Prompt Style, Not Translation Register, Drives Autoformalization of Euclid into Lean 4

Reza Ramji

Abstract

A textbook proof is written for people; Lean 4 is a language a computer can verify line by line. *Autoformalization*—translating ordinary mathematical prose into a proof assistant automatically—is a long-standing goal that large language models have made newly plausible. Using Euclid's *Elements* Book I and GPT-4o, we ask a narrow, testable question: does the English *wording* of a proposition change how well a model formalizes it? A counterbalanced 3×3 design crossing three translation registers with three few-shot prompt styles (2,653 runs) shows that prompt style is the lever and translation register is not. Prompt style spans a 24-point range in compile rate (72.6% vs. 48.4%, $p < 0.001$); translation register has no significant effect once prompt style is held fixed ($p = 0.603$). An earlier "one translation wins" result ($p = 0.008$) proved to be a prompt–translation confound, visible in only one column of the crossed design and reversed in the other two. A separate probe showed that a small local model (Qwen2.5-Coder-7B) had memorized the benchmark—it compiled most often from *no math at all*—forcing the switch to GPT-4o. Counterbalanced prompt designs and an empty-input memorization check should be standard practice when reporting autoformalization results.

1 Background and context

A proof written in natural language carries meaning that only a human reader fully reconstructs. A proof written in Lean 4 carries meaning a machine can check: Lean is a dependently typed proof assistant in which a theorem and its proof are terms whose types the kernel verifies. If a Lean proof type-checks, it is correct—no appeal to authority, intuition, or the reader's goodwill. This is what makes formalization attractive and also what makes it expensive: encoding informal mathematics in Lean by hand is slow expert work, and the formal mathematics library Mathlib, though large, covers only a fraction of the literature.

Autoformalization produces that formal encoding automatically from informal input. Recent large language models can generate syntactically plausible Lean from a natural-language statement, which raises a practical question for anyone deploying such a tool on their own mathematics: *what determines whether it works?* Two candidate levers are easy to confuse. One is the *input*—the exact wording of the mathematics handed to the model. The other is the *prompt*—the instructions and worked examples (few-shot demonstrations) that frame the task. This paper isolates the two.

We use Euclid's *Elements*, Book I, as the source text: an ideal testbed for a wording study, because the same 48 propositions survive in sharply different English registers, all expressing identical geometric content:

- **Heath** (1908): the canonical Victorian translation—archaic and formal, with constructions in flowing prose and few explicitly named variables.
- **Fitzpatrick** (2007): a modern academic edition that names points and segments explicitly ("the straight line AB," "the triangle ABC"), closer in surface form to a formal statement.

- **Modern**: a plain contemporary paraphrase that strips the archaism without Fitzpatrick's heavy variable naming.

All three say the same things; they differ only in how they say them. The question is whether GPT-4o formalizes one register into Lean more successfully than another.

We build on the **LeanEuclid** benchmark of Murphy et al. (2024), which provides a formal vocabulary, a system semantics ("E3") for Euclidean geometry in Lean, and reference formalizations of Book I. Our infrastructure—prompt construction, model orchestration, compilation, and semantic checking—is implemented in the **Youklid** toolchain.

2 Method

2.1 Factors and design

The study crosses two factors fully:

- **Translation register** (the input text): Fitzpatrick, Heath, or Modern.
- **Prompt style** (the register of the five worked examples embedded in the few-shot prompt): Fitzpatrick-style, Heath-style, or Modern-style demonstrations.

This 3×3 design formalizes every translation under every prompt style, separating the effect of *what you feed in* from the effect of *how you frame the request*. It covers all 48 propositions of Book I and is replicated on 100 theorems from the UniGeo geometry corpus—143 propositions in total, for **2,653 individual runs**.

2.2 Outcome measures

The primary outcome is the *compile rate*: the fraction of runs whose generated Lean type-checks against the LeanEuclid environment. Compilation is necessary for a usable formalization but not sufficient—a proof can compile while stating the wrong theorem. We therefore also report two stricter semantic measures on the subset scored with the E3 equivalence checker: *strict equivalence* (the generated statement is logically equivalent to the reference) and *correct-direction* entailment (it implies, or is implied by, the reference). These are discussed in Section 7; the headline effects in Section 4 are measured on compilation.

2.3 Models and platform

The counterbalanced study uses **GPT-4o** via the OpenAI API. An earlier exploratory phase used a small local model, **Qwen2.5-Coder-7B** (Q4_K_M quantization) on an NVIDIA Jetson Orin Nano Super (8 GB, roughly 9.5 tokens/s). As Section 5 explains, those runs were discarded after a contamination probe, and only GPT-4o results enter the main analysis.

