Automatic (In)formalization of Mathematics via Language-Model Probabilistic Programming and Cycle Consistency

Fabian Zaiser, Mauricio Barba da Costa, Katherine Collins, Timothy O'Donnell, Alexander Lew, Joshua Tenenbaum, Vikash Mansinghka, Cameron Freer

Massachusetts Institute of Technology

EuroProofNet

Outline

- Introducing autoformalization as inference
- Preliminary experiments and simple case studies

Previous talk

- Useful constraints/signals for autoformalization?
- How to systematically combine these ingredients?
- Preliminary experimental results

This talk

Formalizing Mathematics in Lean

Quanta magazine | Physics | Mathematics | Biology | Computer Science | Topics | Archive

PROOFS

Proof Assistant Makes Jump to Big-League Math

nature | View all journals | Search | Log

Publish with us

s welcome computer-assisted 'unification' theory

bstract concept at the cutting edge of research, revealing a bigger role for

Mathematics

Mathematicians plan computer proof of Fermat's last theorem

Fermat's last theorem puzzled mathematicians for centuries until it was finally proven in 1993. Now, researchers want to create a version of the proof that can be formally checked by a computer for any errors in logic

By Alex Wilkins

18 March 2024

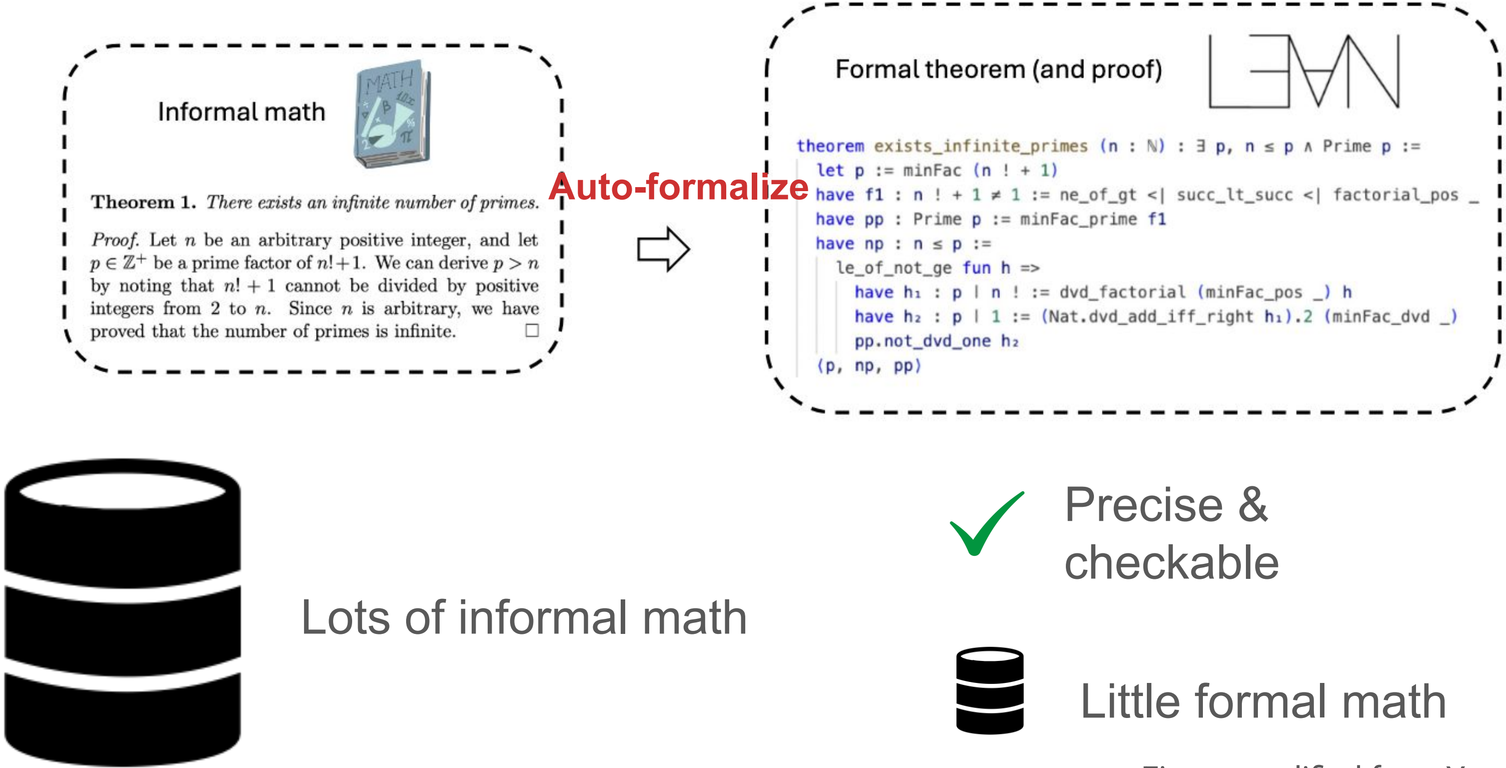


Figure modified from Yang et al., 2024

Challenges of (in)formalization

If x is an element of infinite order in G, prove that the elements x^n are all distinct.

