Section 4 had already been conjectured by Jensen and Villadsen [10], but was, to the best of my knowledge, first proved and formalized in the mentioned book chapter [23] and paper [24]. The results in Section 5 were also conjectured by Jensen and Villadsen [10], but the results are, to the best of my knowledge, proved and formalized in the present paper for the first time. The result in Section 6 was conjectured by Villadsen and myself [24] and is, to the best of my knowledge, proved and formalized in the present paper for the first time except for a brief mention in the abstract of a talk by me [14]. For a summary of the appearances of the results see Table 1. Thus, there are no previous informal proofs to refer to for these results, and this paper will therefore both present the formalization of these results and their informal proofs. The full formalization is available online – 1500 lines of code are already in an *Archive of Formal Proofs* entry by Villadsen and myself [17], and the 800 lines corresponding to Sections 5 and 6 [16] will be added later. To make the paper easier to read, its notation is slightly different from the formalization.

2 A Paraconsistent Infinite-Valued Logic

The paraconsistent infinite-valued propositional logic $\mathbb{V}$ has two classical truth values, namely true ($\bullet$) and false ($\circ$). These are called the determinate truth values. True ($\bullet$) is the only designated value. The logic also has countably many different non-classical truth values ($_{\mathrm{I}}$, $_{\mathrm{II}}$, $_{\mathrm{III}}$, ...) [10]. These are called the indeterminate truth values. This is represented as a datatype *tv*.

datatype *tv* = *Det bool* | *Indet nat*

Det True and *Det False* represent $\bullet$ and $\circ$ respectively, and constructor *Indet* maps each natural number (0, 1, 2, ...) to the corresponding indeterminate truth value ($_{\mathrm{I}}$, $_{\mathrm{II}}$, $_{\mathrm{III}}$, ...).

The propositional symbols of $\mathbb{V}$ are strings of a finite alphabet. Here, the symbols are denoted as $\mathsf{p}, \mathsf{q}, \mathsf{r}, \ldots$. Interpretations are functions from propositional symbols into truth values. The formulas of the logic are built from the propositional symbols and operators $\neg$, $\wedge$, $\Leftrightarrow$ and $\leftrightarrow$ as well as a symbol for truth $\top$. To make them distinguishable, the logical operators in the paraconsistent logic are bold, while Isabelle/HOL's logical operators are not (e.g. $\neg$, $\wedge$, $\vee$, $\longleftrightarrow$). $\Leftrightarrow$ represents equality whereas $\neg$, $\wedge$ and $\leftrightarrow$ are designed to be generalizations of their classical counterparts. In the Isabelle/HOL formalization, the formulas are defined by a datatype *fm*, with a constructor for atomic formulas consisting of propositional symbols and with constructors for each of the operators. Additionally, a number of derived operators are defined:

$$\begin{array}{ll}
\bot \equiv \neg\, \top & \neg\!\!\neg\, p \equiv \Box\, (\neg\, p) \\
p \vee q \equiv \neg\, (\neg\, p \wedge \neg\, q) & p \barwedge q \equiv \Box\, (p \wedge q) \\
p \Rightarrow q \equiv p \Leftrightarrow (p \wedge q) & p \barvee q \equiv \Box\, (p \vee q) \\
p \rightarrow q \equiv p \leftrightarrow (p \wedge q) & \boldsymbol{\Delta}\ p \equiv (\Box\ p) \barvee (p \Leftrightarrow \bot) \\
\Box\ p \equiv p \Leftrightarrow \top & \boldsymbol{\nabla}\ p \equiv \neg\!\!\neg\, (\boldsymbol{\Delta}\ p)
\end{array}$$

In the semantics, Villadsen motivated the different cases by equalities of classical logic that also hold in $\mathbb{V}$ [19]. These motivating equalities are shown to the right of their case:

$$\begin{array}{l}
\mathit{eval}\ i\ x\ =\ i\ x \text{ if } x \text{ is a propositional symbol} \\
\mathit{eval}\ i\ \top\ =\ \bullet
\end{array}$$

$$\mathit{eval}\ i\ (\neg\, p)\ =\ \begin{cases}
\bullet & \text{if } \mathit{eval}\ i\ p = \circ \qquad \top \Leftrightarrow \neg\, \bot \\
\circ & \text{if } \mathit{eval}\ i\ p = \bullet \qquad \bot \Leftrightarrow \neg\, \top \\
\mathit{eval}\ i\ p & \text{otherwise}
\end{cases}$$

$$\mathit{eval}\ i\ (p \wedge q)\ =\ \begin{cases}
\mathit{eval}\ i\ p & \text{if } \mathit{eval}\ i\ p = \mathit{eval}\ i\ q \qquad p \Leftrightarrow p \wedge p \\
\mathit{eval}\ i\ q & \text{if } \mathit{eval}\ i\ p = \bullet \qquad q \Leftrightarrow \top \wedge q \\
\mathit{eval}\ i\ p & \text{if } \mathit{eval}\ i\ q = \bullet \qquad p \Leftrightarrow p \wedge \top \\
\circ & \text{otherwise}
\end{cases}$$

$$eval\ i\ (p \Leftrightarrow q) \;=\; \begin{cases} \bullet & \text{if } eval\ i\ p = eval\ i\ q \\ \circ & \text{otherwise} \end{cases}$$

$$eval\ i\ (p \leftrightarrow q) \;=\; \begin{cases} \bullet & \text{if } eval\ i\ p = eval\ i\ q & \top \;\Leftrightarrow\; p \leftrightarrow p \\ eval\ i\ q & \text{if } eval\ i\ p = \bullet & q \;\Leftrightarrow\; \top \leftrightarrow q \\ eval\ i\ p & \text{if } eval\ i\ q = \bullet & p \;\Leftrightarrow\; p \leftrightarrow \top \\ eval\ i\ (\neg\ q) & \text{if } eval\ i\ p = \circ & \neg q \;\Leftrightarrow\; \bot \leftrightarrow q \\ eval\ i\ (\neg\ p) & \text{if } eval\ i\ q = \circ & \neg\ p \;\Leftrightarrow\; p \leftrightarrow \bot \\ \circ & \text{otherwise} \end{cases}$$

Among the derived operators $\Box$, $\mathbf{\Delta}$ and $\mathbf{\nabla}$ are of special interest. $\Box$ maps $\bullet$ to $\bullet$ and any other value to $\circ$. In other words $\Box\ p$ states "p is true". Similarly $\mathbf{\Delta}\ p$ states "p is determinate" and $\mathbf{\nabla}\ p$ states "p is indeterminate".

