Autoformalization Experiments with Felix

Adrian De Lon[1,2] and Atle Hahn[2]

[1] University of Bonn, Germany
[2] AI4REASON, Czechia

1 Introduction

Autoformalization has recently become feasible at textbook scale [4, 2], but scalability alone does not make the resulting artefacts useful to mathematicians. In current LLM-based workflows, an informal proof often expands into long chains of low-level helper claims, prover-oriented citations, and system-specific proof engineering. Thus we see *understandability* as a central challenge for autoformalization. The resulting development should remain readable, reviewable, maintainable, and traceable to its informal source. Source fidelity cannot be checked via formal verification, so readability and traceability are important for making the autoformalizations auditable.

Felix (formerly Naproche-ZF)[1] is an experimental proof assistant for natural formalization [5]. It reimplements key ideas from SAD and the older Naproche system [7, 6], with larger interconnected formalizations and faster iteration in mind. Felix's improved parser, more efficient internal processing, parallel ATP calls, and caching enable larger formalizations such as the autoformalization project described here. The input format is a controlled fragment of mathematical English embedded in LaTeX, so autoformalizations targeting Felix can stay reasonably close to the textbook source. Felix's declarative proof vernacular uses steps such as 'Assume', 'Take', and 'Show that', while routine reasoning is delegated to ATPs. ATPs are useful for autoformalization because they allow a coding agent to simply state intermediate claims without having to spell out all low-level logical bookkeeping. This keeps proof steps closer to ordinary mathematical exposition and accelerates the autoformalization loop. Reducing the number of explicit proof steps also makes the formalization easier for humans to review. For our chosen topic of point-set topology, Felix's set-theoretic foundation is also a natural fit.

Felix formalizations are interpreted in higher-order Tarski–Grothendieck set theory, formulated in classical simple type theory. Felix extracts user-defined vocabulary and parses its controlled language into an internal logical representation. User-defined nouns, adjectives, and verbs are represented by predicates, while bounds in quantifier phrases become guards; for example, 'Every natural number is an ordinal' becomes $\forall n\,(\text{natural\_number}(n) \to \text{ordinal}(n))$. A Felix text is considered *verified* when all formal content parses unambiguously, passes well-formedness checks, and all generated logical obligations are accepted relative to the available definitions, theorems, assumptions, and background axioms. For this autoformalization project, all external proof obligations were exported in TPTP FOF format to Vampire 5.0 [3]. Felix currently trusts Vampire's answers without reconstructing proof objects.

2 Autoformalization of Munkres's *Topology*

As a first experiment, we autoformalized parts of Munkres's *Topology* in Felix [1], using Codex with GPT-5.2-Codex and reasoning effort set to *high*. To do so, we employed a generate–check–repair loop inspired by the one used in [2] for the Megalodon prover, but suitably adapted to

[1] Source code: https://github.com/adelon/felix.

Felix. Our autoformalization tool translates the textbook's informal LaTeX section by section to Felix's controlled language. Felix then gives parsing and well-formedness feedback and checks the generated proof obligations through Vampire, with failures fed back into the next repair step.

In larger proofs, the autoformalizer tends to insert explicit references to guide premise selection; in the final topology corpus, about 80% of proof steps include such justifications. In the case study on Munkres's *Topology*, this approach covered about 100 pages from Chapters 2–4 and produced a verified corpus of more than 50,000 lines, including arguments such as epsilon–delta continuity and compact–Hausdorff homeomorphism theorems. From-scratch verification takes about 10 minutes on typical consumer hardware.

Before the distillation step discussed below, the Felix version was roughly half as long as the Megalodon autoformalization of the same textbook source [2]. In terms of computation time, we estimate that our autoformalization tool was at most a factor of two slower than the corresponding Megalodon tool. This comparison depends on the underlying LLM used by the two systems and should therefore be regarded as provisional, given the rapid pace of model development.

3 Proof distillation

As a second experiment, we pursued *proof distillation* [1], i.e., a postprocessing step that takes an already verified formalization and attempts to remove locally superfluous proof detail while preserving correctness. Note that proof distillation is more than proof compression. The goal is not the shortest possible proof, but a formal document that remains verifiable, readable, structurally clear, and traceable to the source text. This is especially important for natural formalizations, where the formal document is intended to function as a mathematical text in its own right.

For our starting point, we extended the existing autoformalization, this time using GPT-5.4, again with *high* reasoning effort, up to the first genuinely substantial result in Munkres's book: Urysohn's lemma. At that stage, the formalization comprised 64,000 lines in total, including about 59,000 lines within proof blocks.

The distillation experiments were carried out with an automated loop using the same coding agent, Felix's parser, and Felix's proof checker. Since verification with Felix can take several minutes for files of the size encountered in our experiment, the workflow partitioned the large source file into pieces. This allowed earlier checked pieces to be cached while later pieces were excluded until needed.

