To QED and Beyond: Adventures in Vibe Coding

– Zar Goertzel

- This talk is a brief summary of my experience in autoformalization++ since Sep 2025.
  - Challenges and successes.
- One of the biggest is the “Productivity-Reliability Paradox”:
  - AI coding amplifies the trade-off between doing more and verifying the work.

- Most of the work is on this github: https://github.com/zariuq/ai-agents/tree/main/lean-projects/mettapedia/papers.
- Feel free to clone it and as your friendly AI further questions.

Sep 2025
Metamath Verifier in Go [Failed]

1. Asked Codex to write a Metamath verifier in Go.
2. It passed all the “tests” and superficially “looked plausible”.
3. Matthew House on the Google Groups found numerous facepalm-worthy omissions.

Lesson: LLM coding agents ‘miss the point’ but ‘pass the tests’.

Oct 2025
Autoformalizing Ben Goertzel's "human-style" 4CP Proof Attempt

- Ben Goertzel (dad) tried to complete, with AI support, the 4CP proof idea of Lou Kauffman and George Spencer-Brown (of Laws of Form).
- I took the opportunity to explore autoformalization in Mizar, MM0, Isabelle, Lean, and Megalodon.
  - [Mizar proved the most painful for the AIs!]
- Eventually in Megalodon, Claude foiled the proof attempt.
  - https://github.com/zariuq/ai-agents/tree/main/megalodon/4CT
  - After which, all the AIs 'agreed'.
- Eventually, we are trying again with v23.5 of a 4CP proof attempt.. :p

Hindsight: testing out LLM agent and ITP compatibility on proving false might be a bad idea…

Algebraic Foundations of Inference

From Autoformalization through Proof Repair to Theory Extensions

[Dec 25 - Feb 26] – Zar Goertzel

- Knuth & Skilling's *Foundations of Inference* article aims to provide an intuitive, more general derivation of probability theory from algebraic symmetries over *an event lattice* and a *valuation space*, improving upon Cox, Jaynes, and de Finetti.
- However, the proofs seemed difficult to go through and check for correctness….
  - Like Thierry Coquand, I sought increased (non-total) understanding.
- Initial *autoformalization* in Lean 4, Mathlib 4.25, with Claude Code and Codex ran into endless complaints of *"circularities"* in the proof (and Claude does make *excuses*).
- Upon request, Claude introduced *KS Separation* as a lemma, which it tried to prove, and eventually found a counterexample proving it necessary (after Ben Goertzel suggested axiomatizing it).

Algebraic Foundations of Inference

- Codex suggested a *direct cuts*-based proof, which is much simpler than the one in the paper.
- And then GPT 5.2 Pro dug up Alimov's work in 1950 on deriving commutativity in ordered semigroups without "anomalous pairs" (essentially infinitesimals), further simplifying the development, and this combined with Hölder's embedding has been formalized in Lean already by Eric Paul)
- Unlike in "most" reports, the AIs' proofs seem fundamentally superior:
  - FoI Appendix A "grid induction": 20,063 lines
  - Dedekind cuts: 2,344 lines
  - Alimov–Hölder route: 773 lines (+ 3,552 lines of Eric Paul's)

Algebraic Foundations of Inference

- The AIs found alternative proofs for the other main results of KS's paper, too.
- By the end, they found *completeness and Scott continuity* conditions over the event lattice and valuations that lead to this KS set-up extending to σ-Additivity, i.e., conjecturing and theory exploration.

- This project convinced me that we're entering the QED era.

Lessons:

- The AIs really love finding counterexamples!
- Request thorough background literature reviews.
- Sometimes alt proof ideas are worth following.

Algebraic Foundations of Inference

The example necessitating the separation property.

```
-- Type definitions (lines 38-40)
abbrev SDBase := PN ×ₗ N
abbrev SD := WithBot SDBase

-- Semidirect operation: (u,x) ⊕ (v,y) = (u+v, x + 2^u·y) (lines 52-59)
def baseOp (p q : SDBase) : SDBase :=
  (p.1 + q.1, p.2 + (N.pow 2 (p.1 : N)) * q.2)
def op : SD → SD → SD
  | ⊥, b => b | a, ⊥ => a | some p, some q => some (baseOp p q)

-- All K&S base axioms satisfied (lines 315-324)
instance : KnuthSkillingAlgebra SD where
  op := op; ident := ⊥
  op_assoc := op_assoc; op_ident_right := ...; op_ident_left := ...
  op_strictMono_left := ...; op_strictMono_right := ...; ident_le := ...

-- Key theorems: non-commutative, fails separation (lines 333, 447-449)
theorem op_not_comm : op exX exY ≠ op exY exX := by simp; decide
theorem not_KSSeparation : ¬ KSSeparation SD := ...
```

Sep 2025
Metamath Verifier in Go [Failed]

1. Asked Codex to write a Metamath verifier in Go.
2. It passed all the “tests” and superficially “looked plausible”.
3. Matthew House on the Google Groups found numerous facepalm-worthy omissions.

