s2n-bignum-bench

A practical benchmark for evaluating low-level code reasoning of LLMs

Balaji Rao
John Harrison, Ph.D. (AWS)
Soonho Kong, Ph.D. (AWS)
Juneyoung Lee, Ph.D. (AWS)
Carlo Lipizzi, Ph.D.

05/18/2026

Motivation

- Neural theorem proving has made visible progress on mathematical benchmarks.
- But verified systems proofs often require reasoning (including but not limited to):
  - instruction-set semantics
  - large project-specific proof context
- In these proofs, the theorem prover reasons about actual machine state: registers, flags, memory, program counters, and fixed-width arithmetic.
- **Question:** can current LLMs synthesize proofs in this setting?

What is s2n-bignum?

- s2n-bignum is an AWS open-source library of hand-written ARM and x86 assembly routines for big-integer cryptographic arithmetic.
- Used in elliptic-curve arithmetic, Montgomery multiplication/reduction, modular arithmetic, Curve25519/NIST curves, and SHA-3/Keccak.
- For each routine, two artifacts are shipped:
  - the assembly source (ARM and x86) and
  - a HOL Light proof that the assembly's effect on registers and memory implements a specified mathematical function
- The proof obligation is functional correctness against a bit-precise machine semantics

Related Work

- Mathematical NTP benchmarks
  - miniF2F, PutnamBench, ProofNet
  - emphasize competition-style mathematics and formal proof synthesis

- Verification-oriented benchmarks
  - NTP4VC, VeriSoftBench, miniCTX
  - move toward program verification and proof-context reasoning

- Stepwise and agentic provers
  - MPS-Prover: stepwise theorem proving with search over intermediate proof states
  - Ax-Prover: agentic theorem proving through MCP-connected Lean tools

- **s2n-bignum-bench** targets a different proof regime
  - HOL Light proofs from verified cryptographic assembly
  - Current: static tactic scoring; planned: hol_server + MCP

What is s2n-bignum-bench?

- s2n-bignum-bench is a benchmark derived from s2n-bignum.
- 2,284+ machine-checkable proof obligations.
- The first NTP benchmark targeting industrial low-level assembly programs in HOL Light.

- The benchmark asks a prover to fill in the proof script:

```
prove(goal, <candidate tactic>)
```

- The original proof body is hidden.
- The candidate answer is accepted only if HOL Light checks the proof and no new axioms are introduced.

Verification Project Context

- For each routine, the proof connects:

```
initial machine state
    +
decoded ARM or x86 instructions
    +
formal ISA semantics
    ->
final machine state satisfying the mathematical specification
```

- The proof symbolically executes decoded instructions under a trusted ISA model.
- The trusted base is the HOL Light kernel plus the formal ISA semantics; the rest is derived proof
- The original specifications and proof scripts were written by human experts; neural theorem proving was not used to construct them.

What is an ISA?

- An instruction set architecture, or ISA, is the contract between software and a processor.
- Examples:
  - ARM AArch64
  - x86-64

- A formal ISA model gives mathematical meaning to each instruction:

```
step : MachineState -> MachineState
```

- MachineState contains registers, flags, memory, and the program counter.

What does each Problem contain?

- Each problem is independently runnable, but the context can be large.

- It contains:
  - query.txt: the HOL Light goal term
  - setup.ml: the loaded definitions, lemmas, and proof infrastructure
  - answer.txt: the tactic to be synthesized (by the LLM)

Benchmark Construction

- Starting point: human-authored HOL Light proof files.
- Extraction pipeline:
  - proof file
    - identify top-level prove(goal, proof)
    - keep the goal
    - replace proof body with CHEAT_TAC
    - package goal plus context as a standalone task

- The ground-truth proof body is replaced by CHEAT_TAC; challengers see the goal and context, not the human tactic script.
- Problem ID format: `arch.filename.thm.N` (addresses name collisions across files).

