

Autoformalization with Naproche-ZF

Atle Hahn

Czech Institute of Informatics, Robotics and Cybernetics

October 5, 2025

Joint work with Adrian De Lon, Josef Urban and Peter Koepke

Introduction

Main Task

Autoformalization of definitions and lemmas with Naproche-ZF.
(Autoformalization of proofs is left to future work.)

Comments

Exploratory project for studying topics like

- comparison: prompt engineering vs fine-tuning
- comparison: AWS API vs OpenAI API and different models
- experimentation with some simple ways of scaffolding
- exploration of possible ways of extending the grammar of Naproche-ZF in order to make AF easier
-

Additional Tasks

- Develop a simple “Formalization IDE” for Naproche-ZF
- Improve the native error messages of Naproche-ZF
- Extend the Naproche-ZF library

Approach: Preliminaries

- For the main task we took a source file in LaTeX format (a short textbook on inverse semigroups)
- In the textbook most definitions did **not** appear inside a `\begin{definition} ... \end{definition}` environment but were part of the main text.
- Currently we only use prompt engineering and a simple feedback loop involving the syntax checker.



Step 1: Preprocessing with state of the art LLMs like o3, o4 mini and later GPT-5 Thinking:

- Extract all definitions from the source text and rewrite them in separate LaTeX environments
- Split definitions which introduce more than one concept into several definitions
- Choose a good label for each definition
- Highlight the (single) concepts introduced in a definition (e.g. by using `\bf` or `\emph`) so that Step 2 below gets easier
- Extract a (hopefully) complete list of well-known concepts not explicitly introduced in the source text and convert them to a “dummy definition”

Step 2: Actual formalization with smaller & cheaper reasoning model (like e.g. DeepSeek R1) using prompt engineering and a simple feedback loop based on the syntax checker

Results

Original source text

Original source text

Results of preprocessing

The results of preprocessing step are given in the following files:

- Definitions to be formalized
- “Dummy Definitions”
- Short initial “seed” list of manually formalized definitions

Semi-automatic AF

The following screenshot should give an idea of the workflow of the semi-automatic AF.

Example for use of “Formalize Button” in IDE

Batchwise AF

Example for Round 1 (out of 4) of AF in IDE

Example of prompt used for the first formalization

Semi-automatic AF, revisited

Example for use of “Candidates Button” in IDE

Main Results

- Percentage of definitions with syntactically correct formalization proposals: 97.7% (duration: 2.5 hours)
- See

List of syntactically correct formalization proposals (after 4 runs)

for getting an impression of the semantic quality of the formalization proposals

- **First practical benefits:** Acceleration by 2x or even 3x for the manual completion of the formalization of all definitions (exploiting the combined formalization proposals by the LLMs of 4 AF runs)

Remark

For the case of lemmas we expect similar results. In order to get meaningful values we will repeat the AF experiments for definitions with the new and much larger list of definitions extracted with GPT-5 Thinking.

Outlook

Short Term Goals

- Experiment with different additional/alternative extensions of the Naproche-ZF grammar.
- Improve the error messages of the syntax checker fed back into the LLM prompt for a retry.
- Shorten the formalization proposals by asking a LLM to make better use of previous definitions.
- Work with Naproche structures (instead of tuples like in the examples above).
- Experiments with fine-tuning once the Mizar → Naproche export is completed.
- Begin an exploratory project for the AF of proofs.

Medium Term Goals

- Error messages and formalization quality provided via the IDE must be so good that a human mathematician without previous experience with proof assistants can start working successfully with Naproche-ZF after reading the docs for just a short time (ideally just 15 minutes).
- Situations where the user gets stuck on an error must be very rare.

Long Term Goals

- Fully autonomous AF with Naproche-ZF as an alternative to Lean for use in smaller projects by mathematicians with little experience regarding proof assistants.
- Potential inspiration for research groups working with other proof assistants. It is possible that certain ideas/tricks can be discovered more easily for Naproche-ZF first
- (A suitably extended version of) Naproche-ZF could serve as the target language for the informalization of fully formalized mathematics obtained by AF using other proof assistants.

Advantage over plain informalization: Naproche-ZF syntax checker and proof checker.