The other operators can be divided in two groups – general operators ($\neg$, $\wedge$, $\vee$ and $\leftrightarrow$) and purely determinate operators ($\neg\!\!\neg$, $\curlywedge$, $\curlyvee$ and $\Leftrightarrow$). The general operators behave as expected on determinate values, and this behavior is generalized to indeterminate values. Consider for example the truth table in $\mathbb{V}_{\{\mathrm{I},\mathrm{II}\}}$ for $\vee$:

$\vee$	$\bullet$	$\circ$	I	II
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$
$\circ$	$\bullet$	$\circ$	I	II
I	$\bullet$	I	I	$\bullet$
II	$\bullet$	II	$\bullet$	II

The purely determinate operators also behave as expected on determinate values, and their behavior generalizes to indeterminate values – however this time in such a way that they always return a determinate truth value. Consider for example the truth table in $\mathbb{V}_{\{\mathrm{I},\mathrm{II}\}}$ for $\curlyvee$:

$\curlyvee$	$\bullet$	$\circ$	I	II
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$
$\circ$	$\bullet$	$\circ$	$\circ$	$\circ$
I	$\bullet$	$\circ$	$\circ$	$\bullet$
II	$\bullet$	$\circ$	$\bullet$	$\circ$

Validity is defined in the usual way, i.e. a formula is valid if it is true in all interpretations.

definition *valid* :: *"fm ⇒ bool"*
where
"valid p ≡ ∀ i. eval i p = •"

Weber [25] explains that the literature contains two competing views on paraconsistency. One states that a logic is paraconsistent iff some formulas p and q exists such that $p, \neg\ p \nvdash q$. Another view states that a logic is paraconsistent iff some formulas p and q exist such that

$\vdash p$, $\vdash \neg\, p$ and $\nvdash q$. The logic $\mathbb{V}$ is paraconsistent with respect to the first of these views. Note that with this definition, paraconsistency is a property of entailment. Villadsen [21, 19] instead encodes this as the non-validity of a formula $(\mathsf{p} \wedge (\neg\, \mathsf{p})) \Rrightarrow \mathsf{q}$. This formula is not valid in $\mathbb{V}$ since it has e.g. the counter-model mapping p to $\shortmid$ and q to $\circ$. If one insists on a notion of entailment it can, for finite sets of formulas, simply be introduced by defining that $p_1, \ldots, p_n \vdash q$ iff $p_1 \wedge \ldots \wedge p_n \Rrightarrow q$ is valid [21, 20]. With this definition it follows that $\mathbb{V}$ is paraconsistent because then the above non-validity implies that there exist formulas p and q such that $\mathsf{p}, \neg\, \mathsf{p} \nvdash \mathsf{q}$.

3 Paraconsistent Finite-Valued Logics

For any set U of indeterminate truth values, the logic $\mathbb{V}_U$ is defined as follows: $\mathbb{V}_U$ is defined in the same way as $\mathbb{V}$, except that it has a different notion of interpretations. An interpretation in $\mathbb{V}_U$ is a function from propositional symbols to the set $\{\bullet, \circ\} \cup U$ instead of to the type of all truth values.

A function *domain* constructs $\{\bullet, \circ\} \cup U$ from a set of natural numbers:

definition *domain* :: "*nat set* ⇒ *tv set*"
where
"*domain U* ≡ {*Det True*, *Det False*} ∪ *Indet* ' *U*"

Here, *Indet* '*U* denotes the image of *Indet* on U. Notice that in the formalization, U is a set of natural numbers rather than a set of indeterminate values. This is only because it is less tedious to write $\{0, 1, 2\}$ than $\{\mathit{Indet}\ 0, \mathit{Indet}\ 1, \mathit{Indet}\ 2\}$ and because being able to write *domain* $\{\mathit{Indet}\ 0, \bullet\}$ is rather pointless since $\bullet$ is added by *domain* anyway. For the same reasons, I will from now on also write e.g. $\mathbb{V}_{\{0,1,2\}}$ rather than $\mathbb{V}_{\{\mathit{Indet}\ 0,\ \mathit{Indet}\ 1,\ \mathit{Indet}\ 2\}}$. The function is called *domain* because in the higher-order version of $\mathbb{V}$ one can use the truth values as the domain of discourse.

The notion of being valid in $\mathbb{V}_U$ is formalized. The expression *range i* denotes the function range of i.

definition *valid_in* :: "*nat set* ⇒ *fm* ⇒ *bool*"
where
"*valid_in U p* ≡ ∀ *i*. *range i* ⊆ *domain U* ⟶ *eval i p* = •"

It is clear that validity in $\mathbb{V}$ implies validity in any $\mathbb{V}_U$.

theorem *valid_valid_in*: **assumes** "*valid p*" **shows** "*valid_in U p*"

Proof. If p is valid in $\mathbb{V}$, it is true in all interpretations and thus in particular those with the desired range. Therefore p is valid in $\mathbb{V}_U$. ◀

The set U can be finite or infinite. The former case in particular will be of interest in the following sections.