Proof. ...

formalize →

← informalize

```
theorem formal
  (G : Type*) [Group G] (x : G)
  (h : ∀ n : ℕ, x ^ n ≠ 1)
  : ∀ m n : ℤ, m ≠ n → x ^ m ≠ x ^ n
  := by ... -- proof
```

Challenge	**Example**
Nontrivial syntax	`Type*` is custom syntax defined in mathlib
Implicit types	`n` is implicitly a natural number / integer
Implicit algebraic structure	`G` is implicitly a group
Informal shorthand	*"infinite"* order → formalized without infinity
Language and library evolution	`group` (mathlib 3) → `Group` (mathlib 4)

Probabilistic Viewpoint (cf. previous talk)

- **Probabilistic reasoning** is key in formal math
  - **disambiguate** shorthand and overloaded statements
  - **infer** missing steps and assumptions
  - **fix** errors and statements that are helpful but not literally correct
  - fuse **hard constraints** (parsing, type-checking, validity) with **soft signals** (plausibility, clarity, style)
  - calibrate **uncertainty** and quantify **confidence**
- Language models give us a distribution over text/code
- But we want to tweak this distribution (e.g. enforce Lean 4 instead of Lean 3 code)
- Lack of data → leverage compute at inference time

LM generation samples f

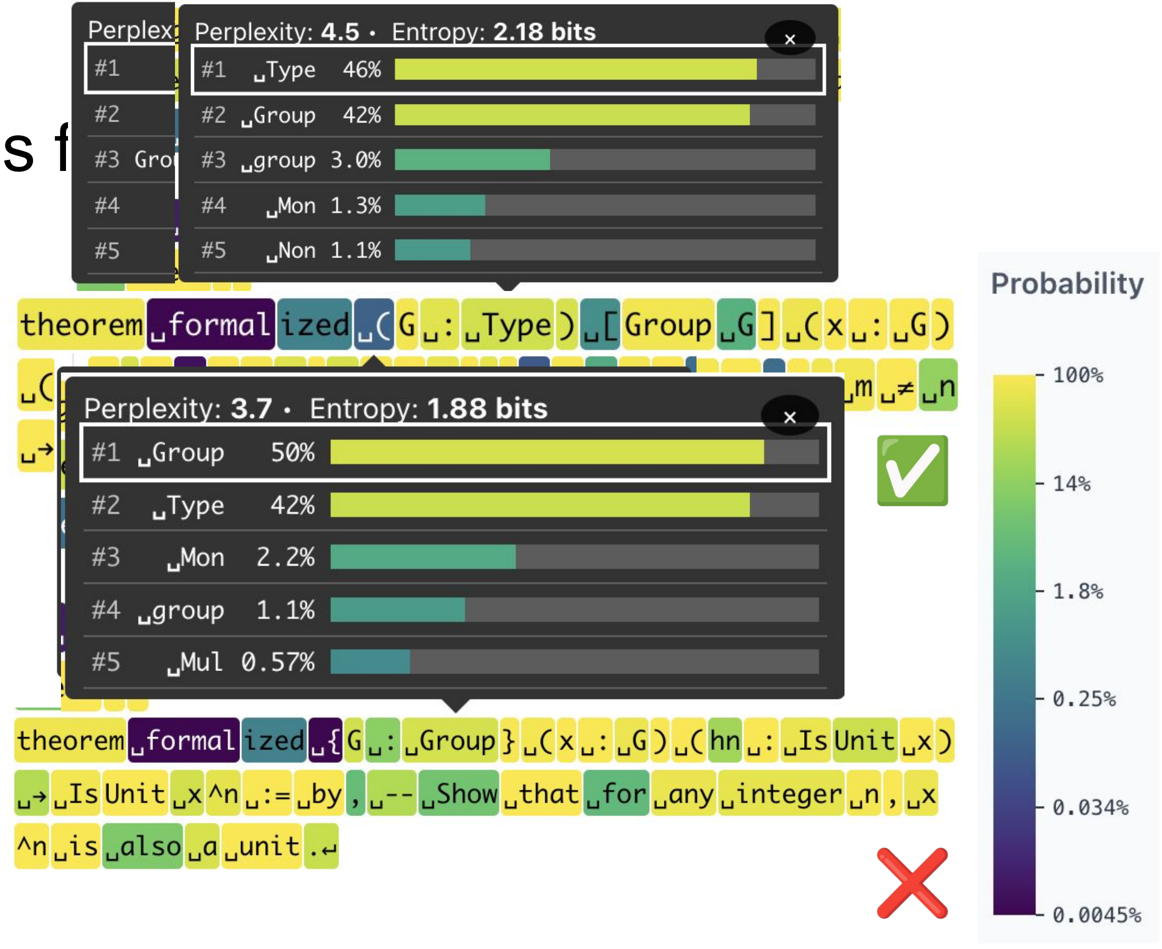
Prompt:

Natural language: *If x is an element of infinite order in G, prove that the elements x^n are all distinct.*

