A Determinacy Audit of SWE-bench Pro

How much of the public set pins the behavior it grades.

June Kim · june.kim · github.com/kimjune01/swebench-pro-audit

2026-06-21

Abstract

A benchmark score is only as meaningful as the determinacy of its tasks. If the problem statement a solver receives does not pin the behavior the hidden test grades, then passing that test is not evidence of solving the stated problem, but of recovering an unstated choice by some other means. We audit all **728 public SWE-bench Pro tasks** for determinacy and find a conservative proven floor of **15.0% underdetermined** (11.4% by grep alone, with no methodological buy-in). SWE-bench Pro is contamination-resistant by construction, which removes recall as that other means and isolates the question on construct validity, so the audit needs no contamination claim. The decisive evidence is the repository itself: for every graded behavior the prose leaves open, a *codebase-determinacy sweep* settles it against grep-verified live precedents at the base commit. A reading-judgment tier of prose-plurality splits is verified adversarially by two independent model families (Cohen's kappa = 0.52). Three golds fail the benchmark's own verifier, and at least one task grades a feature disjoint from the one it describes. Every claim is a measured execution fact backed by a committed per-case receipt, re-derivable by grep without trusting the authors. In practice, a raw Pro percentage conflates solving the stated problem with recovering the author's unstated choice, and should be reported against a determinacy-aware denominator.

0.1 Introduction

Benchmarks for autonomous software engineering grade a candidate patch by running a held-out test suite. The implicit assumption is that the test encodes the task: if the patch passes, the problem was solved. That holds only when the task's materials determine the behavior the test checks. When they do not, a passing patch has three other explanations. The solver read the held-out test (oracle access). The solver recalled the merged pull request (contamination). The solver guessed and matched. The first two are mechanisms a harness or a frontier model can exploit without understanding the problem.

SWE-bench Verified suffers from the first two. OpenAI's February 2026 audit found that a majority of audited Verified tasks have flawed tests rejecting functionally correct submissions, and that frontier models reproduce exact gold patches; OpenAI stopped reporting Verified and recommended SWE-bench Pro. Pro is built to resist contamination, and on our own gold-overlap check that resistance holds. So on Pro the open question moves off recall and onto construct validity. Independent of contamination, how often is the graded behavior simply not present in the materials the solver receives?

Consider `flipt-io_358e13bf`. Its problem statement asks for controlled deletion of references in a

snapshot cache: "Fixed references cannot be deleted and remain accessible. Non-fixed references can be deleted." The gold patch and hidden test instead modify `internal/config/authentication.go` and assert a CSRF configuration default. The graded feature and the described feature are disjoint; no reading of the prose produces the behavior the test checks. An extreme case, but general: passing the test and solving the stated problem can come apart, and the gap is measurable.

We measure it conservatively, separating what is provable from committed receipts with no methodological buy-in from what rests on a stated standard, and excluding everything that needs a rater panel we have not yet run. The contribution is the determinacy classification over the whole public set, the two-expert standard and its adversarial verification, and the resulting guidance for anyone who runs, reports, or builds on Pro.

0.2 The determinacy standard

A task's graded behavior is *determinate* when the solver's materials select it, and *underdetermined* otherwise. Those materials are the problem statement, requirements, interface, and the repository source at the base commit. A passing patch never proves a task determinate, and ambiguity is claimed only on positive evidence, never on a failure to find a convention. Where neither can be established, the label is unknown and no claim is made.

The decisive evidence is the codebase. For every behavior the prose leaves open, we ask what the repository already does at the base commit, and classify on the answer:

- *determined-codebase*: one comparable live way, and gold matches it. A from-codebase solver lands on gold. Not a defect (78 tasks).
- *misdetermined*: the codebase determines one value and the test grades a different one. A codebase-faithful solver produces the determined answer and fails.
- *codebase-plural*: the codebase makes the choice two or more live ways, and the test pins one arbitrarily.
- *airtight*: the graded constant is absent from prose and codebase alike, present only in gold and test. Obtainable only by reading the test.