4 A Reduction from Validity in $\mathbb{V}$ to Validity in $\mathbb{V}_U$

When U is finite, one can find out if a formula is valid by considering all the different cases of what an interpretation might map the formula's propositional symbols to. As an example, consider the formula $(\mathsf{p} \wedge (\neg\, \mathsf{p})) \longrightarrow \mathsf{q}$ in the logic $\mathbb{V}_\emptyset$, which corresponds to classical propositional logic.

```
proposition "valid_in ∅ ((p ∧ (¬ p)) → q)"
  unfolding valid_in_def
proof (rule, rule)
  fix i :: "id ⇒ tv"
  assume "range i ⊆ domain ∅"
  then have
      "i p ∈ {•, ○}"
      "i q ∈ {•, ○}"
    unfolding domain_def
    by auto
  then show "eval i ((p ∧ (¬ p)) → q) = •"
    by (cases "i p"; cases "i q") auto
qed
```

For $\mathbb{V}$ this approach does not work, since there are infinitely many truth values. This section overcomes the problem by showing that there exists a finite subset of the interpretations in $\mathbb{V}_U$ that it is enough to enumerate. The idea is that looking at the semantics of $\mathbb{V}$ reveals that there is a lot of symmetry between the indeterminate truth values $\mathrm{I}, \mathrm{II}, \mathrm{III}, \ldots$ Specifically, the indeterminate values are all different and can be told apart using $\Leftrightarrow$, but none of them play any special role compared with the others. Intuitively, this means that one just needs to consider enough interpretations to ensure that one has considered all different possibilities of interpreting the different pairs of propositional symbols as either different or equal indeterminate truth values. Therefore it is only necessary to consider enough truth values to ensure that this is possible and thus, for any formula p, it should be sufficient to consider all the interpretations in the logic $\mathbb{V}_U$, where $|U|$ is at least the number of propositional symbols in p.

The first step towards proving this is to prove that interpretations that agree on the propositional symbols occurring in a formula also evaluate the formula to the same result. The set of propositional symbols occurring is defined recursively by the following equations:

$$
\begin{aligned}
&\mathit{props}\ \top = \{\} \\
&\mathit{props}\ x = \{x\} \text{ if } x \text{ is a propositional symbol} \\
&\mathit{props}\ (\neg\ p) = \mathit{props}\ p \\
&\mathit{props}\ (p \wedge q) = \mathit{props}\ p \cup \mathit{props}\ q \\
&\mathit{props}\ (p \Leftrightarrow q) = \mathit{props}\ p \cup \mathit{props}\ q \\
&\mathit{props}\ (p \leftrightarrow q) = \mathit{props}\ p \cup \mathit{props}\ q
\end{aligned}
$$

Hereafter, the mentioned property is proved:

```
lemma relevant_props: assumes "∀ s ∈ props p. i₁ s = i₂ s" shows "eval i₁ p = eval i₂ p"
```

Proof. Follows by induction on the formula and the definitions of *props* and *eval*. ◀

The next step is to consider an interpretation i in $\mathbb{V}$ and see that it behaves the same as a corresponding interpretation in $\mathbb{V}_U$. The idea is that i can be changed to an interpretation in $\mathbb{V}_U$ by applying a function from *nat* into U to the indeterminate values that the interpretation returns.

Given a function f of type $\mathit{nat} \Rightarrow \mathit{nat}$ and an interpretation, its application $f\ x$ to a truth value x is defined as

$$
f\ x = \begin{cases} x & \text{if } x \text{ is determinate} \\ \mathit{Indet}\ (f\ n) & \text{if } x = \mathit{Indet}\ n \end{cases}
$$

A function can also be applied to an interpretation:

$f\ i = \lambda s.\ f\ (i\ s)$

If f is an injection, then applying f to the result or to the interpretation gives the same result when evaluating a formula.

lemma *eval_change*: **assumes** "*inj f*" **shows** "*eval* (*f i*) *p* = *f* (*eval i p*)"

Proof. The proof is by induction on the formula. In each inductive case the formula consists of one of the (non-derived) logical constructors and a number of immediate subformulas. Now look at how the semantics for that logical constructor was defined. For each operator, consider the different cases of what the subformulas could evaluate to under i as specified in the semantics. Doing this generates all in all 17 different cases. Consider for instance the semantics' "otherwise"-case for $p \leftrightarrow q$. Here, it is the case that $eval\ i\ p \neq eval\ i\ q$ and that there exists a natural number n such that $eval\ i\ p = Indet\ n$ and some m such that $eval\ i\ q = Indet\ m$. Hence $Indet\ n \neq Indet\ m$ and therefore $n \neq m$. Since f is injective, also $f\ n \neq f\ m$ and $Indet\ (f\ n) \neq Indet\ (f\ m)$. The induction hypotheses are $eval\ (f\ i)\ p = f\ (eval\ i\ p)$ and $eval\ (f\ i)\ q = f\ (eval\ i\ q)$. Consider the first one. Here it is the case that $eval\ (f\ i)\ p = f\ (eval\ i\ p) = f\ (Indet\ n) = Indet\ (f\ n)$. Likewise from the second it follows that $eval\ (f\ i)\ q = f\ (eval\ i\ q) = f\ (Indet\ m) = Indet\ (f\ m)$. This implies that $eval\ (f\ i)\ p \neq eval\ (f\ i)\ q$. From this and the semantics of $\leftrightarrow$ follows $eval\ (f\ i)\ (p \leftrightarrow q) = \circ$. Likewise, from $eval\ i\ p \neq eval\ i\ q$ and the semantics of $\leftrightarrow$ follows $eval\ i\ (p \leftrightarrow q) = \circ$. These two facts allow us to establish $eval\ (f\ i)\ (p \leftrightarrow q) = \circ = f\ \circ = f\ (eval\ i\ (p \leftrightarrow q))$. And this part of the proof is done. This was just one out of the 17 cases mentioned above. For the rest I refer to the formalization. ◀

Writing out all 17 cases mentioned above would be tedious and checking all of them by hand requires discipline. Therefore, there is always the danger of overlooking a needed argument, because one case looked similar to another but really was not. Formalization enforces this discipline.

