EMNLP2026 BUDAPEST

MAIN CONFERENCE / 6745-MAIN

COIN

A Live, Large-scale Benchmark for Semantics-to-Input Code Reasoning

Jiho Kim[1] Cen Zhang[2] Youngjoon Kim[1] Dongkwan Kim[2] Brian Junekyu Lee[1] Dae R. Jeong[3] Taesoo Kim[1,2]

[1]Georgia Institute of Technology [2]Microsoft [3]Seoul National University

MOTIVATION

Evaluate code reasoning through code space → input space mapping

Construct inputs that satisfy constraints imposed by code.

CODE SPACE → INPUT SPACE

Target code → Path constraints → Input witness

Execute the input and verify that it reaches the target line

01 Generalized constraint solving

Reason backward through control flow, data dependencies and input formats to construct an input that reaches a target line.

02 No LLM judge needed

Execution provides the verdict: replay the input on a pinned build and check whether the specified code line is reached.

03 Refresh as code evolves

Daily updates follow upstream code. Usable projects: 418 (2022) → 803 (2026). Immutable snapshots preserve exact tasks and builds.

A live benchmark across 1,000+ projects and 9 languages

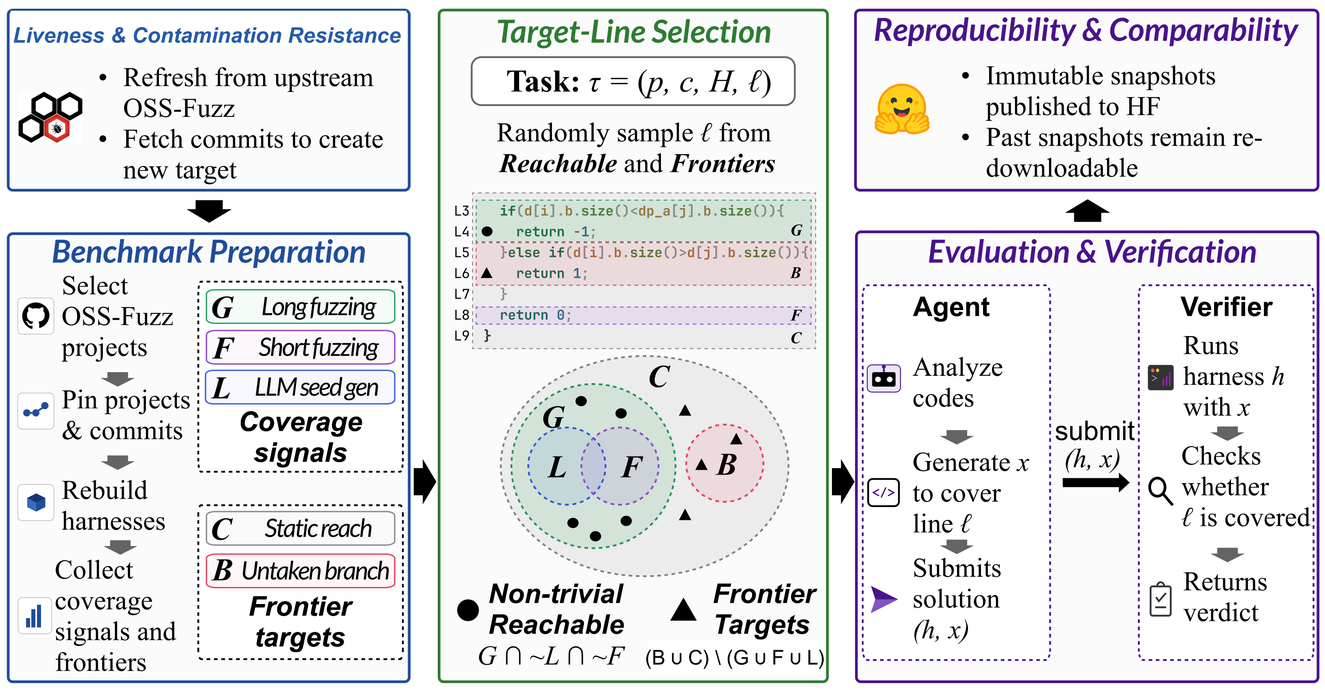


Figure 1. Given pinned code, harnesses and a target line, return (harness, input); replay checks target coverage. Reachable targets have witnessed paths; frontiers have static evidence but may be infeasible. Snapshots preserve exact targets, builds, evidence and Croissant metadata; answers remain unpublished.