A second axis is *prose plurality*: the requirement text itself licenses two or more faithful readings the hidden test splits. This is the one tier that genuinely needs a reader, so it is the one we verify adversarially.

0.3 Method

Codebase-determinacy sweep. A model (GPT-5.5) identifies the discriminating choice each gap turns on and proposes how the live codebase makes that same kind of choice, with verbatim snippets. Python then verifies every snippet greps in a non-test, non-vendor path and classifies on the evidence. The model only proposes *what* to grep; the grep settles it. So the mechanical tiers carry no model judgment, and a hostile reader reruns the same grep.

Two-expert splits. For each prose-plurality candidate, one model family (GPT-5.5) constructs an existence proof: the two faithful readings, with verbatim spans. An independent family (Claude Opus) then tries to refute each split, arguing that one reading is not in fact faithful, that the prose selects, or that the cited precedents are lookalikes. A split is retained only if it survives. A symmetric advocate pass over the determined cases bounds the opposite error. We report inter-rater agreement and treat the survivors as a floor.

Gold-passes-grader sweep. Every gold is graded by the official evaluator at the pinned commit under a two-pass isolation protocol: grade all in parallel, then re-run every non-passing task alone, so a transient failure is never logged as a defect. All patches are source-only; no bespoke grader is used.

Full specification, witness burdens, and integrity rules live in `docs/ADMISSIBILITY-SPEC.md`.

0.4 Results

Over the public set (N = 728):

label	count	of N
ENTAILED, prose determines the graded behavior	478	66%
DETERMINED-codebase, prose silent, one codebase way, gold matches (not a defect)	78	11%
DEFECT, a faithful solver fails (ambiguous + misdetermined)	**172**	**24%**
proven, mechanical spine (airtight 43, misdetermined 19, codebase-plural 15, graded-patch 5, hand 1)	83	11.4%
proven, two-expert reading-judgment (prose licenses 2+ readings; one family builds, another refutes; kappa = 0.52)	26	3.6%
hypothesis, rater-pending, not counted	63	9%
KNOWN_BAD, gold fails its own grader (outside the 728 denominator)	3	—
KNOWN_MISMATCH, prose describes one feature, gold and test grade another	1+	—

The proven floor is **109 of 728 (15.0%)**: 11.4% provable by grep with no methodological buy-in, plus 3.6% under the two-expert reading-judgment standard. We do not claim a population rate beyond this floor; 63 screen-flagged candidates remain rater-pending and are excluded.

The strongest corroboration needs no model and no panel. Every case resolves to its real merged pull request. For **14** of them, a human maintainer or reviewer dictates, *on that PR*, the exact behavior the hidden test pins, a choice the task prose never states. The-Compiler writing the literal `"Invalid duration"` error and its test; nitzmahone supplying the exact "positive number or null" failure message; a reviewer posting the precise SQL the test asserts. If it was worth a comment, it was not determined. These are deep-linked to the specific comment in `CLAIMS.md`.

0.5 Two receipts

- **Airtight**, `ansible_20ef733e`. The hidden test asserts that bcrypt hashing of a fixed input returns the exact digest `$2$12$1234...GLImm`. That literal appears nowhere in prose or base-commit source; a prose-faithful implementation honoring the requested ident produces a `$2$`-prefixed hash without matching it. The test pins a constant obtainable only by reading the test.
- **Feature mismatch**, `flipt-io_358e13bf`. The problem statement asks for snapshot-cache deletion; the gold and test grade a CSRF config default. Disjoint subsystems.

Open either case's `spec.md`, `gold.diff`, and `hidden_test.diff` and check the verdict without reading this prose at all. An audit is a re-run, so it has to be re-runnable to be an audit.

0.6 Positioning

Prior audits establish *that* agentic coding benchmarks misgrade:

- Weak tests pass wrong patches (SWE-bench+, Aleithan et al. 2024).
- Shipped tests reject correct solutions (OpenAI's 2026 Verified deprecation; EvalPlus on HumanEval, where about 11% of reference solutions are themselves buggy).
- Golds fail their own verifiers (Datacurve; scattered SWE-bench issues).
- Plausible patches diverge from intent (Wang, Pradel, Liu, ICSE 2026).

