Inconsistent mathematics in Isabelle

and some reflections on metalanguage

David Ripley

Background: PIM and subDMQ

Paradoxes and Inconsistent Mathematics (Weber 2021, henceforth PIM) is the state of the art in inconsistent mathematics.

It builds within the long AAL tradition of inconsistent naive theories.

The book includes set theory, arithmetic, algebra, analysis, topology.

Today: just set theory!

Two axioms

Abstraction: $x \in \{y \mid A\} \longleftrightarrow A[y \mapsto x]$

Extensionality: $x = y \longleftrightarrow \forall w(w \in x \longleftrightarrow w \in y)$

Two axioms

really, **Strong abstraction:** $x \in \{y, z \mid A\} \longleftrightarrow A[y \mapsto x, z \mapsto \{y, z \mid A\}]$

Extensionality: $x = y \longleftrightarrow \forall w(w \in x \longleftrightarrow w \in y)$

What logic, tho?

These set-theoretic axioms can be added to many different logics, resulting in many different set theories.

In the early 20th century, set theorists took the logic as given and fussed about paradox-proof set-theoretic axioms.

In the early 21st, we do the reverse: take the naive set-theoretic axioms as given and fuss about paradox-proof logics.

"[W]e must, on the one hand, restrict these principles sufficiently to exclude [triviality] and, on the other, take them sufficiently wide to retain all that is valuable in this theory."

"[W]e must, on the one hand, restrict these principles sufficiently to exclude [triviality] and, on the other, take them sufficiently wide to retain all that is valuable in this theory."

That's Zermelo, but could have been Weber.

The goal is not just to find any old logic where the axioms don't trivialize.

LP has long been known as one such, and there are many others.

The aim is to support sustained mathematical reasoning,
and success is measured by doing that reasoning.

PIM uses a logic called SUBDLQ
I work in a close relative that I call SUBDMQ.

I won't present either logic here;
suffice to say they don't have the vice of being oversimple.

Happy to talk about these,
but the session is short,
and this is still the introduction.

What you should know:

SUBDMQ has two arrows $\Rightarrow$, $\rightarrow$

$\Rightarrow$ is the main engine of reasoning; $\rightarrow$ appears in the set-theoretic axioms

$(A \rightarrow B) \Rightarrow (A \Rightarrow B)$ is an axiom

both obey modus ponens, so neither can contract, for Curry reasons

that is, $(A \rightarrow (A \rightarrow B)) \Rightarrow (A \Rightarrow B)$ had better not be provable

Community and software

How do errors get prevented and fixed in mathematics?

How do errors get prevented and fixed in mathematics?

Through community.

The community of folks working in this area is small.

That means it's easy for errors to go undetected and uncorrected.

Plus, these logics are both weird and complex.

It takes time to build good intuitions.

For almost all of us, our intuitions built on other logics are deep and convincing; errors are easy to make.

Finally, the logic is a moving target.

PIM's SUBDLQ is at least the second logic by that name;
here I've used a slightly different logic yet again.

The project is one of continual redesign based on experience.

Doing new work often involves being the only person
working with a given logic.

Proof assistants can help!

The general idea

There are many different proof asssistants:
Lean, Coq, Isabelle, Agda, …

Each implements some general logic, and provides a structured way to input proofs in that logic.

The software checks whether what's entered is a correct proof, and often provides tools for developing proofs as well.

The general idea

There are many different proof asssistants:
Lean, Coq, Isabelle, Agda, …

Each implements some general logic, and provides a structured way to input proofs in that logic.

The software checks whether what's entered is a correct proof, and often provides tools for developing proofs as well.

```
lemma implB': "(A ⇛ B) ⇛ (B ⇛ C) ⇛ A ⇛ C"
  proof -
    from implC and implB show "(A ⇛ B) ⇛ (B ⇛ C) ⇛ A ⇛ C" ..
  qed
```

```
lemma entl_antecedent_strengthening: "(A ⟶ B) ⟹ (A & C ⟶ B)"
  proof -
    from entl_impl and entl_contra have "(A ⟶ B) ⟹ (¬B ⟶ ¬A)" ..
    from this and entl_disj_inl have step1:"(A ⟶ B) ⟹ (¬B ⟶ ¬A ∨ ¬C)"
      by(rule entl_trans_10)
    from entl_impl and entl_contra have "(¬B ⟶ ¬A ∨ ¬C) ⟹ (¬(¬A ∨ ¬C) ⟶ ¬¬B)" ..
    from step1 and this have "(A ⟶ B) ⟹ (¬(¬A ∨ ¬C) ⟶ ¬¬B)" ..
    from this and dne have "(A ⟶ B) ⟹ (¬(¬A ∨ ¬C) ⟶ B)"
      by(rule entl_trans_10)
    from dm_cnd and this show "(A ⟶ B) ⟹ (A & C ⟶ B)"
      by(rule entl_trans_01)
  qed
```