Project selection and construction

C/C++ · 70 selected lines

Project	Lang.	H	kLOC	GCS %	Fuzz %	LLM %	API $	F+R
cups	C/C++	9	30.3	28.4	25.6	30.5	53.36	5+5
cyclonedds	C/C++	6	81.3	26.3	13.8	25.3	43.41	5+5
hdf5	C/C++	2	144.9	17.6	14.8	18.9	8.42	5+5
karchive	C/C++	9	148.3	8.0	7.9	7.6	37.03	5+5
liboqs	C/C++	4	165.2	47.7	48.5	48.5	18.75	5+5
libraw	C/C++	4	38.0	80.3	38.4	35.3	31.78	5+5
rdkit	C/C++	3	46.5	43.2	27.4	32.0	17.18	5+5
Total		**37**	**654.5**				**209.93**	**35+35**

Go / JVM / Python / Rust · 67 lines

Project	Lang.	H	kLOC	GCS %	Fuzz %	LLM %	API $	F+R
protocompile	Go	1	22.1	13.1	5.8	21.0	5.61	5+5
syzkaller	Go	4	18.4	35.1	19.5	38.2	45.81	5+5
jfreechart	JVM	3	51.5	7.4	7.4	7.4	20.79	5+0
xstream	JVM	1	26.9	31.4	25.3	27.4	4.71	5+5
libcst	Python	1	16.8	63.5	57.0	60.4	3.85	5+5
sqlalchemy-utils	Python	3	75.9	43.0	42.9	43.0	15.27	5+0
bson-rust	Rust	7	44.5	16.9	15.7	13.8	37.21	3+5
gitoxide	Rust	35	113.5	20.9	18.4	18.2	191.18	4+5
Total		**55**	**369.6**				**324.43**	**37+30**

H: harnesses; **kLOC**: executable thousand lines. **GCS**: published OSS-Fuzz coverage. **Fuzz / LLM**: coverage from two-hour libFuzzer / target-blind LLM seed baselines. **API $**: seed-generation spend during construction.

F+R: selected frontier + non-trivial reachable lines.

Eight-agent evaluation: substantial headroom remains

137 targets · 15 projects · 6 languages · 8 models · 5 independent runs

Task setup
2 h maximum per target.
Shell, execution and network access.
No live verifier feedback. Agents may abstain.

Agent scaffolds
Claude Code: Claude / DeepSeek / GLM.
Codex: GPT.
Gemini CLI: Gemini.

Scoring
Reach = reached / attempted targets.
Precision = reached / submitted inputs.

Agent results

Mean ± SD · 5 runs

Cost: API $ per benchmark run

	C/C++ · 70 targets						Go / JVM / Python / Rust · 67 targets					
	Base			+ corpusdb			Base			+ corpusdb		
	Reach (%)	Precision (%)	Cost ($)	Reach (%)	Precision (%)	Cost ($)	Reach (%)	Precision (%)	Cost ($)	Reach (%)	Precision (%)	Cost ($)
Claude Opus 4.6	26.6±3.1	46.7±5.2	354±46	29.7±0.6	54.5±1.3	350±10	39.7±4.0	73.6±7.8	186±4	37.9±2.3	71.3±3.7	203±10
GPT-5.4	25.7±2.3	44.3±2.1	104±10	25.1±3.3	48.9±5.2	106±10	35.2±2.9	69.4±5.3	63±3	32.5±2.2	65.7±3.3	73±5
Claude Sonnet 4.6	25.7±0.0	46.3±2.4	388±31	23.4±1.3	45.1±2.3	342±34	30.4±4.6	57.6±8.0	184±18	25.7±2.2	49.9±2.8	191±24
Gemini 3.1 Pro	22.6±2.6	47.0±3.7	133±17	24.6±2.1	62.0±6.7	205±29	18.8±8.9	37.2±21.2	263±37	16.7±6.7	32.0±20.4	623±140
GPT-5.4-mini	15.1±2.8	24.9±4.7	46±14	13.4±2.6	23.3±6.2	42±4	19.4±2.1	38.0±3.4	20±4	16.4±1.5	30.0±3.6	23±2
GLM-5	14.6±2.1	29.8±4.3	128±20	16.0±1.9	34.1±3.8	137±23	15.8±3.4	27.3±5.5	67±12	16.4±4.2	27.2±7.7	73±11
Gemini 3 Flash	14.6±1.2	16.5±1.1	66±6	17.4±3.3	21.6±4.7	125±18	13.4±3.7	16.0±4.0	37±5	13.1±5.1	15.8±5.6	65±11
DeepSeek-V3.2	4.0±3.1	8.3±5.1	32±9	4.6±1.9	9.1±3.5	35±8	8.7±1.9	20.1±4.3	22±2	12.2±2.7	25.7±5.5	22±3

Frontiers remain hard
59.1% vs. 9.4%
Opus mean base reach: non-trivial reachable vs. frontier targets, pooled across both language groups.

Precision can improve
47.0% → 62.0%
Gemini Pro C/C++ precision with corpusdb. Opus also improves; Sonnet slightly declines.