3 The confound, and the fix

The study began with a single prompt template whose five worked examples were all in Fitzpatrick's style. Under that template, Fitzpatrick input appeared to formalize best, beating Heath by 8.4 percentage points ($p = 0.008$)—a clean-looking "wording matters" result.

It was an artifact. The Fitzpatrick *translation* also matched the register of the *examples in the prompt*, confounding two explanations: perhaps Fitzpatrick prose is genuinely easier to formalize, or perhaps the model simply does better when the input resembles its demonstrations. A single template cannot distinguish them.

The counterbalanced 3×3 design exists to break this confound. Pairing each translation with each prompt style lets the main effect of translation be estimated with prompt style held fixed, and vice versa.

4 Results

4.1 Prompt style dominates; translation does not

How you prompt matters; which translation you feed in does not. Across prompt styles the compile rate swings over a **24-point range**—72.6% with the best style versus 48.4% with the worst ($p < 0.001$). With prompt style held fixed, translation register has no significant effect ($p = 0.603$).

Figure 1 shows the full 3×3 table of compile rates for Book I, and the structure is the whole story. Read *down* any column—prompt style fixed, translation varying—and the three rates cluster within a few points. Read *across* any row—prompt style changing, translation fixed—and the rates fall off a cliff, from the Fitzpatrick-prompt column to the Modern-prompt column.

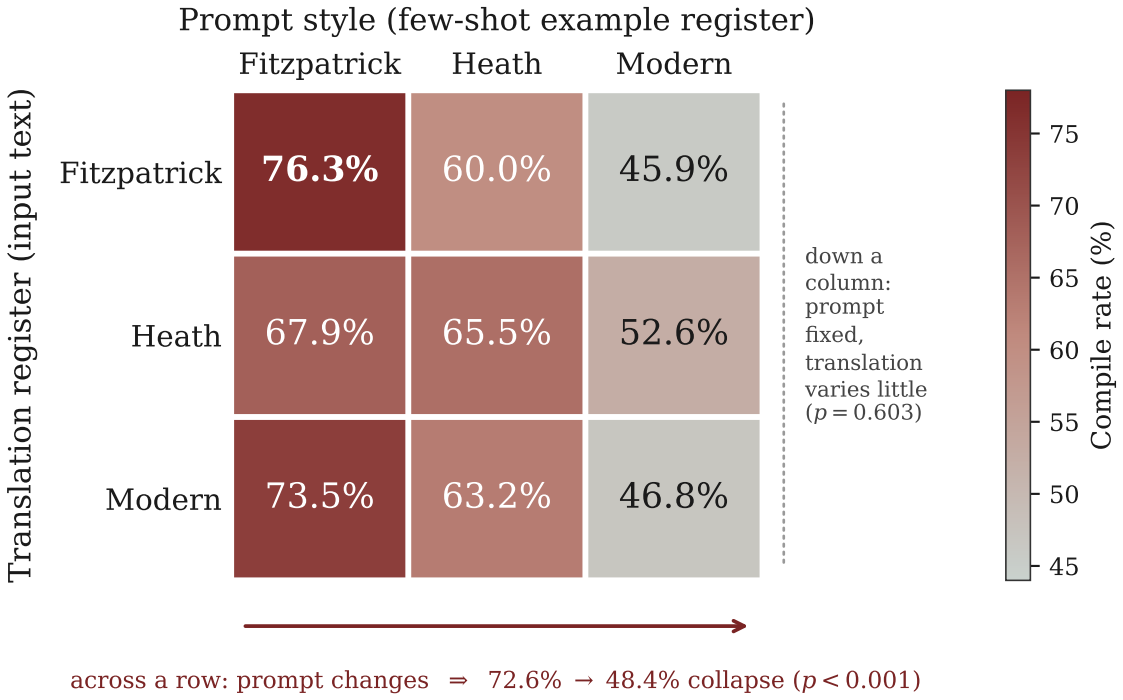

Figure 1: Compile rate by translation register (rows) × prompt style (columns) for Book I under GPT-4o. Color encodes compile rate. The rate collapses *across* each row (changing the prompt) but is nearly flat *down* each column (changing the translation), the visual signature of a strong prompt effect and a negligible translation effect.

Table 1: Compile rate by translation × prompt style (Book I, GPT-4o).