Trustworthy

If a proof assistant is implemented correctly,
then what it says is a proof really *is* one.

The community detecting errors
is the community of users of that proof assistant.

Other Isabelle users don't need to know about subDMQ or naive sets;
we can benefit from their expertise nonetheless.

Build intuitions

Working in a proof assistant quickly builds intuitions for an unfamiliar logic.

You can try many angles and get immediate feedback;
it's easy to quickly develop strategies and identify patterns.

Adaptable

When you remove an axiom, the assistant will notice,
and complain about everything you used that axiom to prove.

So you can immediately carry over everything else;
no need to re-prove.

```
axiomatization
  where entlI:           "A ⟶ A"
    and entl_cs:         "(A ⟶ B) & (B ⟶ C) ⟶ (A ⟶ C)"
    and entl_contra:     "(A ⟶ B) ⟶ ¬B ⟶ ¬A"
    and entl_cel:        "A & B ⟶ A"
    and entl_conj_comm:  "A & B ⟶ B & A"
    and entl_conj_ass:   "A & (B & C) ⟶ (A & B) & C"
    and entl_disj_inl:   "A ⟶ A ∨ B"
    and entl_disj_inr:   "B ⟶ A ∨ B"
    and lem:             "A ∨ ¬A"
    and dn_bi:           "¬¬A ⟷ A"
    and dm_dnc_bi:       "A ∨ B ⟷ ¬(¬A & ¬B)"
    and dm_cnd_bi:       "A & B ⟷ ¬(¬A ∨ ¬B)"
    and dist_cd_bi:      "A & (B ∨ C) ⟷ (A & B) ∨ (A & C)"
    and dist_dc_bi:      "A ∨ (B & C) ⟷ (A ∨ B) & (A ∨ C)"
    and dm_ans_bi:       "∀ P ⟷ ¬∃(%x. ¬P(x))"
    and dm_sna_bi:       "∃ P ⟷ ¬∀(%x. ¬P(x))"
    and entl_ui:         "∀ P ⟶ P(t)"
    and all_disj:        "∀(%x. A ∨ P(x)) ⟶ A ∨ ∀ P"
    and entl_impl:       "(A ⟶ B) ⟹ A ⟹ B"
    and nimpl_nentl:     "¬(A ⟹ B) ⟹ ¬(A ⟶ B)"
    and impl_cex:        "A & ¬B ⟹ ¬(A ⟹ B)"
    and implB:           "(A ⟹ B) ⟹ (C ⟹ A) ⟹ C ⟹ B"
    and implC:           "(A ⟹ B ⟹ C) ⟹ B ⟹ A ⟹ C"
    and implK:           "A ⟹ B ⟹ A"
    and impl_disj_left:  "(A ⟹ C) ⟹ (B ⟹ C) ⟹ A ∨ B ⟹ C"
    and impl_conj_in:    "A ⟹ B ⟹ A & B"
    and conj_import:     "(A ⟹ B ⟹ C) ⟹ A & B ⟹ C"
    and all_ante:        "∀(%x. P(x) ⟶ A) ⟶ ¬ P ⟶ A"
```

```
axiomatization
  where entlI:           "A ⟶ A"
    (* and entl_cs:        "(A ⟶ B) & (B ⟶ C) ⟶ (A ⟶ C)" *)
    and entl_contra:     "(A ⟶ B) ⟶ ¬B ⟶ ¬A"
    and entl_cel:        "A & B ⟶ A"
    and entl_conj_comm:  "A & B ⟶ B & A"
    and entl_conj_ass:   "A & (B & C) ⟶ (A & B) & C"
    and entl_disj_inl:   "A ⟶ A ∨ B"
    and entl_disj_inr:   "B ⟶ A ∨ B"
    and lem:             "A ∨ ¬A"
    and dn_bi:           "¬¬A ⟷ A"
    and dm_dnc_bi:       "A ∨ B ⟷ ¬(¬A & ¬B)"
    and dm_cnd_bi:       "A & B ⟷ ¬(¬A ∨ ¬B)"
    and dist_cd_bi:      "A & (B ∨ C) ⟷ (A & B) ∨ (A & C)"
    and dist_dc_bi:      "A ∨ (B & C) ⟷ (A ∨ B) & (A ∨ C)"
    and dm_ans_bi:       "∀ P ⟷ ¬∃(%x. ¬P(x))"
    and dm_sna_bi:       "∃ P ⟷ ¬∀(%x. ¬P(x))"
    and entl_ui:         "∀ P ⟶ P(t)"
    and all_disj:        "∀(%x. A ∨ P(x)) ⟶ A ∨ ∀ P"
    and entl_impl:       "(A ⟶ B) ⟹ A ⟹ B"
    and nimpl_nentl:     "¬(A ⟹ B) ⟹ ¬(A ⟶ B)"
    and impl_cex:        "A & ¬B ⟹ ¬(A ⟹ B)"
    and implB:           "(A ⟹ B) ⟹ (C ⟹ A) ⟹ C ⟹ B"
    and implC:           "(A ⟹ B ⟹ C) ⟹ B ⟹ A ⟹ C"
    and implK:           "A ⟹ B ⟹ A"
    and impl_disj_left:  "(A ⟹ C) ⟹ (B ⟹ C) ⟹ A ∨ B ⟹ C"
    and impl_conj_in:    "A ⟹ B ⟹ A & B"
    and conj_import:     "(A ⟹ B ⟹ C) ⟹ A & B ⟹ C"
    and all_ante:        "∀(%x. P(x)) ⟶ A) ⟶ ¬ P ⟶ A"
```