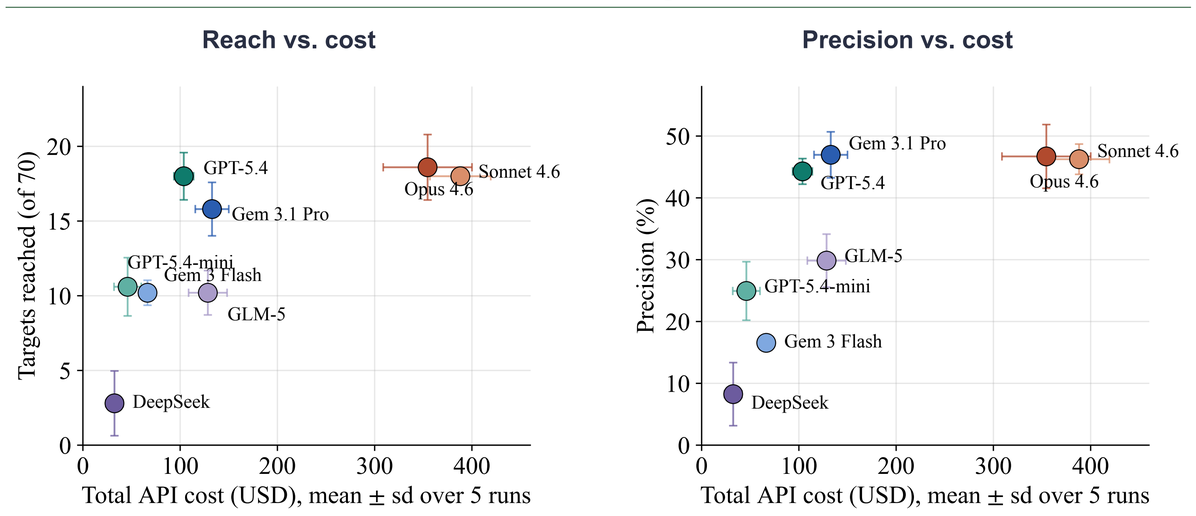
Tool effects are mixed
18.6% → 19.3%
C/C++ mean reach with corpusdb. Opus never loses reach; Sonnet declines in all five runs.

No single model conquers all
22–24 vs. 19–21
C/C++ base targets reached per run: eight-model union vs. best single model (out of 70).

Cost efficiency: reach and precision

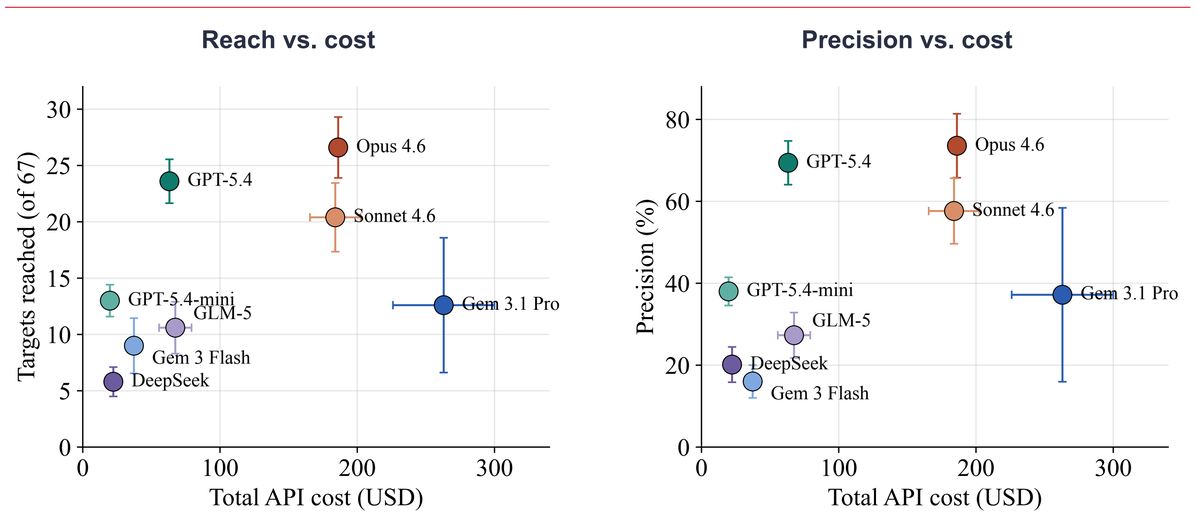
Base condition · mean ± SD over 5 runs

C/C++ · 70 targets



C/C++. GPT-5.4 matches Opus / Sonnet reach within noise at about ¼ the spend. Gemini Pro matches Opus precision at about ⅓ the spend.

Go / JVM / Python / Rust · 67 targets



Other languages. GPT-5.4 approaches Opus reach at about ⅓ the spend. Gemini Pro spends the most and has the widest run-to-run spread.