	Fitzpatrick prompt	Heath prompt	Modern prompt
Fitzpatrick	**76.3%**	60.0%	45.9%
Heath	67.9%	65.5%	52.6%
Modern	73.5%	63.2%	46.8%

4.2 The original gap lived in one column

The earlier "Fitzpatrick wins" gap appears in Table 1 only within the Fitzpatrick-prompt column (76.3% vs. Heath's 67.9%). Under Heath-style prompts it reverses—Heath 65.5% exceeds Fitzpatrick 60.0%—and under Modern-style prompts it reverses again (52.6% vs. 45.9%). The apparent advantage of Fitzpatrick text was an advantage of Fitzpatrick *prompts*, not Fitzpatrick *prose*. The independent UniGeo replication agrees: there Heath beat Fitzpatrick by 6.2 percentage points, the opposite of the original claim.

4.3 Statistical model

A mixed-effects linear regression with proposition as a random intercept ($N = 2{,}653$) confirms the pattern: translation coefficients are tiny and non-significant, prompt coefficients large and significant (Table 2).

Table 2: Mixed-effects model coefficients (random intercept per proposition, $N = 2{,}653$). Negative prompt coefficients are relative to the Fitzpatrick-style prompt baseline.

Predictor	Coefficient	SE	p
Translation (Fitzpatrick vs. Heath)	+0.010	0.019	0.603
Translation (Modern vs. Heath)	+0.009	0.021	0.665
Prompt (Heath vs. Fitzpatrick)	**−0.097**	0.022	$<$ **0.001**
Prompt (Modern vs. Fitzpatrick)	**−0.242**	0.024	$<$ **0.001**

Adding prompt style improves fit by 99.5 AIC points; adding a prompt × translation interaction does *not* help (AIC rises by 4.3), so the prompt effect is broadly uniform across translations rather than concentrated in particular pairings. A complementary paired t-test on per-proposition rates ($N = 66$ items, marginalized over prompt) leaves the two translations indistinguishable: Heath 70.0% versus Fitzpatrick 62.3%, $t(65) = -1.66$, $p = 0.102$, Cohen's $d = -0.20$.

5 The contamination caveat

Before settling on GPT-4o we evaluated a small local model, Qwen2.5-Coder-7B, for cost and to test feasibility on edge hardware. The results were anomalous: its *best* compile rate came from giving it no mathematics at all—just the stub "Proposition N."

To diagnose this we ran a contamination probe, feeding the model four kinds of input and measuring compile rate and how much LeanEuclid-specific vocabulary the output reused (Figure 2, left). Whether given the full Heath text, an empty stub, a version with its mathematical vocabulary garbled, or one with its clauses shuffled, the model compiled at roughly 60–63% and reused 100%

of the benchmark vocabulary—the empty stub scoring *highest* (62.8%). A model that produces compiling, benchmark-flavored Lean from an empty prompt is reciting memorized structure, not reading the mathematics.