Lean 4 translation:

```
```lean4
import Mathlib

theorem formalized...
```



→ want to **constrain/condition** the distribution of texts
```

Autoformalization as Bayesian Inference

$$\underbrace{\mathbb{P}[\,text \mid constraint\,]}_{\text{Posterior}} = \frac{\mathbb{P}[text] \cdot \mathbb{P}[\,constraint \mid text\,]}{\mathbb{P}[constraint]} \propto \underbrace{\mathbb{P}[text]}_{\text{Prior}} \cdot \underbrace{\mathbb{P}[\,constraint \mid text\,]}_{\text{Likelihood/Potential}}$$

- **Prior** distribution: completions of an LM prompted to formalize a given statement
- **Conditioned** on **likelihood/potential**:
  - Hard constraints (0 or 1)
    - Syntactically valid?
    - Well-typed?
    - Provable?
  - Soft constraints (continuous signal)
    - Plausibility
    - Style
    - Embedding distances (see previous talk)
- **Posterior** distribution: “better” formalizations

Potentials for conditioning

Potentials: Likelihood function P(constraint | text)

Binary potentials (hard conditioning): `φ: Text → {0,1}`

- Is the generated code syntactically valid?
- Does it type check?
- Linter warnings?
- Can a counterexample be found? (cf. `plausible` tactic in previous talk)

Continuous potentials (soft conditioning): `φ: Text → [0,1]`

- How does another LM score the output?
- Cycle consistency (later in the talk)

Want to sample from **prior *reweighted* by potentials**:

$$\mathbb{P}[\,text \mid constraint\,] \propto \underbrace{\mathbb{P}[text]}_{\text{Prior}} \cdot \underbrace{\mathbb{P}[\,constraint \mid text\,]}_{\text{Likelihood/Potential}}$$

Language-Model Probabilistic Programming

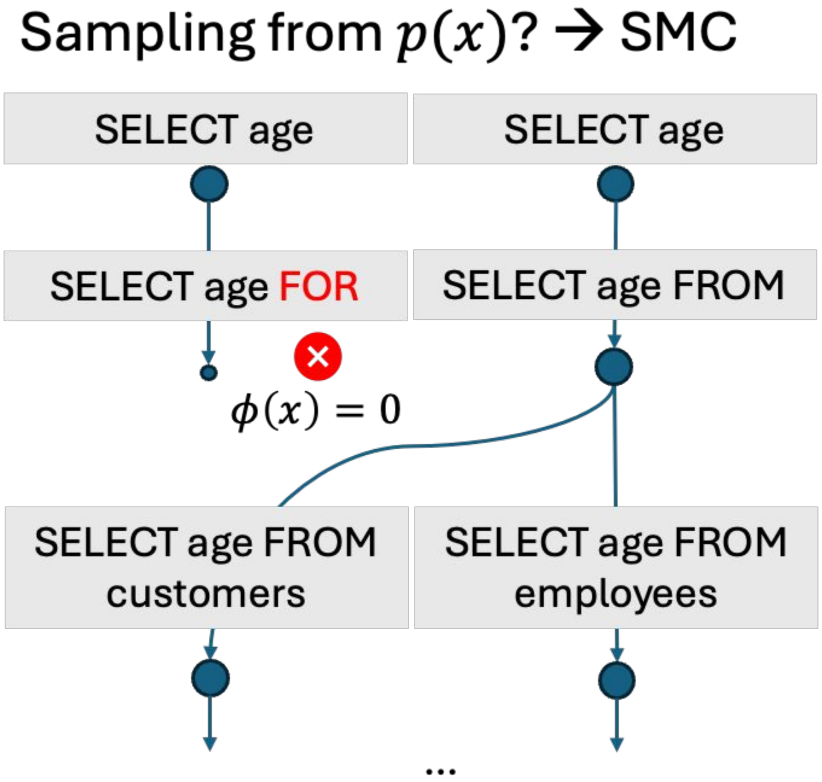
Task: generate text/code satisfying constraints

→ e.g.: valid SQL query

LM probabilistic programming approach: [Loula et al., ICLR 2025]

- Prior: $p_{LM}(x \mid \text{prompt})$
- Likelihood: $\phi : \text{Text} \to \mathbb{R}_{\geq 0}$ (“potential”)
  - Hard constraints: syntax checker, linter, test suite, …
  - Soft constraints: critique by another LM, …
- Posterior:

