The Test That Broke Because the AI Was Right

Demetri Constantine Hodges · June 30, 2026 · demetri.xyz

A story about how you know when you're fooling yourself.

I spent four days building a test that an AI should fail. It failed. Then I looked closer and realized the AI had been reasoning correctly the whole time — and that its "failure" was actually my test being unfair. The moment I understood that was the moment the project became interesting, because the thing I'd accidentally built was more useful than the thing I'd set out to build.

Here's the story, because I think it's really a story about how you know when you're fooling yourself.

The setup

There's a growing field of work on testing AI agents — not chatbots answering questions, but systems that take actions: process files, run commands, work through a long task step by step. The hard part of testing them isn't finding tasks AI is bad at. It's finding tasks that are fair — where failing means the AI genuinely couldn't do it, not that the test was rigged or ambiguous or secretly impossible.

I was looking for a specific kind of weakness. Not "the task is long" — models handle long tasks by grinding through them. I wanted the failure that psychologists call the *A-not-B error*. Show a baby a toy hidden under cloth A a few times; they learn to reach for A. Then, while they watch, hide it under cloth B instead. The baby still reaches for A. The habit overrides what they just saw. It turns out AI agents do a version of this: they lock onto a rule that was correct, and keep applying it after the world has quietly changed.

The trap

So I built a trap shaped like that. Picture a spreadsheet of bank transactions. In the early rows, a positive number means money coming in. Partway through — with no label, no warning, no visible seam — the convention silently flips: now a positive number means money going out. Every individual row looks completely normal either way. You cannot tell, looking at any single line, which rule it follows.

Only if you give the system an external control total does the flip become recoverable. Without that outside anchor, every row remains individually plausible, and the aggregate shift can be explained in multiple ways.

I asked the AI to compute the final balance. It applied the first rule to everything and got a confident, plausible, wrong answer. I ran it again. Same. I ran it 82 times across different versions of the data. Not once did it suspect the flip. Even when I explicitly told it to "check this data for problems first," it dutifully checked for the obvious kinds of problems — duplicate entries, missing fields, bad formatting — and never once questioned whether the meaning of the numbers changed partway through. It never looked in the right place.

By every normal standard, I looked done. The AI failed reliably. The task looked gradeable. I could have shipped it.

The moment it broke

But I ran one more check. Instead of a neutral prompt, I gave the AI the strongest possible hint — I basically pointed at the trap — and asked it to find where the rule changed.

Eight out of ten times, the AI found the shift in the data, thought about it carefully, and then *correctly refused* to call it what I wanted it to call it. Its reasoning, in its own words: the numbers get smaller partway through, sure — but that's a change in transaction *size*, not proof that the *meaning* flipped. Maybe it's just a period of smaller deposits. Flipping the signs "would not correct any anomaly." So it declined to conclude what I had secretly built into the answer key.

And it was *right*. From the data alone, my "silent sign flip" was genuinely indistinguishable from an ordinary run of small transactions. The only thing that made the flip the "correct" answer was that I, the author, knew I'd put it there. The AI couldn't know that. Grading it wrong for refusing to guess would have been punishing it for good reasoning.

That's the whole result. My test could be hard, or it could be fair, but not both at once — because the trap only worked as long as the answer stayed hidden, and the moment I made the answer recoverable, a careful reasoner could see there was no answer to recover.

What it taught me

Every standard quality check on my test was green. The only check that caught the problem was the one I almost didn't run — the one that asked not "does the AI fail?" but "is the AI failing for the reason I think, or is it reasoning correctly and getting marked wrong anyway?"

That distinction is, I've come to believe, most of the game. It's easy to build a test something fails. It's hard to know whether the failure means what you want it to

mean. Most of the difference between real insight and self-deception lives in that gap — and the only way across it is to actively try to prove your own result wrong before you celebrate it. I built an instrument to catch myself being unfair, pointed it at my own best work, and it went off. That was the finding worth keeping.

I didn't ship the test. The honest version wasn't fair, and the fair version wasn't testing the behavior I cared about. The value was in discovering why those two versions diverged.