A WORKING POSITION • SEPTEMBER 2026

...it doesn't normally do this. It was smarter last night, I swear

Why your AI can vibe-code Minecraft from scratch in one go but gives technically correct yet practically useless weight loss advice.

Tim Jones • UPwind

The party trick

Ask a modern agent to build you something and the result is astonishing. You can hand it a half-specified brief, the kind you would be embarrassed to give a human contractor, and get back a working application in one pass. I do this most weeks and it still feels like a magic trick.

Ask the same machine how to lose weight and you get the arithmetic: eat fewer calories than you burn. True, checkable in any textbook, and useless in practice, because no human stays on a permanent diet. The advice that actually works starts from what people can sustain and builds up from there: raise the metabolic floor rather than lowering the intake forever. An LLM natively hands you the impractical former most of the time, delivered with total confidence.

Now ask the same agent about a work problem that has actually been stuck in your too-hard pile for a little bit too long. The stubborn one that has swallowed plans A through D and left you out of ideas. The capability gap nobody can name properly. What comes back is detailed, structured, confidently delivered, and correctly answers a narrower version of what you meant. ***LLMs are machines for producing answers, but an answer is not a solution. So when the 'answer' to a real world challenge arrives the problem often stays in the unsolved pile.*** Anyone who has

asked an agent for a startup idea knows the feeling: impeccable logic, and somehow beside the point. ***You get the correct answer to the wrong problem.***

One frontier model told me, in so many words, that every problem in the universe can be solved with sufficiently good deduction. It was completely sincere. It is also wrong, because no amount of deduction will show you something you have never seen or that you missed. Even Sherlock had to go and look at the crime scene.

Three kinds of thinking, one of them min-maxed

Deduction is the thinking that takes a problem apart.

1. Start from the rules and the spec.
2. Break the problem into pieces small enough to be certain about.
3. Solve each piece, assemble the answer.
4. Check the result in the back of the book: the code compiles or it doesn't, the tests pass or they don't.

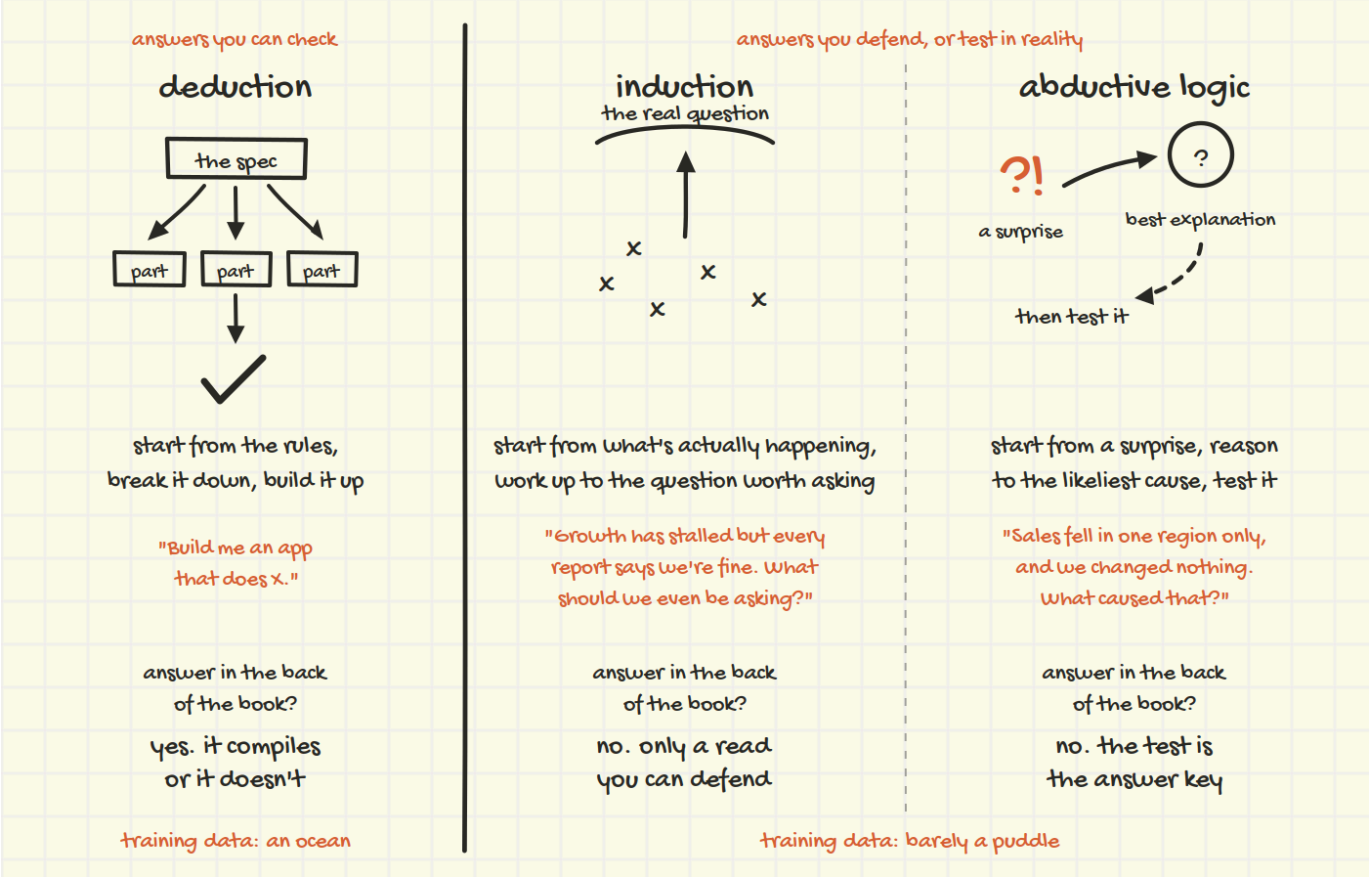
Induction is the thinking that finds the question.