$$p(x) \propto p_{LM}(x \mid \text{prompt}) \cdot \phi(x)$$



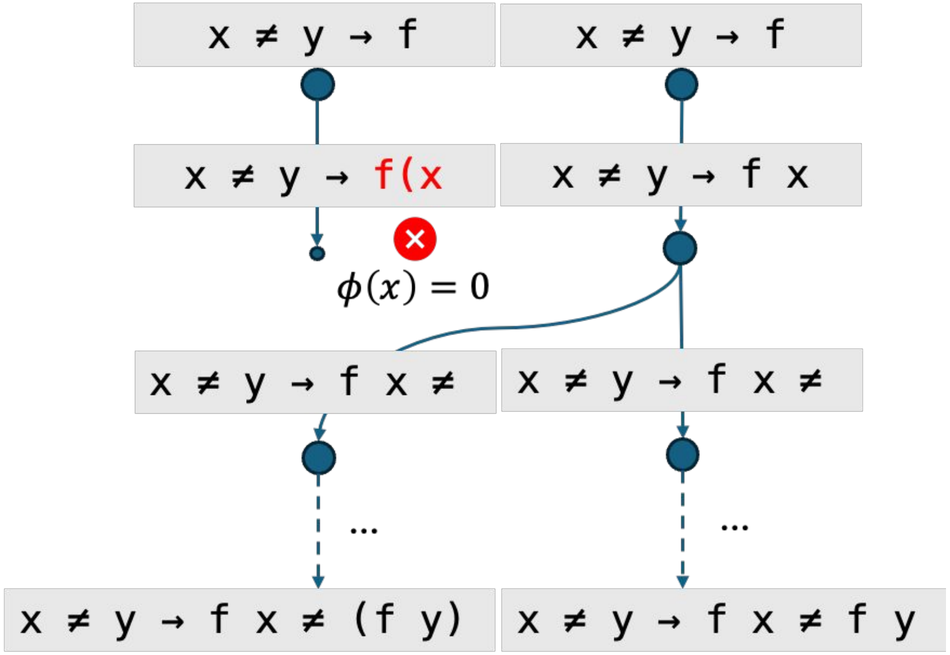
Symbolic constraints for Lean

Potentials:

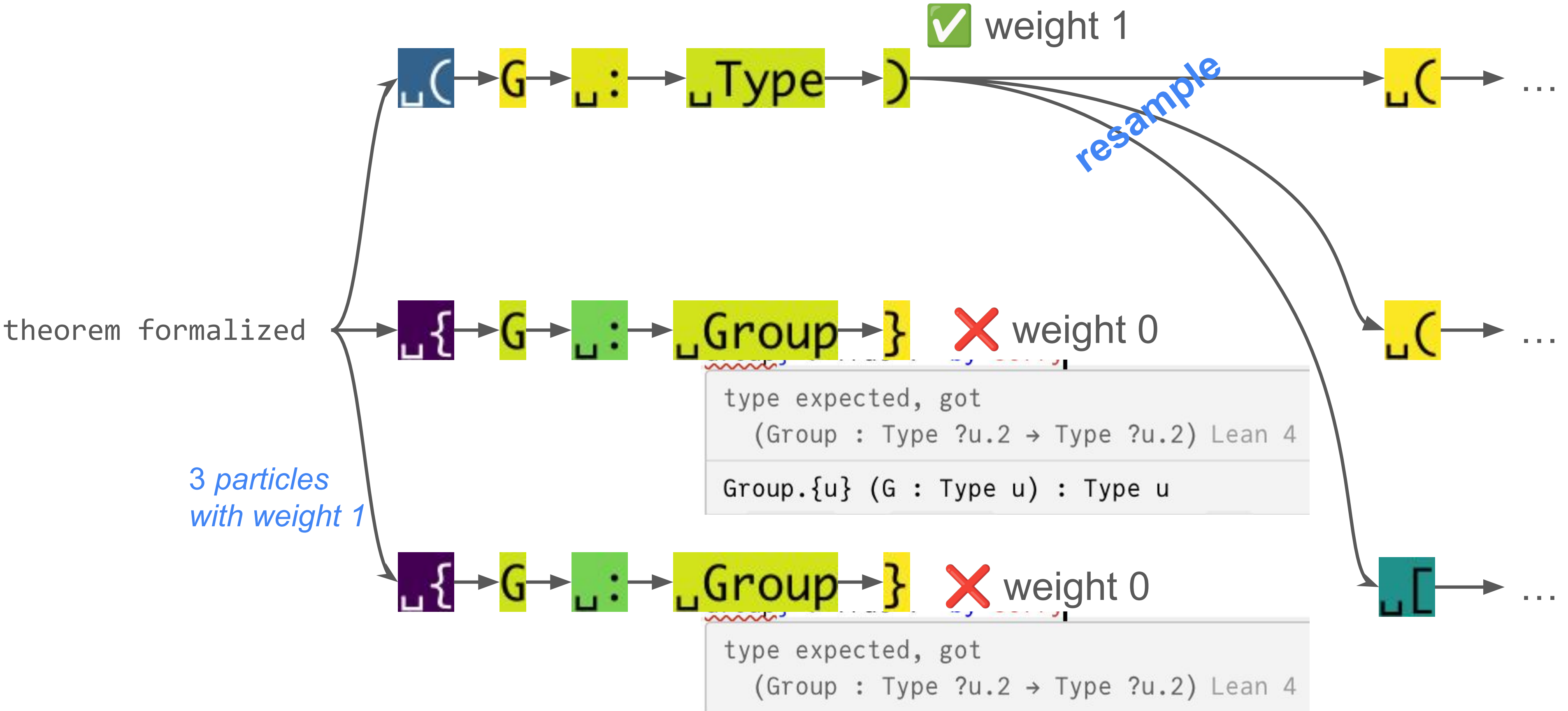
- Syntactic correctness
- Type correctness
- Counterexample generation
- Validity of proof steps

Challenge: Is given Lean code a prefix of a correct formalization?