The agent was given custom instructions, including general recommendations and examples of recurring distillation patterns. For example, it was encouraged to introduce reusable helper lemmas whenever possible, in order to reduce or eliminate proof steps in later proofs.

In [1], we used two distillation patterns. The first removes intermediate proof steps that are used only once, moving their cited premises into a later step. The second replaces subproofs of certain universal claims with a single declarative statement justified by the accumulated references from the eliminated subproof. For each pattern, we included two or three simple examples. In our experiments, these relatively simple examples were sufficient for the coding agent to handle substantially more complicated situations and to generalize the narrow patterns to many other cases. After nine rounds of distillation, taking about ten days in total, the total proof length was reduced from about 59,000 lines across 429 proofs to 19,500 lines across 465 proofs.

An additional postprocessing step performed in [1] was what we call *proof elaboration*, a step complementary to distillation. Elaboration can improve readability by adding or reorganizing subproof blocks, adding informative intermediate claims and other helpful anchoring constructs, and using transitions such as `Thus`, `Hence`, and `Therefore` more deliberately. A first elaboration experiment increased the total proof length slightly, from about 19,500 to 20,500 lines, while improving the presentation of several proofs in the generated HTML view; see the next section.

4 HTML rendering tool

To facilitate review of the formalization, we implemented an HTML rendering tool that converts structured mathematical TEX sources into interactive documents featuring a table of contents, collapsible proofs and subproofs, MathJax rendering, optional display of the raw source, clickable cross-references, tooltip previews, global search, and controls for selectively showing or hiding helper lemmas and definitions.

The HTML version of the distilled and elaborated formalization is available online at

`https://atlehahn.github.io/Felix-Formalizations/docs/index.html`.

Note. The original version of the formalization covered Munkres's book only through Urysohn's lemma. The current version covers Munkres's entire treatment of general topology (§§12–50). Moreover, all proofs in the standard library that were previously marked "Omitted." have now been completed.

The natural-language character of Felix, together with our HTML rendering tool, makes it relatively easy to compare the informal textbook version of each item (definition, lemma, theorem, or proof) with its formal counterpart.

We plan to add another feature to the HTML rendering tool that should make this comparison even easier. Namely, we intend to embed the informal versions of all textbook definitions, lemmas, and theorems, together with their proofs, into the HTML output. These will be accessible through a suitable tooltip mechanism, for example when the user hovers over keywords such as "Definition," "Lemma," "Theorem," or "Proof." Helper lemmas and definitions are an exception: since they were introduced by our autoformalization tool and have no counterpart in the textbook, there is no informal version to display.

Acknowledgements

This work was supported by the EU through the ERC grant *NextReason* (doi:10.3030/101200949).

References

[1] Adrian De Lon, Atle Hahn and Josef Urban. 'Distilling Autoformalized Proofs'. In: *Proceedings of the 19th Conference on Intelligent Computer Mathematics (CICM 2026)*. To appear. 2026.

[2] Josef Urban. *130k Lines of Formal Topology in Two Weeks: Simple and Cheap Autoformalization for Everyone?* arXiv:2601.03298. 2026.

[3] Filip Bártek, Ahmed Bhayat et al. 'The Vampire Diary'. In: *Computer Aided Verification*. Ed. by Ruzica Piskac and Zvonimir Rakamarić. Cham: Springer Nature Switzerland, 2025, pp. 57–71. ISBN: 978-3-031-98682-6.

[4] Ke Weng, Lun Du, Sirui Li, Wangyue Lu, Haozhe Sun, Hengyu Liu and Tiancheng Zhang. *Autoformalization in the Era of Large Language Models: A Survey*. arXiv:2505.23486. 2025.

[5] Adrian De Lon. 'The Naproche-ZF Theorem Prover (Short Paper)'. In: *Automated Reasoning*. Ed. by Christoph Benzmüller, Marijn J.H. Heule and Renate A. Schmidt. Cham: Springer Nature Switzerland, 2024, pp. 105–114. ISBN: 978-3-031-63498-7.

[6] Adrian De Lon, Peter Koepke, Anton Lorenzen, Adrian Marti, Marcel Schütz and Makarius Wenzel. 'The Isabelle/Naproche Natural Language Proof Assistant'. In: *Automated Deduction – CADE 28*. Ed. by André Platzer and Geoff Sutcliffe. Cham: Springer International Publishing, 2021, pp. 614–624. ISBN: 978-3-030-79876-5. URL: https://doi.org/10.1007/978-3-030-79876-5_36.

[7] Konstantin Verchinine, Alexander Lyaletski and Andrei Paskevich. 'System for Automated Deduction (SAD): a tool for proof verification'. In: *Automated Deduction–CADE-21* (2007), pp. 398–403.