Proof Corpus (as of Feb-27-2026)

Category	Problems
`generic`	562
`program_state`	552
`bit_vector`	311
`functional_correctness_arm`	437
`functional_correctness_x86`	422
Total	**2,284**

ARM And x86

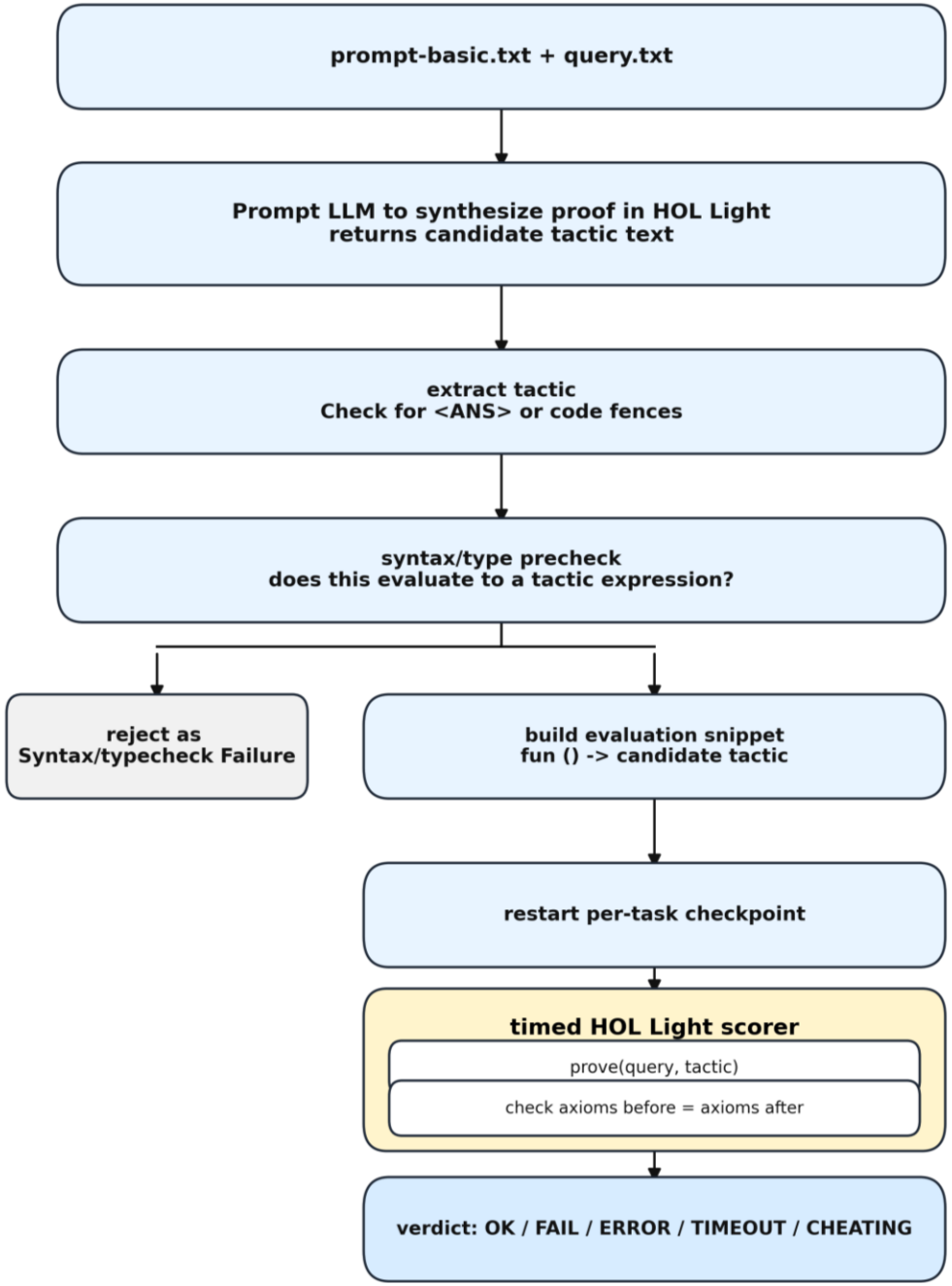
- The benchmark includes both ARM and x86 proofs.