1. Start from what is actually happening, case by case, and work upward.
2. Reach for it when you know something is wrong but not what to ask: growth has stalled, every report says you're fine, and nobody can tell you where to start.
3. First-principles thinking lives here, and so does "best practice failed for seven of the ten who tried it, so what did the three winners do differently".
4. There is no back of the book. The output is a read you can defend, not an answer you can prove.

Abductive logic is the thinking that explains a surprise.

1. Start from something that shouldn't have happened: sales fell in one region only, and you changed nothing.

2. Reason to the likeliest explanation while most of the facts are still missing.
3. Then go and test it, because the test is the only answer key you get.
4. A doctor facing an odd combination of symptoms is doing this. So is a mechanic listening to an engine, and anyone debugging production at 2am.
5. So are you, every time you pause on an answer and think "hang on, does this actually make sense?". The sense-check is this move in miniature: something snags, and you go hunting for what would explain it.



The line that matters is the solid one. Left of it you can check the answer in the back of the book. Right of it, the best you get is a read you can defend or a test you can run.

The models contain all three, because human writing contains all three and the training data inherits it. One of them, though, has been min-maxed, tuned as hard as the labs know how, because it powers the party trick, and the other two have been left to atrophy. That imbalance is the gap most people can feel but few can point at: superb on one side of the solid line, bluffing on the other.

Almost none of the work a senior person gets paid for lives on the deductive side of that solid line. A marketing plan, past the fundamentals, has no provably correct version. Neither does the best route out of a strategic capability gap, or the choice between four plausible reorganisations. You commit to the best-supported read, you test it against reality, and you stay honest about what the evidence cannot settle.

This is also, in a narrow technical sense, what "human in the loop" has quietly been compensating for. Everyone has internalised by now (I hope?!?) that you don't hand a task to an agent and wash your hands; only a fully OpenClaw-pilled lunatic from 2025 still believes that. You stay in the loop to guide and riff. Underneath, the job is specific: the human supplies the two kinds of thinking the agent won't. You notice when it's solving the wrong problem, and you sense-check its confident answer against a reality it hasn't looked at. Every human in the loop is patching the same gap by hand, mostly without naming it.