Sequential Monte Carlo with syntax check



Sequential Monte Carlo for Bayesian Inference



Incremental type checking

Run Lean parser & type checker?

Prefixes are rarely valid Lean

```
1 import Mathlib
2
3 theorem
```

```
unexpected end of input;
```

Ignore errors with a location at the end of the input?

Error location unreliable

```
wlog hmn' : m < n generalizing m n
· symm
  apply this
```

```
unsolved goals
```

Check validity after each parameter, but adding a “dummy” conclusion

```
3 theorem formal (G : Type*) [Group G]
```

```
unexpected end of input;
```

```
3 theorem formal (G : Type*) [Group G] : True := by trivial
```

Being well-typed does not guarantee correctly matching intent

Problem 6: Munkres|exercise_17_4

Informal Statement:

Show that if U is open in X and A is closed in X, then $U-A$ is open in X, and $A-U$ is closed in X.

All Formalizations:

[GENLEAN 1] ✗ INCORRECT
Formalization: `theorem open_diff_closed_eq_open_sub_closed {X : Type*} [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : ∃ (U' : Set X), IsOpen U' ∧ U' = U \ A := by sorry`

[GENLEAN 2] ✓ CORRECT
Formalization: `theorem open_diff_closed_eq_open_sub_closed {X : Type*} [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : IsOpen (U \ A) ∧ IsClosed (A \ U) := by sorry`

[GENLEAN 3] ✗ INCORRECT
Formalization: `theorem open_sub_sub_is_open (X : Type*) [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : IsOpen (U \ A) := by sorry`

[GENLEAN 4] ✗ INCORRECT
Formalization: `theorem open_diff_closed_eq_open_sub_closed {X : Type*} [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : ∃ (U' : Set X), IsOpen U' ∧ U' = U \ A ∧ IsClosed (A \ U) := by sorry`

[BASELINE 1] ✗ INCORRECT
Formalization: `theorem open_sub_open_isOpen [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : IsOpen (U \ A) := by sorry`

[BASELINE 2] ✗ INCORRECT
Formalization: `theorem open_sub_isOpen [TopologicalSpace X] {U : Set X} (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : IsOpen (U \ A) := by sorry`

[BASELINE 3] ✗ INCORRECT
Formalization: `theorem open_sub_equal [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : IsOpen (U \ A) := by sorry`

[BASELINE 4] ✗ INCORRECT
Formalization: `theorem open_sub_closed_eqs {X : Type*} [TopologicalSpace X] (U : Set X) (hU : IsOpen U) (A : Set X) (hA : IsClosed A) : ∅ = U \ A ∧ A \ U = ∅ := by sorry`

Correct formalizations can vary in wording or details

Problem 5: Dummit-Foote|exercise_7_1_15

Informal Statement:
A ring R is called a Boolean ring if $a^{2}=a$ for all $a \in R$. Prove that every Boolean ring is commutative.

All Formalizations:

[GENLEAN 1] ✓ CORRECT
Formalization: `theorem isBoolean_ring_of_coeq_mul {R : Type*} [Ring R] (h : ∀ a : R, a ^ 2 = a) : IsCommutative R := by sorry`

[GENLEAN 2] ✓ CORRECT
Formalization: `theorem comm_of_bool_ring {R : Type*} [Ring R] (h : ∀ a : R, a ^ 2 = a) : IsCommutative R := by sorry`

[GENLEAN 3] ✓ CORRECT
Formalization: `theorem isBoolean_ring_of_coeq_mul {R : Type*} [Ring R] (h : ∀ a : R, a * a = a) : IsCommutative R := by sorry`

[BASELINE 1] ✗ INCORRECT
Formalization: `theorem is_bool_ring_commutative (R : Type*) [Ring R] : IsCommutative R := by sorry`

For some problems, one key detail is often incorrect

Problem 9: Munkres|exercise_27_4

Informal Statement:

Show that a connected metric space having more than one point is uncountable.

All Formalizations:

[GENLEAN 1] ✗ INCORRECT
Formalization: `theorem is_connected_iff_uncountable_of_nontrivial (X : Type*) [TopologicalSpace X] (hX : ConnectedSpace X) (hX' : Infinite X) : Uncountable X := by sorry`

[GENLEAN 2] ✗ INCORRECT
Formalization: `theorem uncountable_iff_connected_nontrivial (X : Type*) [MetricSpace X] [ConnectedSpace X] : Uncountable X := by sorry`

[GENLEAN 3] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable (X : Type*) [TopologicalSpace X] (hX : ConnectedSpace X) : Uncountable X := by sorry`

[GENLEAN 4] ✓ CORRECT
Formalization: `theorem connected_iff_uncountable_in_nontrivial (X : Type*) [MetricSpace X] (hX : ConnectedSpace X) (hX' : Nontrivial X) : Uncountable X := by sorry`

