Evaluating Program Semantics Reasoning with Type Inference in System *F*

Yifeng He, Luning Yang, Chris Gaw Gonzalo, Hao Chen

UC DAVIS COMPUTER SCIENCE

JC STEM Lab of Intelligent Cybersecurity
賽馬會智能數位安全創科實驗室

NEURAL INFORMATION PROCESSING SYSTEMS

Introduction

Test-test compute (TTC) empowers coding LLMs the ability to reason on programs. However, are LLMs really "reasoning" on the semantics (logic) of the code?

- The evaluation gap for reasoning LLMs
  - Math: competition and answering problems
  - Code: Coding and patch
- How about reasoning about the logic behind math & code?
  - program-centric deductive system

Background: propositions as types

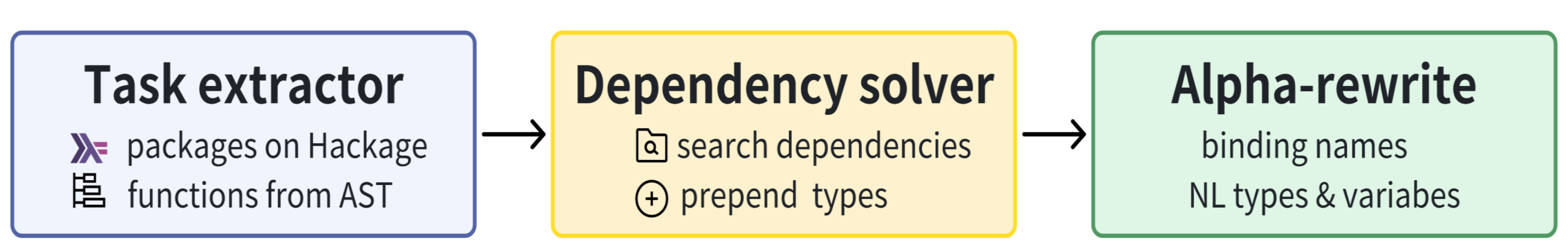
- Types in programming ≅ propositions in logic
- Type inference is natural deduction
- Type inference results are formally verifiable
  - Mutate the tasks, prove the results!

⟹ type inference is a task for *program semantics reasoning*

TF-Bench: benchmark construction

We use the Haskell Prelude to construct TF-Bench

- Formal deductive type system: System F (and System F<).
- The type signature is concise and is decoupled from the function body.
- Type equivalence is formally verifiable (w/ signature only).
- Haskell is the most popular language that meets these above conditions



Pipeline to construct TF-Bench.

Alpha-rewrite: removing natural language from tasks

We design three alpha-rewrite operators to transform the tasks:

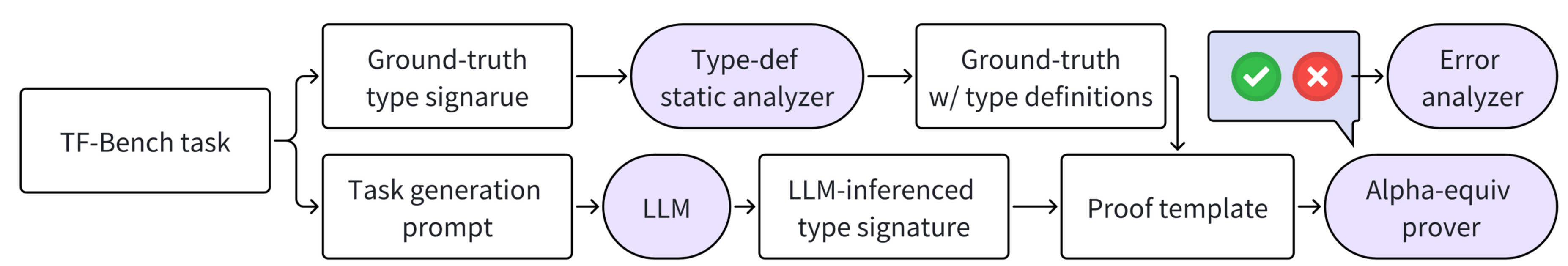
- NL-Type: `Int`, `Char`, `Bool`, `Eq`, `Ord` ... ⟶ `T1`, `T2`, `T3`, `T4`, `T5` ...
- Type-Var: `a`, `b`, `c`, `d`, `e` ... ⟶ `t1`, `t2`, `t3`, `t4`, `t5` ...
- Binding: `map`, `not`, `foldl`, `(+)` ... ⟶ `f1`, `f2`, `f3`, `f4` ...