Shared	Per ISA
HOL Light kernel	ISA decode/execute model
proof infrastructure	assembly implementation
bignum algebra lemmas	symbolic execution steps
high-level mathematical specs	per-routine correctness proofs

- Verification effort is not doubled from scratch, but the architecture split is real.

Evaluation Pipeline

- Prompt the LLM with the goal term
- Generate candidate tactic:
  - syntax/type check
  - timed tactic construction
  - HOL Light execution
  - ➢ Verdict



FAIL vs ERROR

- FAIL means the candidate raised HOL Light's standard Failure exception.

```
the tactic ran, but the proof did not go through
```

- ERROR means some other exception escaped.
  - ill-typed runtime behavior
  - unbound or malformed tactic expression
  - unexpected OCaml exception

- Syntax/type failures are often rejected before proof execution, so ERROR is a runtime diagnostic, not the main failure mode.

Baseline Results

- We evaluate GPT-5.3-Codex through codex-cli under zero-shot, query-only prompting regime.
- High-effort GPT-5.3-Codex solves 121/2,284 problems, but 0 full ARM/x86 functional-correctness problems.

	Medium Effort					High Effort				
Category	**Total**	**Eval**	**OK**	**FAIL**	**OK/Total**	**Total**	**Eval**	**OK**	**FAIL**	**OK/Total**
generic	562	330	59	239	10.5%	562	349	66	246	11.7%
bit_vector	311	142	26	109	8.4%	311	140	27	107	8.7%
program_state	552	235	16	211	2.9%	552	229	28	193	5.1%
functional_correctness_arm	437	33	0	33	0.0%	437	42	0	42	0.0%
functional_correctness_x86	422	3	0	3	0.0%	422	6	0	6	0.0%
Total	2,284	743	101	595	4.4%	2,284	766	121	594	5.3%

Integrity And Contamination Defences

- Evolving Benchmark Problems

- Obfuscation
  - Inflate query characters and rename invented type identifiers
  - But preserve the goal's logical structure
  - Succeeds for ~87% of the problem set
  - Type-annotation expansion helps against string matching
  - Does not prevent structural memorization of goal shapes.

Leaderboard

s2n-bignum-bench Leaderboard

A Practical Benchmark for Evaluating Low-Level Code Reasoning of LLMs

GITHUB PAPER SUBMIT

Total Problems: **2,284** ARM FC: **437** x86 FC: **422** Prog State: **552** Bit Vector: **311** Generic: **562** Submissions: **2**

METRIC: Pass@1

CATEGORY: All | ARM FC | x86 FC | Prog State | Bit Vector | Generic

#	MODEL	SUBMITTER	DATE	BADGE	COMPUTE	SOLVED ▼	% SOLVED	ARM FC	X86 FC	PROG STATE
1	Codex-5.3-High-Effort	Balaji Rao	2026-03-14	–		121 / 2284	5.3%	0	0	28
2	Codex-5.3-Medium-Effort	Balaji Rao	2026-03-09	–		101 / 2284	4.4%	0	0	16

s2n-bignum-bench — 2,284 HOL Light proof problems from s2n-bignum

Leaderboard Repo · Benchmark Guide

Leaderboard template adapted from EvalPlus and PutnamBench.

Conclusion

- ***s2n-bignum-bench*** moves NTP evaluation from competition mathematics to industrial low-level verification.
- The benchmark stresses machine-code semantics, bit-precise arithmetic, memory/program state, and large HOL Light context.
- Current frontier LLMs solve only a small fraction of supporting lemmas and no full functional-correctness ARM/x86 proofs in isolation (zero-shot, query term only).

- **Future work**
  - step-level interaction through hol_server
  - MCP binding for agentic theorem-proving systems
  - possible cross-prover goal exports

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Balaji Rao,
brao@stevens.edu

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030

Relevant Links

- Paper: https://arxiv.org/abs/2603.14628
- Benchmark: https://github.com/kings-crown/s2n-bignum-bench
- Leaderboard: https://kings-crown.github.io/s2n-bignum-leaderboard