Now it is time to prove that if there are at least as many indeterminate truth values in U as the number of propositional symbols in p, then the validity of p in $\mathbb{V}_U$ implies the validity of p in $\mathbb{V}$. The lemma is expressed using Isabelle/HOL's *card* function, which for finite sets returns their cardinality and for infinite sets returns 0.

theorem *valid_in_valid*:
assumes "*card U* ≥ *card* (*props p*)"
assumes "*valid_in U p*"
shows "*valid p*"

Proof. p is proved valid by fixing an arbitrary interpretation i: First, obtain an injection f of type $nat \Rightarrow nat$ such that f maps any value in i ' (*props p*) to a value in *domain U*. This is possible because $|domain\ U| \geq |props\ p|$.

Now define the following interpretation:

$$i'\ s = \begin{cases} (f\ i)\ s & \text{if } s \in props\ p \\ \bullet & \text{otherwise} \end{cases}$$

From the properties of f and definition of i' it follows that $range\ i' \subseteq domain\ U$ and then by the validity of p in U it follows that $eval\ i'\ p = \bullet$. Furthermore, i' and $f\ i$ coincide on all

symbols in p, and therefore, by the lemma *relevant_props*, it also follows that $eval\ (f\ i)\ p = \bullet$. Now from *eval_change* follows that $f\ (eval\ i\ p) = \bullet$. By definition of the application of a $nat \Rightarrow nat$ to a truth-value it is the case that $eval\ i\ p = \bullet$. Thus any interpretation evaluates to $\bullet$ and therefore the formula is valid. ◀

theorem *valid_iff_valid_in*:
assumes "*card U* ≥ *card* (*props p*)"
shows "*valid p* ⟷ *valid_in U p*"

Proof. Follows from *valid_valid_in* and *valid_in_valid*. ◀

5 Sets of Equal Cardinality Define the Same Logic

Recall that while the indeterminate values are all different and can be told apart using ⇔, none of them play any special role compared to the others. Therefore one would expect $\mathbb{V}_U$ and $\mathbb{V}_W$ to be the same when U and W have the same cardinality. In the same way, consider what happens when $|U| < |W|$. In this case one can think of $\mathbb{V}_U$ as being $\mathbb{V}_W$ with some truth values, and thus interpretations, removed. Removing interpretations only makes it easier for a formula to be valid and thus any formula that is valid in $\mathbb{V}_W$ should also be valid in $\mathbb{V}_U$.

Isabelle/HOL defines *inj_on* such that *inj_on f A* expresses that f is an injection from A into the return type of f. In order to be able to talk about one set having smaller cardinality than another, it is useful to also define the notion of an injection from a set into another set.

definition *inj_from_to* :: "('*a* ⇒ '*b*) ⇒ '*a set* ⇒ '*b set* ⇒ *bool*" **where**
"*inj_from_to f X Y* ≡ *inj_on f X* ∧ *f* ' *X* ⊆ *Y*"

The lemma *eval_change* is generalized from the type *nat* to sets of *nat*s.

lemma *eval_change_inj_on*:
assumes "*inj_on f U*"
assumes "*range i* ⊆ *domain U*"
shows "*eval* (*f i*) *p* = *f* (*eval i p*)"

Proof. The proof is analogous to that of *eval_change*. ◀

This is enough to prove the following lemma:

lemma *inj_from_to_valid_in*:
assumes "*inj_from_to f W U*"
assumes "*valid_in U p*"
shows "*valid_in W p*"

Proof. The plan is to fix an arbitrary interpretation in $\mathbb{V}_W$ and prove that it makes p true. First, realize that $range\ (f\ i) \subseteq domain\ U$; this follows from the fact that for any x it is the case that $(f\ i)\ x = f\ (i\ x)$ and here the application of i will give an element in $domain\ W$ and then the application of f will give an element in $domain\ U$. Therefore $eval\ (f\ i)\ p = \bullet$ by the validity of p in $\mathbb{V}_U$. Then use *eval_change_inj_on* to get that $f\ (eval\ i\ p) = \bullet$ and then from the definition of the application of f to a truth value that $eval\ i\ p = \bullet$. ◀

It is now time to prove that if U and W have equal cardinality, they define the same logic.

lemma *bij_betw_valid_in*:
assumes "*bij_betw f U W*"
shows "*valid_in U p* ⟷ *valid_in W p*"

Proof. f is an injection from U into W. f^- is an injection from W into U. The lemma therefore follows from *inj_from_to_valid_in*. ◀

6 The Difference Between $\mathbb{V}$ and $\mathbb{V}_U$ for a Finite U

Section 4 showed that the question of the validity of p in $\mathbb{V}$ can be reduced to the question of its validity in $\mathbb{V}_{\{0..<|prop\ p|\}}$, where $\{n..<m\} = \{k \mid n \leq k < m\}$ for any n and m. This section shows that this does not mean that $\mathbb{V}$ collapses to a finite valued $\mathbb{V}_U$. The approach is to demonstrate a formula that is true in $\mathbb{V}_{0..n}$ but false in $\mathbb{V}$. The formula is called the pigeonhole formula. For $n = 3$ the pigeonhole formula π_3 is

$$\pi_3 = \nabla \mathsf{x}_0 \mathbin{⩓} \nabla \mathsf{x}_1 \mathbin{⩓} \nabla \mathsf{x}_2 \Rightarrow (\mathsf{x}_0 \Leftrightarrow \mathsf{x}_1) \mathbin{⩔} (\mathsf{x}_0 \Leftrightarrow \mathsf{x}_2) \mathbin{⩔} (\mathsf{x}_0 \Leftrightarrow \mathsf{x}_1).$$

I.e. it states that, assuming that x_0, x_1 and x_2 refer to indeterminate values, two of them will be the same. This is of course not true in an interpretation where they map to three different values, but if one only considers two indeterminate values there are no such interpretations. Therefore the formula is not valid in general but it is valid in $\mathbb{V}_{\{\mathsf{I},\ \mathsf{II}\}}$. Propositions x_0 and x_1 and x_2 can be thought of as pigeons and the values I and II as pigeonholes.