Automation

Since proofs in these assistants are structured text,
they can be produced by programs.

Two broad roles for automation:

Automate away the boring stuff

Enlist automation in finding new proofs

Where things stand

logical claims

PIM presents SUBDLQ first,
and PIM claims 32 theorems of SUBDLQ in presenting the logic.

I've got all of these verified in SUBDMQ in Isabelle.

set-theoretic claims

There are a lot of set-theoretic claims made in PIM;
Over 100 proved, including all major claims of the first 2 (of 3) sections.

There are only two minor claims so far I've been unable to prove:

- $\{x\} \subseteq \{x\} \cap \{x\}$
- $x \subsetneq y \otimes y \subsetneq z \Rightarrow x \subsetneq z$

but I can show the weaker:
$\{x\} \sqsubseteq \{x\} \cap \{x\}$

this one I doubt altogether

some proved claims

comprehension: $\exists x \forall y (y \in x \longleftrightarrow P(x))$

Basic Law V: $\forall x (P(x) \longleftrightarrow Q(x)) \Leftrightarrow \{x \mid P(x)\} = \{x \mid Q(x)\}$

empty set: $x \in \emptyset \rightarrow A$ and universe: $A \rightarrow x \in \mathcal{V}, \quad \mathcal{P}(\mathcal{V}) \equiv \mathcal{V}$

singletons: $x = y \Leftrightarrow \{x\} = \{y\}$

pairs: $\{x, y\} = \{u, v\} \Leftrightarrow (x = u \otimes y = v) \vee (x = v \otimes y = u)$

ordered pairs: $\langle x, y \rangle = \langle u, v \rangle \Rightarrow x = y \otimes u = v$