Lesson: LLM coding agents ‘miss the point’ but ‘pass the tests’.

Additional Autoformalizations [Lean 4]

Warning: may be PAC (probably approximately correct)

- Ramsey36 [Dec 17,: Lawrence Paulson's idea to verify an algorithm to use in the proof helped remove native_decide use!
- Operational Semantics in Logical Form [1-3.26; 40k LoC]: Meredith, Stay, and Wells' algorithm from rewrite systems to modal (◇ ⊣ □) type systems over their behavior.
  - Seems 'operational', but the output types were only applied to toy examples….
- Some papers on ρ-calc + π-calc + μ-calc [2.26]
- Hutter's Universal Artificial Intelligence book, incl. AIXI, especially chapters 2-5 [12.25]
  - And papers: Self-Modification of Policy and Utility Function (Everitt et al. AGI-16)
- World Model Calculus, extending Probabilistic Logic Networks (2008, Goertzel et al.)
  - Also ProbLog and (infinite) Markov Logic Networks to show their connections.
  - And Prolog syntax and semantics to help analyze languages written in Prolog.
- Markov de Finetti representation theorems [2-4.26, 24k LoC] to try to ground PLN in Solomonoff induction in some cases.
  - This was one of the most difficult results to obtain.

Sep 2025
Lean 4-verified Metamath Verifier

- Not giving up, we added 70 new unit tests based on *The Metamath Book*'s specs, allowing AI agents to easily make "correct" verifiers.
- Mario Carneiro had the specs in Lean 4 already, too.
- The AIs extended his verifier to cover the full specs, and set out to prove it compliant.
  - They kvetched up a storm about inductive loop invariants and pushed hard to axiomatize parser invariants.
- They also had two passes of the parser, which Mario rightly chided them into correcting.
- Now it seems plausibly correct and can handle metamath-exe, metamath-knife, and our own interpretations of the specs.

- Software verification seems harder than math formalization.

Mar - May 2026
CeTTa: C-based MeTTa (MeTTa Type Talk)

- MeTTa is a novel programming language that centers on pattern matching and rewriting, aiming to be an “AGI cognitive language”. https://metta-lang.dev/
- PeTTa is the current *best implementation*, which translates directly to swi-pl, making it literally a “functional Prolog” dialect.
  - PeTTa and the original HE MeTTa have *almost* the same specs.
- Which begs the question: why didn’t we just use Prolog sooner?
  - Or going further, why not just use C instead of Rust?
- Getting a fully-functional HE MeTTa with performance closer to PeTTa was *surprisingly easy* (roughly 28k LoC).
  - Faster startup, comparable memory consumption, etc.
- Scaling well robustly, running longer, covering edge-cases etc is a harder long-tail.
- “Advanced” features such as term-sharing, control over non-deterministic eval (e.g., between BFS and DFS), etc finally led to “getting noodled beyond repair”.
  - Lesson?: implement one tricky feature at a time?

May 2026
AI Hub – Extending Josef Urban's Automation

1. AI coding agent with goal-specific meta-prompt
2. AI reviewing agent to advise *continue, nudge, or escalate*
3. Watchdog who spams (1) with meta-prompt depending on (2)
4. Project manager agent to discuss progress with and dynamically finetune the AI Hub.

- So far, this seems to help a lot in some cases.
  - When there's a clear task where the AIs mostly just need to "follow the proofs".
- The AIs still seem to benefit a lot from my human "big picture guidance" and "long-term memory".
  - Perhaps upgraded premise selection and grepping built into (3) will help?