In order to define the formula for any n, first define the conjunction and disjunction of any list $[p_1, \ldots, p_n]$ of formulas:

$$[⩓][p_1, \ldots, p_n] = p_1 \mathbin{⩓} \cdots \mathbin{⩓} p_n$$

$$[⩔][p_1, \ldots, p_n] = p_1 \mathbin{⩔} \cdots \mathbin{⩔} p_n$$

Extend $\nabla$to a symbol that characterizes lists of indeterminate values:

$$[\nabla][p_1, \ldots, p_n] = [⩓][\nabla p_1, \ldots, \nabla p_n]$$

Given two sets S_1 and S_2, the concept of their cartesian product $S_1 \times S_2$ is well known. Their *off-diagonal product* is defined as

$$S_1 \times_{\text{off-diag}} S_2 = \{(s_1, s_2) \in S_1 \times S_2 \mid s_1 \neq s_2\}$$

Isabelle/HOL offers the function *List.product* of type $'a\ list \Rightarrow 'a\ list \Rightarrow ('a \times 'a)\ list$, which implements the cartesian product on lists representing sets. From this the *list off-diagonal product* is defined:

$$L_1 \times_{\text{off-diag}} L_2 = \mathit{filter}\ (\lambda(x, y).\ x \neq y)\ (\mathit{List.product}\ L_1 L_2)$$

The list off-diagonal product is used to introduce equivalence existence, which given a list of formulas expresses that two of the formulas in the list are equivalent.

$$[\exists\,{=}][p_1, \ldots, p_n] = [⩔]([=]((p_1, \ldots, p_n) \times_{\text{off-diag}} (p_1, \ldots, p_n)))$$

where

$$[=][(p_{11}, p_{12}), \ldots, (p_{n1}, p_{n2})] = p_{11} \Leftrightarrow p_{12}, \ldots, p_{n1} \Leftrightarrow p_{n2}$$

Let $\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2, \ldots$ be a sequence of different variables. These will form the pigeonholes. Implication, ∇, equivalence existence and the pigeonholes are combined to form the pigeonhole formula:

$$\pi_n = [\nabla][\mathsf{x}_0, \cdots, \mathsf{x}_{n-1}] \Rightarrow [\exists\,{=}][\mathsf{x}_0, \cdots, \mathsf{x}_{n-1}]$$

6.1 π_n is not valid in $\mathbb{V}$

In order to prove that the pigeonhole formula is not valid, a counter-model for it is demonstrated. This counter-model is in $\mathbb{V}_{\{0..<n\}}$ and is thus also a counter-model for the validity of the pigeonhole formula in $\mathbb{V}_{\{0..<n\}}$. The counter-model for pigeonhole formula number n is

$$c_n(y) = \begin{cases} \mathit{Indet}\ i & \text{if } y = \mathsf{x}_i \text{ and } i < n \\ \bullet & \text{otherwise} \end{cases}$$

In order to prove that it indeed is a counter-model of the pigeonhole formula, a number of lemmas are introduced that characterize the semantics of the formula's components:

lemma *cla_false_Imp*:
assumes "*eval i a* = •"
assumes "*eval i b* = ∘"
shows "*eval i* (*a* ⇒ *b*) = ∘"

Proof. Follows directly from the involved definitions. ◀

lemma *eval_CON*:
"*eval i* ([⩓] *ps*) = *Det* (∀ *p* ∈ *set ps*. *eval i p* = •)"

Proof. Note that *set ps* denotes the set of members in the list *ps*. The lemma follows by induction on the list *ps* from the involved definitions. ◀

lemma *eval_DIS*:
"*eval i* ([⩔] *ps*) = *Det* (∃ *p* ∈ *set ps*. *eval i p* = •)"

Proof. Follows by induction on the list *ps* from the involved definitions. ◀

lemma *eval_ExiEql*:
"*eval i* ([∃ =] *ps*) = *Det* (∃ (*p*₁, *p*₂)∈(*set ps* ×$_{\mathit{off\text{-}diag}}$ *set ps*). *eval i p*₁ = *eval i p*₂)"

Proof. Follows from the definition of [∃ =], the definition of $\times_{\text{off-diag}}$ and *eval_DIS*. ◀

is_indet t is defined to be true iff *t* is indeterminate. Likewise *is_det t* is true iff *t* is determinate.

lemma *eval_Nab*: "*eval i* (**∇** *p*) = *Det* (*is_indet* (*eval i p*))"

Proof. Follows directly from the involved definitions. ◀

lemma *eval_NAB*:
"*eval i* ([**∇**] *ps*) = *Det* (∀ *p* ∈ *set ps*. *is_indet* (*eval i p*))"

Proof. Follows from the definition of [**∇**], *eval_CON* and *eval_Nab*. ◀

With this one can prove that the pigeonhole formula is false under the c_n counter-model.

lemma *interp_of_id_pigeonhole_fm_False*: "*eval* c_n π_n = ∘"