more proved claims

$\subseteq$ is a partial order

$\sqsubseteq$ is a preorder

$x \subseteq y \Rightarrow x \sqsubseteq y$

$x \cap y \subseteq x$ and $x \subseteq x \cup y$

$x \cup y \equiv y \Leftrightarrow x \sqsubseteq y$

$x \cap y \equiv x \Rightarrow x \sqsubseteq y$

$x \sqsubseteq y \Rightarrow x \cap x \sqsubseteq x \cap y$

$x \subseteq y \Rightarrow y^c \subseteq x^c$ and $x^{cc} = x$

$x \sqsubseteq y \Rightarrow y \equiv (y \cap x^c) \cup x$

$x \sqsubseteq y \Rightarrow \mathcal{P}(x) \sqsubseteq \mathcal{P}(y)$

some more proved claims

Zermelo pair, union, separation, powerset, infinity

fixed points: $\exists t \forall x (x \in t \Leftrightarrow P(t))$

strong Cantor: any $f : Y \to \mathcal{P}(X)$ is not surjective$_1$

next steps

Build automation, particularly a deduction theorem

Do some more inconsistent maths!

Metalanguage

a universal logic

"[W]hen it comes to pure deductive logic, there are no concessions. Validity is a global property. If an argument form is 'valid' in some domains but not others, then it is not valid. The whole point is that logic should work come what may, and we don't know what will come. …

[I]f some contradictions are true, then nonparaconsistent logic is invalid, tout court. The same logic must be used everywhere—perhaps most importantly, in developing foundational mathematics [PIM p.94]"

object and meta

As part of this, Weber aims to avoid any separation between object- and meta-language. PIM quotes Priest approvingly: “[T]he whole point of the dialetheic solution to the semantic paradoxes is to get rid of the distinction between object language and meta-language.”

wut

I find this deeply mysterious.

The object language is a particular first-order language.
The metalanguage is mathematically-flavoured English.

These are rather obviously different,
and neither Priest nor Weber seems to be doing anything to bridge the gap.

my best stab

It does seem like there's some relationship between classical logic and the kinds of reasoning you typically see in maths journals.

I've got no great theory of that relationship
(other than that it's obviously not identity).

But maybe the idea is that our own informal reasoning should bear the converse of that relation, whatever it is, to SUBDMQ or some successor.

In a proof assistant, on the other hand, we have a clear question in view.

In Isabelle (really, Isabelle/Pure), I'm working in higher-order intuitionistic logic with definitions; let's call that IHOL.

It's in IHOL that I say: these are the axioms of the set theory and logic, and these are the rules. And what Isabelle checks is IHOL proofs from those specifications to the claim that such-and-such is provable in the resulting set theory.

Two reasons not to use subDMQ where Isabelle uses IHOL,
plus a reason it wouldn't matter if we did:

I don't know if pim's arithmetic can support axiomatic proof theory.
(Cancellation of $\times$, used in Godel numbering, is a key open question.)

Even if it does, I don't have the skill or inclination
to write a subDMQ-based proof assistant.

And even if someone did do that, they would surely write it in something
not subDMQ-based, reproducing the issue up a level.

So all this work in Isabelle works in IHOL as a metalanguage,
and some metalanguage mismatch is unavoidable with any proof assistant.

Weber would be suspicious.

IHOL includes, for example, a contracting vertebrate conditional.

So it does not meet PIM's standards for deductive validity;
deductive validity cannot allow any such connective.

not good enough?

Even if results shown in Isabelle are reliable, “Mathematical ‘proofs’ constructed using reliable but logically invalid steps may be convincing enough arguments, but they are not proofs [PIM p.89]”

not good enough?

Even if results shown in Isabelle are reliable, “Mathematical ‘proofs’ constructed using reliable but logically invalid steps may be convincing enough arguments, but they are not proofs [PIM p.89]”

Suppose for a moment that this is so:
the results shown in Isabelle are reliable, just not proved.

metalanguage to the rescue

These reliable results are of the form:
"such-and-such has a proof in the naive set theory".

If we believe that reliably, we believe that such-and-such does have a proof that meets PIM's standards.

We have not proved that it has such a proof, but demanding that is demanding too much.

but is IHOL really reliable?

I think IHOL is reliable in this domain:

we lack any example where it goes wrong;

no complicated reasoning is involved; and

we can extract the PIM-approved proofs from the Isabelle code.

(It's certainly more reliable than I am;
I do not lack examples of my own errors!)

Conclusion

Inconsistent math heads should be using proof assistants:

avoid errors

build intuitions

support trying out different logics

Worries about metalanguage mismatch are misplaced here:

Assistants can reliably say that something has a proof in a target system

That's all we need, and all we can reasonably want