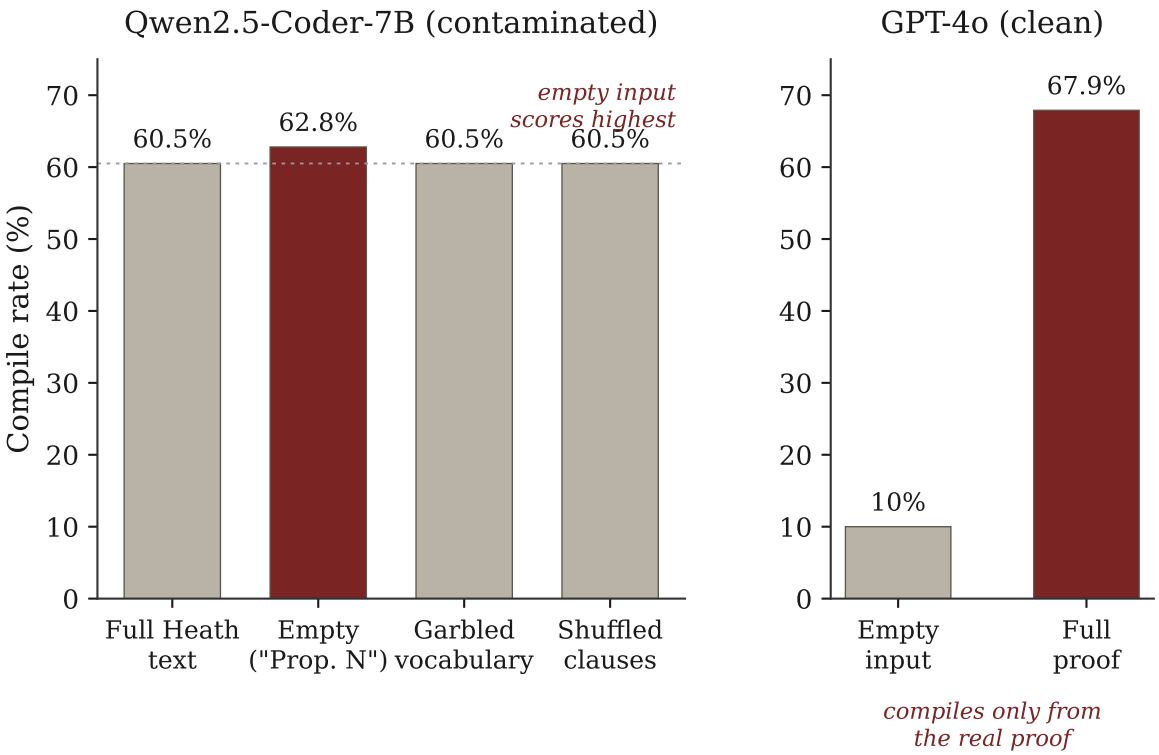

Figure 2: Contamination probe. Left: Qwen2.5-Coder-7B compiles at ∼60% regardless of what input it is given, peaking on an *empty* prompt—a memorization signature. Right: GPT-4o behaves correctly, compiling only 10% of the time from empty input but 67.9% from the full proof.

Semantic equivalence does respond to the input even for Qwen: it drops to 0% on empty input versus 16–27% with garbled text. The model reads the input to refine *meaning*; it just does not need the input to produce something that *compiles*. GPT-4o, by contrast, is clean (Figure 2, right): 10% compile rate from empty input versus 67.9% from the full proof, the expected dependence of output on input. We therefore discarded all Qwen runs and ran the counterbalanced study entirely on GPT-4o.

6 Discussion

The practical lesson for anyone applying autoformalization to their own mathematics is counterintuitive. The instinct is to clean up the *prose*—to rewrite statements in some canonical, model-friendly register before formalizing them. Our results say the prose is not where the gains are. They come from the *prompt*: from styling the few-shot demonstrations to match how the target mathematics is written, and especially in whether the examples introduce variables explicitly. The strongest configuration we found pairs any translation with Fitzpatrick-style demonstrations, whose explicit naming of points and segments evidently scaffolds the model toward well-typed Lean regardless of how the input proposition is phrased.

Two methodological points generalize beyond Euclid. First, a single prompt template can manufacture a spurious "wording" effect: any property of the input that happens to correlate with the prompt's examples will masquerade as a property of the input. Counterbalanced prompt designs, crossing input conditions against prompt conditions, should be the default in autoformalization studies. Second, the empty-input probe—does the model compile from nothing but "Proposition N"?—is a cheap, decisive memorization check. It cost almost nothing and would have invalidated the Qwen numbers before they were ever reported. We recommend it as a routine check before publishing autoformalization compile rates, especially for models that may have seen the benchmark in training.

7 Limitations

- **One model.** The counterbalanced results are GPT-4o only; the Qwen runs are void due to contamination. Prompt sensitivity, and the relative ranking of prompt styles, may differ for other models.
- **Compiling is not correctness.** Compile rates overstate real performance, because a proof can type-check while formalizing the wrong claim. On the subset scored with the E3 semantic checker, only 14% of outputs were strictly equivalent to the reference and 30% were correct in direction. The 24-point prompt effect is an effect on *compilation*, not semantic correctness; whether prompt style moves correctness as strongly is not established here.
- **Uneven cells.** The counterbalanced cells are not equally sized: Fitzpatrick-prompt cells carry about 430 runs each, while Heath- and Modern-prompt cells carry roughly 133–235. The mixed-effects model accommodates the imbalance, but precision varies across cells, and some Book I cells reached only about 45% of their target run counts for the Heath and Modern prompts.
- **UniGeo, one prompt.** The 100-theorem UniGeo replication used only Fitzpatrick-style prompts. It corroborates the *translation* non-effect (and the reversal of the original claim) but does not, on its own, speak to the prompt effect.
- **Single semantic checker, partial coverage.** E3 semantic equivalence was scored most completely for the Heath condition. Full cross-condition semantic scoring remains future work, as does extending the study to additional models and to a two-stage pipeline (a strong model to draft, a cheaper model prompted with human-checked exemplars to autoformalize).

References and credits

- L. Murphy, K. Yang, J. Sun, Z. Li, A. Anandkumar, X. Si. *Autoformalizing Euclidean Geometry* (LeanEuclid benchmark and E3 semantics), 2024. github.com/loganrjmurphy/LeanEuclid.
- Youklid: autoformalization toolchain used for prompt construction, orchestration, compilation, and semantic checking. github.com/Zed-Rez/youklid.
- T. L. Heath, *The Thirteen Books of Euclid's Elements*, Cambridge University Press, 1908.
- R. Fitzpatrick (ed.), *Euclid's Elements of Geometry*, 2007.
- Contamination probe and local inference ran on an NVIDIA Jetson Orin Nano Super.