Proof. The lemma *cla_false_Imp* states that an implication can be proved false by proving its antecedent true and conclusion false. Start by proving the antecedent true: The antecedent is $[\nabla][\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}]$, and this means that all the variables in $\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}$ should refer to indeterminate values, which indeed they do by the definition of c_n. The conclusion $[\exists =][\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}]$ is proved false using *eval_ExiEql*, which reduces the problem to proving that no pair of different symbols among $\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}$ evaluate to the same. That follows from how c_n is defined. ◀

From this follows that the pigeonhole formula is not valid:

theorem *not_valid_pigeonhole_fm*: "¬ *valid* π_n"

Proof. Follows from *interp_of_id_pigeonhole_fm_False*. ◀

It follows that the pigeonhole formula is not valid in $U_{\{0..<n\}}$:

theorem *not_valid_in_n_pigeonhole_fm*: "¬ *valid_in* $\{0..<n\}$ π_n"

Proof. From c_n's definition follows that *range* $c_n \subseteq$ *domain* $\{0..<n\}$. It follows that π_n is not valid in $U_{\{0..<n\}}$ by *interp_of_id_pigeonhole_fm_False* and the definition of validity in $U_{\{0..<n\}}$ ◀

6.2 π_n is valid in $\mathbb{V}_{\{0..<m\}}$ for $m < n$

In order to prove that π_n is valid in $\mathbb{V}_{\{0..<m\}}$ for $m < n$, a new lemma on the semantics of an implication is needed:

lemma *cla_imp_I*:
assumes "*is_det* (*eval i a*)"
assumes "*is_det* (*eval i b*)"
assumes "*eval i a* = • ⟹ *eval i b* = •"
shows "*eval i* (*a* ⇒ *b*) = •"

Proof. Not surprisingly, it follows directly from the involved definitions. ◀

∇ and $[\exists =]$ returning determinate values is also needed.

lemma *is_det_NAB*: "*is_det* (*eval i* ([∇] *ps*))"

Proof. The lemma follows from *eval_NAB*. ◀

lemma *is_det_ExiEql*: "*is_det* (*eval i* ([$\exists =$] *ps*))"

Proof. The lemma follows from *eval_ExiEql*. ◀

Moreover the pigeonhole principle is needed. This theorem is part of Isabelle/HOL in the following formulation:

lemma *pigeonhole*: "*card A* > *card* (*f* ‘ *A*) ⟹ ¬ *inj_on f A*"

It states that if the image of f on A is of smaller cardinality than A, then f cannot be an injection. From this follows a more specific formulation of the principle, which will be applied:

lemma *pigeon_hole_nat_set*:
assumes "*f* ‘ $\{0..<n\}$ ⊆ $\{0..<m\}$"
assumes "*m* < (*n* :: *nat*)"
shows "∃ j_1∈$\{0..<n\}$. ∃ j_2∈$\{0..<n\}$. $j_1 \neq j_2 \wedge f\, j_1 = f\, j_2$"

Proof. From the assumptions follows that *card* $\{0..<n\}$ > *card* $\{0..<m\}$ ≥ *card* (*f* ‘ $\{0..<n\}$). Therefore *pigeonhole* is applicable and the conclusion follows immediately. ◀

The pigeonhole formula will evaluate to true in any interpretation with truth values in $\mathbb{V}_{\{0..m\}}$ where $m < n - 1$:

lemma *eval_true_in_lt_n_pigeonhole_fm*:
assumes "$m < n$"
assumes "*range* $i \subseteq$ *domain* $\{0..<m\}$"
shows "*eval* $i\ \pi_n = \bullet$"

Proof. Apply *cla_imp_I* to break down the conclusion. The two first assumptions of *cla_imp_I* follow from *is_det_NAB* and *is_det_ExiEql*, and then what remains is to prove that the antecedent of π_n implies the conclusion of π_n. Therefore, assume that the antecedent, $[\nabla][\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}]$, evaluates to true. From this and *eval_NAB* follows that $\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}$ all evaluate to indeterminate values. This, together with the fact that the range of i is *domain* $\{0..<m\}$, means that i must map any x_l where $l \in \{0..<n\}$ to *Indet* k for some $k \in \{0..<m\}$. Therefore, by *pigeonhole_nat_set* there are $j_1 < n$ and $j_2 < n$ such that x_{j_1} and x_{j_2} are different but i evaluates them to the same value. This is by *eval_ExiEql* exactly what is required for the conclusion $[\exists =][\mathsf{x}_0, \ldots, \mathsf{x}_{n-1}]$ to evaluate to true. ◀

Therefore the pigeonhole formula must be valid in $\mathbb{V}_{\{0..<m\}}$.

theorem *valid_in_lt_n_pigeonhole_fm*:
assumes "$m<n$"
shows "*valid_in* $\{0..<m\}$ (*pigeonhole_fm* n)"

Proof. Follows immediately from *eval_true_in_lt_n_pigeonhole_fm*. ◀

There are many other finite sets than $\{0..<m\}$. It is therefore desirable to extend the theorem to claim that π_n is valid in any V_U where $|U| < n$. This can be done using the result from Section 5:

theorem *valid_in_pigeonhole_fm_n_gt_card*:
assumes "*finite* U"
assumes "*card* $U < n$"
shows "*valid_in* U (*pigeonhole_fm* n)"

Proof. Follows from *valid_in_lt_n_pigeonhole_fm* and *bij_betw_valid_in* ◀

6.3 $\mathbb{V}$ is different from $\mathbb{V}_U$ where U is finite

The previous subsection demonstrated that π_n is valid in e.g. $\mathbb{V}_U$ where $|U| = n$ but not in $\mathbb{V}$. Therefore the logics are different:

theorem *extend*: "*valid* $\neq$ *valid_in* U" **if** "*finite* U"

Proof. Follows from *valid_in_pigeonhole_fm_n_gt_card* and *not_valid_pigeonhole_fm*. ◀

This can be seen as a justification of the infinitely many values in the logic – they cannot once and for all be replaced by a finite subset. The reduction in Section 4 only worked because there the size of U depended on the considered formula.