Probably Approximately Correct

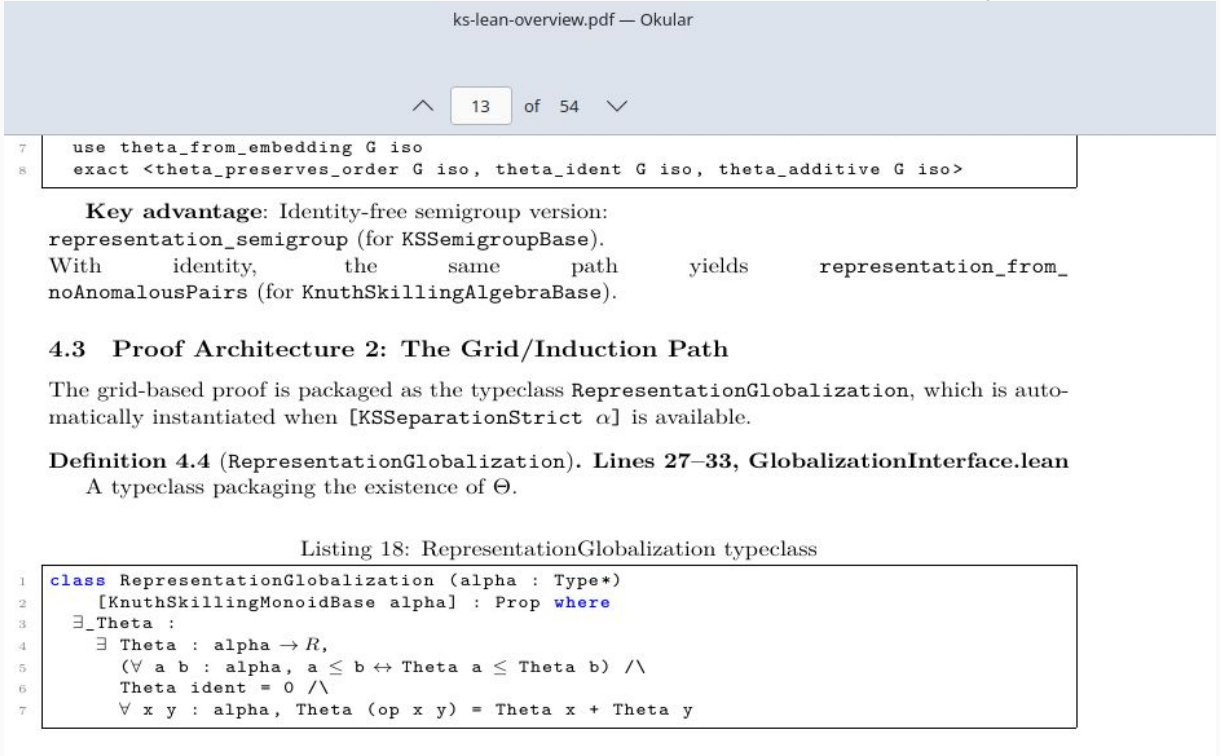
- The big question: how do we know the work is *‘correct’*?

Probably Approximately Correct

- The big question: how do we know the work is *‘correct’*?
- Answer: check every single line carefully and apply *all the usual human rigor*.

Probably Approximately Correct

- For Foundation of Inference, I asked for a *Lean overview* walking me through the primary definitions and theorem statements so I could *“check that they seem ok”* more easily.



Probably Approximately Correct

- For Foundations of Inference, I asked for a *Lean overview* walking me through the primary definitions and theorem statements to *“check that they seem ok”* more easily.
  - There were some odd details (e.g., surrounding the use of identity where not strictly necessary) where I needed to ask questions, resulting in improved results or presentation.
- “3-AI Quorum”: I find it helps to have 2-3 AI systems agree that the work seems solid before “spending my valuable human attention on it” (e.g., Claude Codex, Codex, and ChatGPT Pro).
  - Also ask them to “find bugs in the work”!

Probably Approximately Correct

- For Foundations of Inference, I asked for a *Lean overview* walking me through the primary definitions and theorem statements to *“check that they seem ok”* more easily.
  - There were some odd details (e.g., surrounding the use of identity where not strictly necessary) where I needed to ask questions, resulting in improved results or presentation.
- “3-AI Quorum”: I find it helps to have 2-3 AI systems agree that the work seems solid before “spending my valuable human attention on it” (e.g., Claude Codex, Codex, and ChatGPT Pro).
  - Also ask them to “find bugs in the work”!
- Recursively use developments in future work, unit testing all the way.
  - If I use the ‘Prolog formalization’ to reason about actual PeTTa code, which works, then this is a positive sign that the implementation is ‘not too wrong’.

Probably Approximately Correct

- For Foundations of Inference, I asked for a *Lean overview* walking me through the primary definitions and theorem statements to *“check that they seem ok”* more easily.
  - There were some odd details (e.g., surrounding the use of identity where not strictly necessary) where I needed to ask questions, resulting in improved results or presentation.
- “3-AI Quorum”: I find it helps to have 2-3 AI systems agree that the work seems solid before “spending my valuable human attention on it” (e.g., Claude Codex, Codex, and ChatGPT Pro).
  - Also ask them to “find bugs in the work”!
- Recursively use developments in future work, unit testing all the way.
  - If I use the ‘Prolog formalization’ to reason about actual PeTTa code, which works, then this is a positive sign that the implementation is ‘not too wrong’.
- Lining up ‘specs’ with ‘code’ (in comments) seems to help.
  - I imagine LLMs can hold both formal and informal versions in context.

Probably Approximately Correct

- For Foundations of Inference, I asked for a *Lean overview* walking me through the primary definitions and theorem statements to *“check that they seem ok”* more easily.
  - There were some odd details (e.g., surrounding the use of identity where not strictly necessary) where I needed to ask questions, resulting in improved results or presentation.
- “3-AI Quorum”: I find it helps to have 2-3 AI systems agree that the work seems solid before “spending my valuable human attention on it” (e.g., Claude Codex, Codex, and ChatGPT Pro).
  - Also ask them to “find bugs in the work”!
- Recursively use developments in future work, unit testing all the way.
  - If I use the ‘Prolog formalization’ to reason about actual PeTTa code, which works, then this is a positive sign that the implementation is ‘not too wrong’.
- Lining up ‘specs’ with ‘code’ (in comments) seems to help.
  - I imagine LLMs can hold both formal and informal versions in context.
- PAC: most *issues* seem figureoutable and tend to lead to more correct versions… 🚀