Implementation details:

- Operators are *communitive* and *associative* under composition
- Transformed tasks *are alpha-equivalent* to the original tasks.

TF-Bench$_{\text{pure}}$ = Alpha-rewrite(TF-Bench) reveals the true program semantics reasoning performance, w/o contamination & relying on NL-cues

Benchmark evaluation



Pipeline to evaluate LLMs on TF-Bench.

1. Add definitions for new types after alpha-rewrite.
2. Prompt the LLMs with tasks to generated a type signature.
3. Construct a proof using ground-truth and generated type signatures.
4. If the proof compiles, the two signatures are equivalent.
5. Otherwise, we analyze what the error is.

Findings

1. Reasoning robustness ⟹ LLMs (still!) can't reason on program semantics.
2. Reasoning effectiveness ⟹ not all reinforcement learning are effective
3. Fine-tuning on math data helps reasoning on code!

My homepage

Paper

Code

Results

RQ1. The performance gap

Model	Version	TTC	Acc	Acc$_{\text{pure}}$	RS
Claude-3.5-sonnet	2024-06-20	✗	85.46	48.97	57.3
Claude-3.7-sonnet	2025-02-19	✓	90.42	**55.85**	61.77
GPT-O3-mini	2025-01-31	✓	**90.43**	48.40	53.52
GPT-O3	2025-04-16	✓	81.91	52.66	**64.29**
DeepSeek-V3	2025-03-25	✗	83.51	43.62	52.23
DeepSeek-R1	2025-01-20	✓	86.70	44.15	50.92
Qwen3	30B-A3B	✓	81.38	40.43	49.68
	32B	✓	87.94	43.09	49.00
	235B-A22B-FP8	✓	85.11	44.15	51.87

Table 1: Main evaluation results. $\text{RS}(m) = \text{Acc}_{\text{pure}}(m)/\text{Acc}(m)$.

RQ2. The effectiveness of TTC

Model	TTC	Acc	Acc$_{\text{pure}}$	RE
Qwen3-235B-FP8	✗	80.49	35.64	1.37
	✓	86.70	44.15	
Claude-3.7-sonnet	✗	87.77	46.81	3.41
	✓	90.42	55.85	
Gemini-2.5-flash	✗	78.19	30.32	3.90
	✓	83.51	51.06	

Table 2: Reasoning effectiveness of top LLMs.

$$\text{RE}(m_{\text{ttc}}, m) = \frac{\text{Acc}_{\text{pure}}(m_{\text{ttc}}) - \text{Acc}_{\text{pure}}(m)}{\text{Acc}(m_{\text{ttc}}) - \text{Acc}(m)} = \frac{\Delta_{\text{pure}}}{\Delta}.$$

RQ3: Fine-tuning on math/code I

FT Corpus	Base Model (FT Model)	Size	Acc	FT Acc	Δ	Acc$_{\text{pure}}$	FT Acc$_{\text{pure}}$	Δ$_{\text{pure}}$
Code	Gemma (CodeGemma)	7B	48.94	53.19	+ 4.25	7.45	12.23	+ 4.78
	DeepSeek-V2 (-Coder)	16B	29.79	55.32	+ 25.53	7.98	15.96	+ 7.98
		236B	38.30	80.85	+ 42.55	11.17	36.70	+ 25.53
	Mistral (Codestral)	22B	61.17	63.30	+ 2.13	19.68	11.17	- 8.51
	Qwen2.5 (-Coder)	1.5B	30.32	36.70	+ 6.38	6.91	9.04	+ 2.13
		7B	65.96	61.17	- 4.79	21.28	21.28	0.00
		32B	74.47	82.45	+ 7.98	36.17	31.91	- 4.26
Math	Mistral (Mathstral)	7B	45.21	47.34	+ 2.13	7.99	15.43	+ 7.44
	Qwen2 (-Math)	7B	40.43	43.09	+ 2.66	3.19	10.64	+ 7.45
		72B	63.83	71.28	+ 7.45	21.81	33.51	+ 11.7

Table 3: Result comparison of fine-tuning. FT Corpus: the corresponding fine-tuning corpus. $\Delta, \Delta_{\text{pure}}$: absolute increase in accuracy after fine-tuning.