[GENLEAN 5] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable {X : Type*} [TopologicalSpace X] (h : ConnectedSpace X) : Uncountable X := by sorry`

[GENLEAN 6] ✗ INCORRECT
Formalization: `theorem connected_uncountable_iff_nontrivial (X : Type*) [TopologicalSpace X] (h : ConnectedSpace X) : Uncountable X := by sorry`

[BASELINE 1] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable_of_nontrivial {X : Type*} [TopologicalSpace X] (hX : ConnectedSpace X) (hX' : Infinite X) : Uncountable X := by sorry`

[BASELINE 2] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable_in_nonempty (X : Type*) [TopologicalSpace X] (hX : ConnectedSpace X) (h : Nonempty X) : Uncountable X := by sorry`

[BASELINE 3] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable (X : Type*) [MetricSpace X] (h : ConnectedSpace X) : Uncountable X := by sorry`

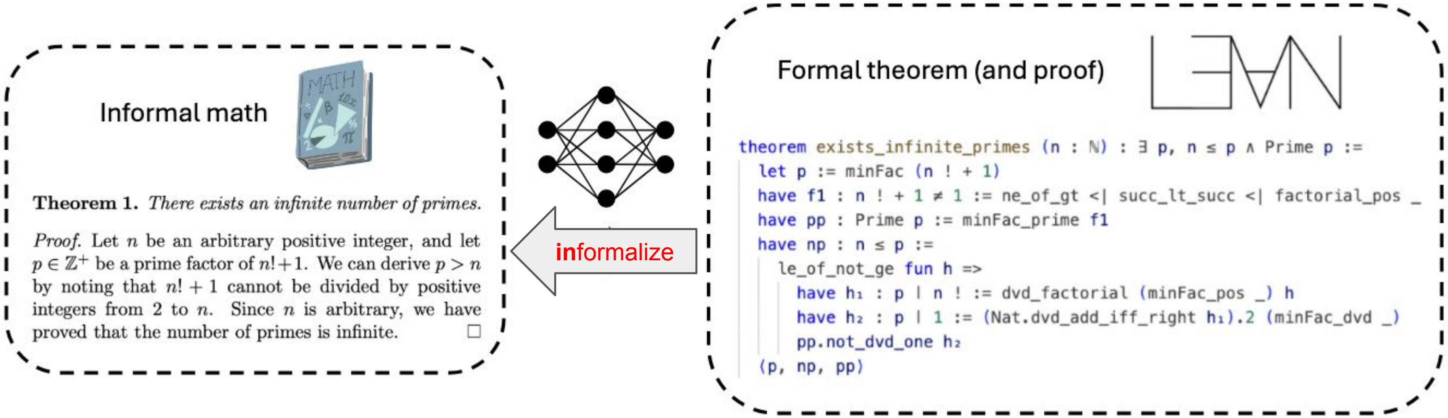
[BASELINE 4] ✗ INCORRECT
Formalization: `theorem connected_iff_uncountable (X : Type*) [TopologicalSpace X] (hX : ConnectedSpace X) : Uncountable X := by sorry`

Assessing Content, Not Just Form

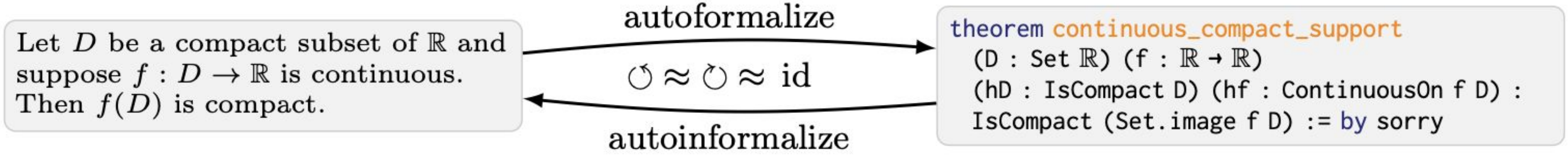
Well-typed potential checks the *"shape"* of the Lean code.

But how do we assess the *content* (i.e. whether it matches the informal meaning)?

Idea: Translating in the other direction (*Informalization*) is easier for LMs!



