Mathesis:

Towards Formal Theorem Proving from Natural Languages

Xuejun Yu* Jianyuan Zhong* Zijin Feng* Pengyi Zhai Roozbeh Mohit Wei Chong Ng Haoxiong Liu Ziyi Shou Jing Xiong Yudong Zhou Claudia Beth Ong Austen Jeremy Sugiarto Yaoxi Zhang Wai Ming Tai Huan Cao Dongcai Lu Jiacheng Sun Qiang Xu Shen Xin† Zhenguo Li†

Huawei · The Chinese University of Hong Kong

github.com/Huawei-AI4Math/Mathesis

Problem & Motivation

The Gap

- Real-world math problems are in natural language
- ATPs (DeepSeek-Prover, Kimina, Goedel) require formal statements in Lean 4
- Manual formalization demands deep expertise and significant effort
- Autoformalization errors lead to misleading proofs or unprovable statements
- Semantically equivalent formalizations can vary significantly in proof difficulty

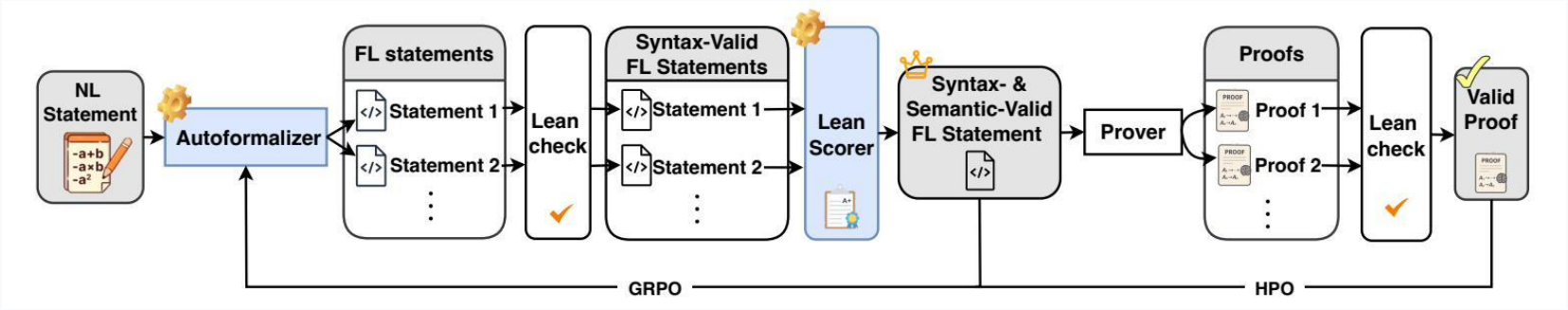
Informal Statement: Let S_n denote the sum of the first n terms of the sequence a_n. Given that $\frac{2S_n}{n} + n = 2a_n + 1$, prove that $\{a_n\}$ is an arithmetic sequence.

Two Formal Statement Cases:

```
theorem case_one (a : ℕ → ℝ)
(ha : ∃ d, ∀ n, a (n + 1) = a n + d) ⇒ Erroneously includes the desired goal in
assumptions
(h : ∀ n, 2 * (∑ i in Finset.range n, a i) / n + n = 2 * a n + 1) :
∃ d, ∀ n, a (n + 1) = a n + d := by sorry

theorem case_two (a : ℕ → ℝ) (S : ℕ → ℝ)
(hS : ∀ (n : ℕ), n ≥ 1, S n = ∑ k in Finset.range n, a k) ⇒
∑ k in Finset.Icc 1 n, a k
(h : ∀ (n : ℕ), n ≥ 1, 2 * S n / (n : ℝ) + (n : ℝ) = 2 * a n + 1) :
∃ (d : ℝ), ∀ (n : ℕ), n ≥ 1, a (n + 1) = a n + d := by sorry
```

Pipeline Overview



1 Autoformalization

NL → Lean 4 candidates
via Mathesis-Autoformalizer
(RL-trained with HPO)

2 Validation

Lean compiler (syntax) +
LeanScorer (semantics)

- Continuous score, not binary
- Decompose the task for fine-grained evaluation

3 Proving

Validated statements →
downstream provers →
machine-verified proofs

Mathesis-Autoformalizer: RL Training

STAGE 1

GRPO + Composite Rewards

Semantic reward R_sem: LLM evaluator judges if meaning is preserved

Syntactic reward R_ver: Lean 4 compiler verifies type validity

Final reward r = R_sem + R_ver (simple but effective)