7 Discussion and Related Work

My previous paper with Villadsen [24] contains a thorough discussion of related work giving an overview of various many-valued logics that have been formalized in Isabelle/HOL. I will refrain from repeating the section here and mention again only the most pertinent works namely by Marcos [12] and Steen and Benzmüller [18]. Marcos developed an ML program

that can generate proof tactics; these tactics implement tableaux that can prove theorems in various finitely many-valued logics. Steen and Benzmüller defined a shallow embedding of the many-valued SIXTEEN logic into classical HOL. That the embedding is shallow means that the authors give formulas in SIXTEEN meaning by translating them to logical expressions of classical HOL. The authors can then use a theorem prover for HOL to prove these formulas. Benzmüller and Woltzenlogel Paleo [5] used the same approach to embed several higher-order modal logics and also showed the approach applied to a sketch of a paraconsistent logic. Several other logics have been embedded in HOL in this way, including conditional logics by Benzmüller, Gabbay, Genovese and Rispoli [2], quantified multimodal logics by Benzmüller and Paulson [3], first-order nominal logic by Steen and Wisniewski [26] and free logic by Benzmüller and Scott [4]. In contrast, the formalization in this paper is a deep – rather than shallow – embedding of $\mathbb{V}$ i.e. formulas in the logic are expressed as values in HOL and a semantics is formalized that gives meaning to these formulas. This formalization thus defines datatypes for formulas and a semantics rather than a tableau or a translation.

Theorems stating that a logic cannot be characterized by finite-valued matrices are quite common in the literature on non-classical logics. For instance, Gödel [8] proved that intuitionistic logic cannot be characterized by finite-valued matrices and Dugundji [7] proved that neither can any of the modal logics S1-S5. Carnielli, Coniglio and Marcos [6] characterize the logics of formal inconsistency which are paraconsistent logics that have a so-called consistency operator, such as the $\mathbf{\Delta}$ operator of $\mathbb{V}$. The authors also prove that a number of these logics cannot be characterized by finite-valued matrices.

A noteworthy characteristic of the present formalization is that all proofs were built from the ground up in the proof assistant – they were not based on any preexisting proofs. Proof assistants make it very clear when a proof is finished, and one does not have to reread it over and over to see if everything adds up. Furthermore, in the development I tried out different definitions of the implication used in the pigeonhole formula and the proof assistant was very helpful in checking that the changes did not break any proofs. Proof assistants of course ensure correctness of proofs. Many times I stated lemmas and proved them directly in the proof assistant. Other times the insurance of correctness was a hindrance in that on the way to a correct proof it was helpful to state lemmas that were "mostly correct" and whose expressions "mostly type checked", i.e. I abstracted away from some of the details. This was often better done on a piece of paper than in the proof assistant. However, after this process was done, it was definitely worth returning to the proof assistant to see if the "mostly correct" proof held up to the challenge of being formalized and thus turned into a correct proof.

The propositional fragment of a paraconsistent infinite-valued higher-order logic has now been formalized. The formalization only considers the case where the logic has a countably infinite set of indeterminate truth values. It could also be interesting to prove and formalize theorems about what happens in case an uncountably infinite type of indeterminate truth values is allowed. This could be done by replacing *nat* in the definition of *tv* with some uncountably infinite type T. Another way would be to replace *nat* with a type variable that could then be instantiated with *nat* or T. With this in place, I conjecture it would be possible to prove that the formulas that are valid with respect to *nat* are the same as those that are valid with respect to an uncountably infinite type T. My argument in the one direction is that if the formula is valid in T then it must also be valid in *nat* since there is an injection from *nat* to T, and thus it should be possible to make a generalization of *inj_from_to_valid_in* that covers the case of uncountable infinity. In the other direction I would argue that since the cardinality of T is larger than any *props p* one should be able to reuse the proof of *valid_in_valid* to prove that if p is valid with respect to T then it is also valid with respect to *nat*.

Another obvious next step would be to formalize the whole paraconsistent higher-order logic. The basis of such an endeavor could be the formalizations of HOL Light in HOL Light and HOL4 by respectively Harrison [9] and Kumar et al. [11]. The challenge is to give a semantics to the language. In the formalization in HOL4 this is done by abstractly specifying set theory in HOL. The same specification could be used for giving a semantics to the paraconsistent higher-order logic.

8 Conclusion

This paper formalizes Villadsen's paraconsistent infinite-valued logic $\mathbb{V}$ and the $|U|$-valued logics $\mathbb{V}_U$ as well as proves and formalizes several meta-theorems of the logic. One meta-theorem shows that, for any formula, the question of its validity in $\mathbb{V}$ can be reduced to the question of its validity in $\mathbb{V}_U$ for a large enough finite U. The other meta-theorems, to my knowledge not previously proved or formalized, characterize how the number of truth-values affects truths of the logic. One of them shows that when $|U| = |W|$ then $\mathbb{V}_U$ has the same truths as $\mathbb{V}_W$. Another shows that for any finite U it is the case that $\mathbb{V}$ and $\mathbb{V}_U$ are different logics. The theory was developed in parallel with its formalization. This illustrates that proof assistants can be used as tools, not only for formalizing established results, but also for developing new results – in this case the meta-theory of a logic.

References

1 Seiki Akama, editor. *Towards Paraconsistent Engineering*, volume 110 of *Intelligent Systems Reference Library*. Springer, 2016.

2 Christoph Benzmüller, Dov Gabbay, Valerio Genovese, and Daniele Rispoli. Embedding and automating conditional logics in classical higher-order logic. *Annals of Mathematics and Artificial Intelligence*, 66(1):257–271, 2012.