Cycle-consistency constraint



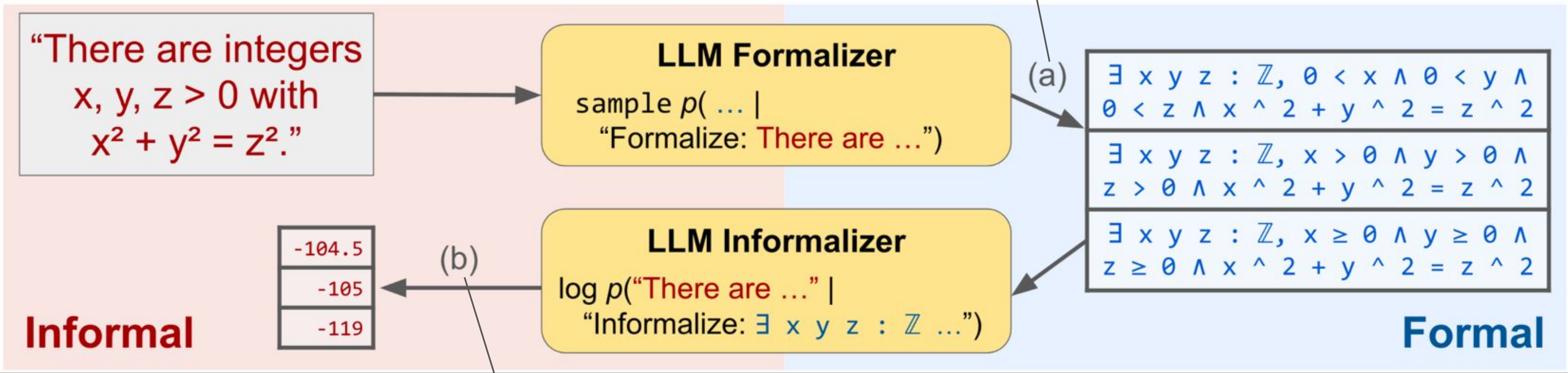
One might try to enforce the constraint that the round trip (*informal* → *formal* → *informal*) should be approximately the identity map.

- Long history in *natural language translation* since the 1950s
- Proposed for *autoformalization* [Szegedy, CICM 2020]
- Related ideas ("distilled backtranslation") have been implemented [Azerbayev et al., MATHAI@NeurIPS 2022]

(Example (in)formal text from Lean Workbook, Ying et al., NeurIPS 2024)

Cycle consistency potential

(a) samples from the distribution of an LLM which is prompted to formalize the input



(b) scores the formalizations according to the log-probability that the LLM generates the original input text when it is prompted to informalize the given formalization

How to measure the quality of the backtranslation?

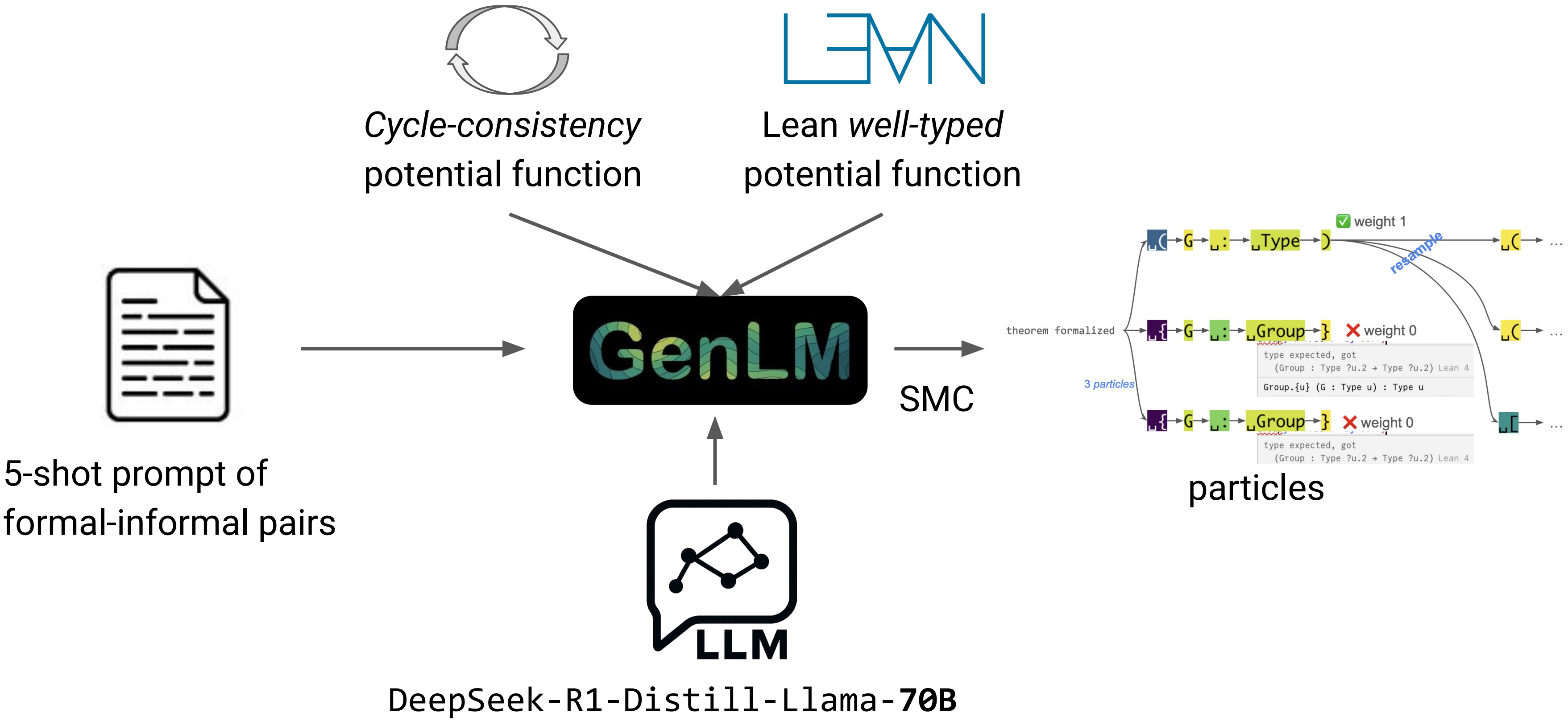
- $\log p_{LM}$(original informal stmt | "Informalize: {formalization}")?
    - May capture other signals in the probability than truth value
- Cosine similarity of embeddings of ground truth and back translation? (cf. [Li et al., NeurIPS2024])
    - cheap, imprecise
- Ask an LM to rank back-translated informalizations?
    - Higher quality, expensive