Optimization GRPO normalizes within-group preferences over G candidates

STAGE 2

DPO for Prover Alignment

Preferences derived from proof success/failure

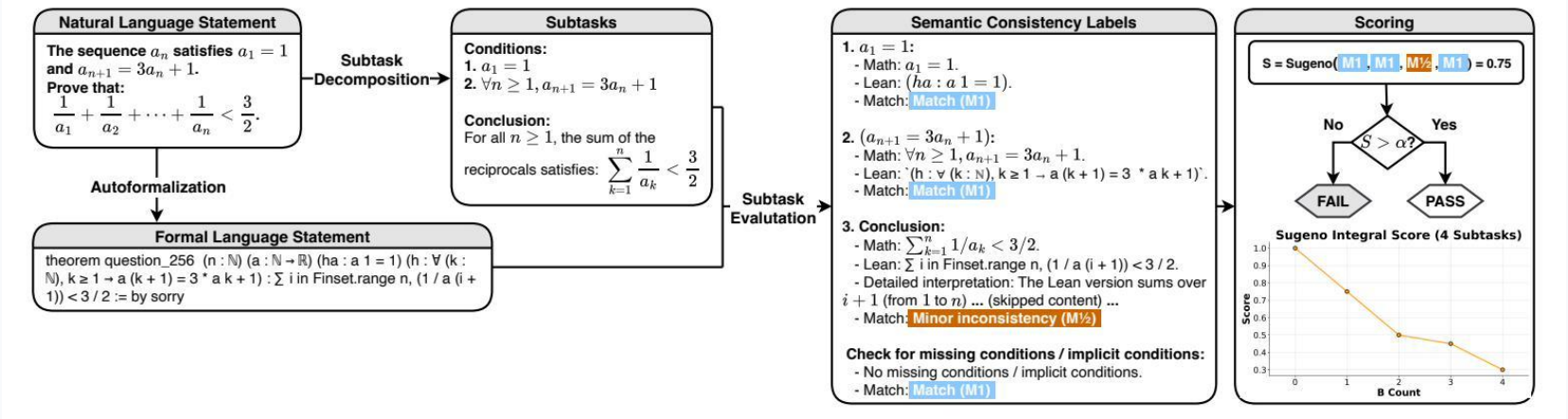
Preferred formalizations leading to Lean-verified proofs

Rejected formalizations where the prover fails

Config β=0.1 KL regularization, single epoch

HPO = GRPO (strong init) → **DPO** (proof-aligned tuning)

LeanScorer: Fine-Grained Semantic Evaluation



Three-Level Labeling

- **M1** Match — exact alignment
- **M½** Minor Inconsistency
- **M0** Major Inconsistency

Aggregated via Sugeno Fuzzy Integral

Evaluation vs Human Annotations

Method	Prec	Rec	F1
LLM-as-a-Judge	73	**100**	0.85
Re-informalization	93	50	0.65
LeanScorer (Ours)	**94**	89	**0.92**

Results: Autoformalization Quality

*LC = Lean Check
LSC = LeanScorer Semantic Check

		MiniF2F-Test		Putnam		Gaokao-Formal	
Model	**k**	**LC**	**LC+LSC**	**LC**	**LC+LSC**	**LC**	**LC+LSC**
API Models							
GPT-4o	6	80	65	24	9	48	28
DeepSeek-V3	6	91	84	22	10	69	56
DeepSeek-R1	6	86	76	58	**37**	81	57
Open-Source 7B							
Herald-AF	6	95	69	64	9	78	27
Kimina-AF	6	100	91	30	10	91	49
Mathesis-HPO	**6**	**100**	**96**	**73**	30	**98**	**71**

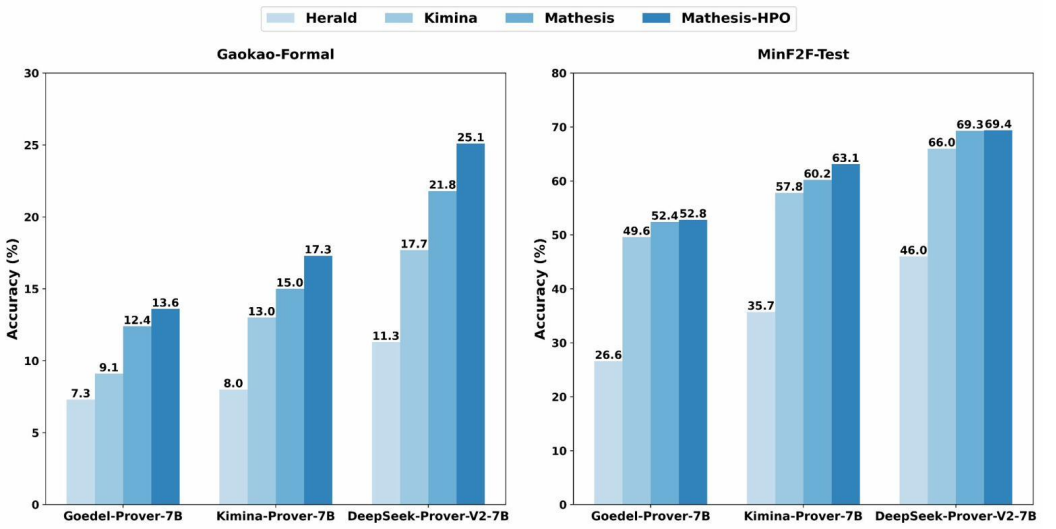
+5 pts
on miniF2F vs Kimina

+20 pts
on Putnam vs Kimina

+22 pts
on Gaokao vs Kimina

7B > API
outperforms GPT-4o, R1, V3

End-to-End Theorem Proving Results



Key Findings

11%-25% Herald → Mathesis-HPO for DSP-V2 on Gaokao

+122%

17%-25% Kimina → Mathesis-HPO for DSP-V2 on Gaokao

+42%

Key Insight

Improving the autoformalizer could yields **larger accuracy gains** than upgrading the prover itself.

Takeaways

01 First systematic study of formal theorem proving from natural language

02 Mathesis-Autoformalizer: first RL-trained autoformalizer with HPO, achieving SOTA

03 LeanScorer: fine-grained semantic evaluation framework (0.92 F1)

04 Gaokao-Formal: new benchmark of 495 complex proof problems

05 Formalization quality is the key bottleneck — more impactful than upgrading the prover

`github.com/Huawei-AI4Math/Mathesis`

Thank you!