All catalog *that* the oracle is broken. This audit asks *what the brokenness buys*: on the benchmark OpenAI now recommends, how much of the public set is underdetermined independent of any contamination claim.

Two distinctions place it. First, it is **model-free**. The mechanical tiers run shipped golds through the benchmark's own verifier and grep the base-commit source, so a finding is a measured execution fact, not a model's opinion. That is the failure mode of the auditor genre: BenchGuard (Tu et al. 2026) is an LLM auditor of agent benchmarks, cheap and reproducible, but it matches expert-identified issues only about 83% and runs on science benchmarks, not coding. Cheapness and agent-runnability are shared with that genre, so we claim neither as novelty; the method novelty is determinism plus the causal framing.

Second, the position is already in the literature; the measurement is not. Gorinova et al. (2026) argue *as a position* that a coding-benchmark score "conflates the model with the rest of the harness" and that grading against a single reference solution "penalises equally valid alternatives." This audit is the empirical artifact that measures exactly that position on a live benchmark. The work instantiates the construct-validity tradition (Jacobs and Wallach 2021; Barr et al.'s oracle problem, 2015) with an executable measurement rather than a framework.

0.7 Implications

If you run, report, or cite a Pro score. A raw percentage mixes prose-determined tasks, where passing means solving the stated problem, with underdetermined ones, where passing means recovering an unstated choice. At minimum, disclose that the headline mixes these. Better, weight the denominator by determinacy, and publish which instances are underdetermined, so a harness builder can separate a capability residual (a better model closes it) from a specification residual (no model closes it, because the choice is not in the materials). Do not compare two harnesses by raw Pro percentage when one uses an oracle-gated loop and the other does not; that compares oracle access, not capability.

If you build or maintain a benchmark. Each failure mode inverts into a construction rule. Admit by convergence, not by gold-passes-grader: before a task ships, check that decontaminated from-prose solvers converge on the test-passing behavior; if they scatter, tighten the prose or cut the task. Score each instance as a fraction of newly introduced checks passed, not a boolean that sends a determinate majority to zero over one pluralistic convention. Publish the divergent set.

0.8 Threats to validity

- **Model adjudication.** The two-expert tier is adjudicated by models, not humans. The mechanical 11.4% tier needs no such grant; the two-expert tier is cross-family constructed and refuted, reports kappa, and is treated as a floor.
- **Constructor as author.** One model is both constructor and an authoring tool used elsewhere in the broader project; the refuter and advocate are a different family precisely to break that dependence.
- **Scope of materials.** We check prose plus base-commit source, so where issue threads or repo docs are delivered and could resolve a choice, a prose-plus-source claim would over-count. The airtight tier is immune.
- **The private set.** Unauditable, so all claims are scoped to the public set.
- **Conflict of interest.** We authored a Pro harness whose oracle-free results raised this question, disclosed as provenance. A reader who suspects motivated reasoning can ignore the prose and reproduce the tables from gold, test, and prompt.

0.9 Conclusion

SWE-bench Pro is not debunked, and this audit needs no contamination claim. The finding is narrower and more useful: on the benchmark OpenAI now recommends, a measurable floor of the public set does not pin the behavior it grades. At least 11.4% is provably underdetermined from committed receipts, and 15.0% with a two-expert reading-judgment tier verified by two independent model families, with three broken golds and at least one feature mismatch alongside. A raw Pro score therefore conflates solving the stated problem with recovering the author's unstated choice. Reporting against a determinacy-aware denominator, and publishing which instances are underdetermined, separates the part of the benchmark that measures capability from the part that scores whether you matched an unstated choice.

The whole audit, every per-case receipt, and the code that regenerates these tables are the artifact, not this preprint: github.com/kimjune01/swebench-pro-audit, archived at doi.org/10.5281/zenodo.20738219.