Rating formalizations – preliminary results

Lean 4 formalization	**Correct?** (manual labeling)	Cosine similarity in **embedding** space in [0, 1]	**LM logprob for informalization*** in [-inf, 0]	**LM rater** (Claude Sonnet 4) in [0, 1]
∃ x y z : ℤ, 0 < x ∧ 0 < y ∧ 0 < z ∧ x ^ 2 + y ^ 2 = z ^ 2	✅	0.9612	-104.5	1.0
∃ x y z : ℤ, x > 0 ∧ y > 0 ∧ z > 0 ∧ x ^ 2 + y ^ 2 = z ^ 2	✅	0.9657	-105	1.0
∃ x y z : ℤ, x ≥ 0 ∧ y ≥ 0 ∧ z ≥ 0 ∧ x ^ 2 + y ^ 2 = z ^ 2	❌	0.9601	-119	0.8

$^{*}\ \log p_{LM}(\text{original informal stmt} \mid \text{“Informalize: \{formalization\}“})?$

GenLean: GenLM with Lean *well-typed* potential function



Prompt template

(adapted from Poiroux et al., 2024)

1. Natural language example 1
   Formal translation of example 1
2. Natural language example 2
   Formal translation of example 2
3. Natural language example 3
   Formal translation of example 3
4. Natural language example 4
   Formal translation of example 4
5. Natural language statement to formalize
   Formal translation: ... (to be completed)

```
Natural language version:
	Let $X$ be a topological space; let $A$ be a subset of
	containing $x$ such that $U \subset A$. Then $A$ is op
Translate the natural language version to a Lean 4 version:
	theorem formalized (X : Type*) [TopologicalSpace X] (A
	⊆ A): IsOpen A := by sorry

Natural language version:
	If $z_1, \ldots, z_n$ are complex, then $|z_1 + z_2 +
Translate the natural language version to a Lean 4 version:
	theorem formalized (n : ℕ) (f : ℕ → ℂ) : abs (∑ i in
	sorry

Natural language version:
	If $x$ is an element of infinite order in $G$, prove t
Translate the natural language version to a Lean 4 version:
	theorem formalized (G : Type*) [Group G] (x : G) (hx_i
	:= by sorry

Natural language version:
	A set of vectors $\{v_i\}_{i \in I}$ orthogonal with r
	independent if for all $i \in I$, $B(v_i, v_i) \neq 0$
Translate the natural language version to a Lean 4 version:
	theorem formalized {V K : Type*} [Field K] [AddCommGro
	(hv1 : B.IsOrtho v) (hv2 : ∀ (i : n), ¬B.IsOrtho (v i

Natural language version:
	[informal statement to be formalized]
Translate the natural language version to a Lean 4 version:
	theorem formalized ...
```

Preliminary Experiments

- **Benchmark:** 50 informal/formal pairs from `miniF2F` (ICLR 2022)
- **LM:** `DeepSeek-R1-Distill-Llama-70B`
- **Particles for SMC:** 10 (for each combination of potentials)
- **Judge:** Claude Sonnet 4
  - prediction to be ✓ CORRECT / ✗ INCORRECT based on **matching intent of informal text**
  - ~97% agreement with manual labels
- **Evaluation metric:** weighted average of each particle's score
  - ✗ INCORRECT = 0%
  - ✓ CORRECT = 100%

If x, y, and z are positive numbers satisfying x + 1/y = 4, y + 1/z = 1, and z + 1/x = 7/3 then x y z = 1.

```
theorem amc12_2000_p20
  (x y z : ℝ)
  (h₀ : 0 < x ∧ 0 < y ∧ 0 < z)
  (h₁ : x + 1/y = 4)
  (h₂ : y + 1/z = 1)
  (h₃ : z + 1/x = 7/3) :
  x*y*z = 1 := by sorry
```

`miniF2F`: informal/formal pair

Constraints	Weighted score average
No constraint	13%
Lean validity	14%
Cycle consistency	13.7%
Both constraints	16.7%

Summary

- Autoformalization as Bayesian inference
  - → language-model probabilistic programming
- Sequential Monte Carlo constrains LM output distribution according to potential functions
  - Lean-validity potential: parses and type checks Lean code incrementally
  - Cycle-consistency potential: score with the quality of the backtranslation
- Preliminary results
  - Each potential improves the performance on miniF2F

Thank you!

Questions?