3 Christoph Benzmüller and Lawrence C. Paulson. Quantified Multimodal Logics in Simple Type Theory. *Logica Universalis*, 7(1):7–20, 2013.

4 Christoph Benzmüller and Dana S. Scott. Automating Free Logic in HOL, with an Experimental Application in Category Theory. *Journal of Automated Reasoning*, January 2019.

5 Christoph Benzmüller and Bruno Woltzenlogel Paleo. Higher-Order Modal Logics: Automation and Applications. In Wolfgang Faber and Adrian Paschke, editors, *Reasoning Web (RW)*, volume 9203 of *LNCS*, pages 32–74. Springer, 2015.

6 Walter Carnielli, Marcelo E. Coniglio, and João Marcos. Logics of Formal Inconsistency. In D.M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 1–93. Springer, 2007.

7 James Dugundji. Note on a Property of Matrices for Lewis and Langford's Calculi of Propositions. *J. Symbolic Logic*, 5(4):150–151, December 1940.

8 Kurt Gödel. Zum intuitionistischen Aussagenkalkül. *Akademie der Wissenschaften in Wien*, 69:65–66, 1932.

9 John Harrison. Towards self-verification of HOL Light. In Ulrich Furbach and Natarajan Shankar, editors, *International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *LNCS*, pages 177–191. Springer, 2006.

10 Andreas Schmidt Jensen and Jørgen Villadsen. Paraconsistent Computational Logic. In P. Blackburn, K. F. Jørgensen, N. Jones, and E. Palmgren, editors, *8th Scandinavian Logic Symposium: Abstracts*, pages 59–61. Roskilde University, 2012.

11 Ramana Kumar, Rob Arthan, Magnus O. Myreen, and Scott Owens. Self-Formalisation of Higher-Order Logic: Semantics, Soundness, and a Verified Implementation. *Journal of Automated Reasoning*, 56(3):221–259, 2016.

12 João Marcos. Automatic Generation of Proof Tactics for Finite-Valued Logics. In Ian Mackie and Anamaria Martins Moreira, editors, *Tenth International Workshop on Rule-Based Programming, Proceedings*, volume 21 of *Electronic Proceedings in Theoretical Computer Science*, pages 91–98. Open Publishing Association, 2010.

13 Graham Priest, Koji Tanaka, and Zach Weber. Paraconsistent Logic. In Edward N. Zalta, editor, *Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2018 edition, 2018. URL: `https://plato.stanford.edu/archives/sum2018/entries/logic-paraconsistent/`.

14 Anders Schlichtkrull. Formalization of a Paraconsistent Infinite-Valued Logic (short abstract). *Automated Reasoning in Quantified Non-Classical Logics – 3rd International Workshop – Program*, 2018. URL: `https://easychair.org/smart-program/FLoC2018/ARQNL-program.html`.

15 Anders Schlichtkrull. *Formalization of Logic in the Isabelle Proof Assistant*. PhD thesis, Technical University of Denmark, 2018. URL: `http://matryoshka.gforge.inria.fr/pubs/schlichtkrull_phd_thesis.pdf`.

16 Anders Schlichtkrull and Jørgen Villadsen. IsaFoL: Paraconsistency. `https://bitbucket.org/isafol/isafol/src/master/Paraconsistency/`.

17 Anders Schlichtkrull and Jørgen Villadsen. Paraconsistency. *Archive of Formal Proofs*, December 2016. , Formal proof development. URL: `http://isa-afp.org/entries/Paraconsistency.html`.

18 Alexander Steen and Christoph Benzmüller. Sweet SIXTEEN: Automation via Embedding into Classical Higher-Order Logic. *Logic and Logical Philosophy*, 25(4):535–554, 2016.

19 Jørgen Villadsen. Combinators for Paraconsistent Attitudes. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics (LACL)*, volume 2099 of *LNCS*, pages 261–278. Springer, 2001.

20 Jørgen Villadsen. A Paraconsistent Higher Order Logic. In B. Buchberger and J. A. Campbell, editors, *Artificial Intelligence and Symbolic Computation (AISC)*, volume 3249 of *LNCS*, pages 38–51. Springer, 2004.

21 Jørgen Villadsen. Paraconsistent Assertions. In G. Lindemann, J. Denzinger, I. J. Timm, and R. Unland, editors, *Multi-Agent System Technologies (MATES)*, volume 3187 of *LNCS*, pages 99–113, 2004.

22 Jørgen Villadsen. Supra-logic: Using Transfinite Type Theory with Type Variables for Paraconsistency. *Logical Approaches to Paraconsistency, Journal of Applied Non-Classical Logics*, 15(1):45–58, 2005.

23 Jørgen Villadsen and Anders Schlichtkrull. Formalization of Many-Valued Logics. In H. Christiansen, M.D. Jiménez-López, R. Loukanova, and L.S. Moss, editors, *Partiality and Underspecification in Information, Languages, and Knowledge*, chapter 7. Cambridge Scholars Publishing, 2017.

24 Jørgen Villadsen and Anders Schlichtkrull. Formalizing a Paraconsistent Logic in the Isabelle Proof Assistant. In Abdelkader Hameurlain, Josef Küng, Roland Wagner, and Hendrik Decker, editors, *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, volume 10620 of *LNCS*, pages 92–122. Springer, 2017.

25 Zach Weber. Paraconsistent Logic, 2019. URL: `https://www.iep.utm.edu/para-log/`.

26 Max Wisniewski and Alexander Steen. Embedding of Quantified Higher-Order Nominal Modal Logic into Classical Higher-Order Logic. In Christoph Benzmüller and Jens Otten, editors, *ARQNL 2014. Automated Reasoning in Quantified Non-Classical Logics*, volume 33 of *EPiC Series in Computing*, pages 59–64. EasyChair, 2015. `doi:10.29007/dzc2`.