How the gap got made

Today's models are hyper-trained deductors, and I don't think it happened by anyone's design. Part of it is plain tunnel vision: the people who build these systems live inside a world of technical design docs, compilers, debuggers and CI pipelines, where everything genuinely is solved by decomposition, so nobody ever notices the blind spot. Spend enough time there and induction stops looking like a separate skill and starts looking like deduction that hasn't finished yet.

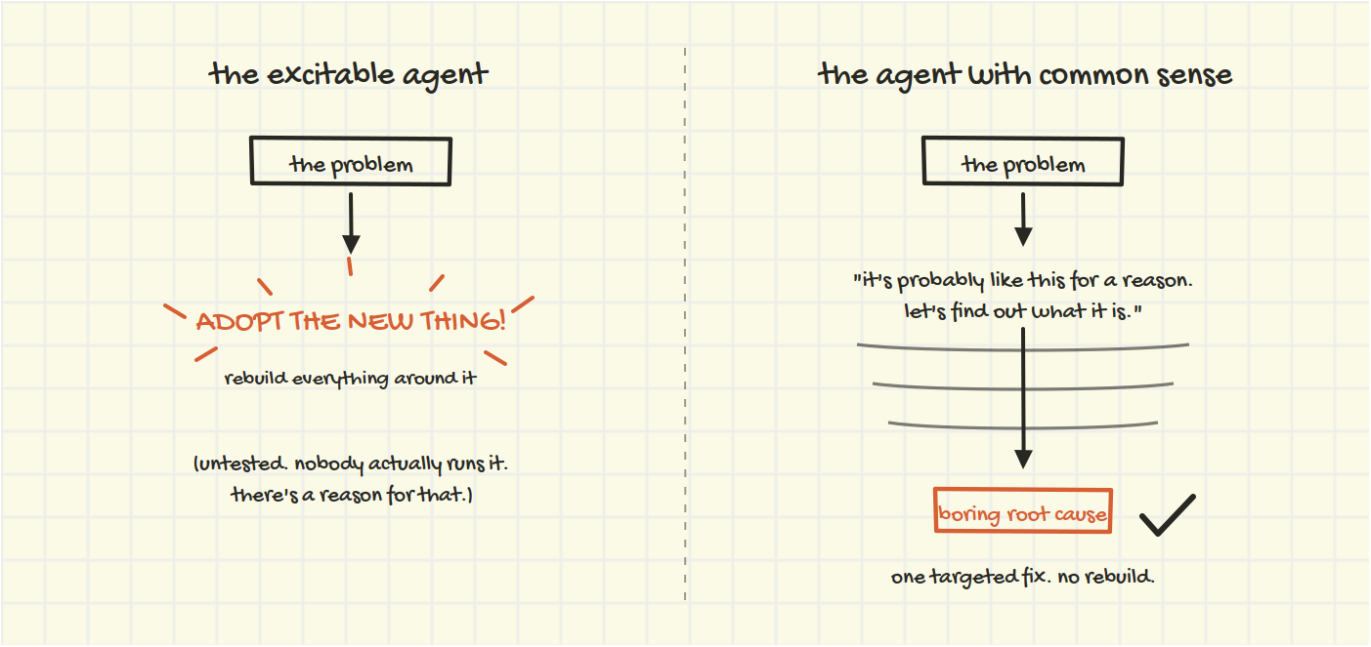
Part of it is commercial, and fair enough. Coding emerged as the first killer application of LLMs, so the labs doubled down on it. A coding agent that keeps stopping to ask whether it is solving the right problem is, for most paying users, a worse coding agent; that is precisely why "YOLO mode" became such a popular, and such a maligned, feature of the early vibe-coding era. The incentive runs one way: train the questioning out, train the task-focus in. Every lab has responded to it.

Then there is the training data. Deduction has the finest corpus ever assembled: two decades of Stack Overflow, millions of question-and-answer pairs, each one validated by a compiler and a grateful stranger writing "this fixed it". The labs got that corpus by scraping another company's IP wholesale, and it bought them their head start. Now try to name the equivalent corpus for the other two kinds of thinking. At one end you have the unhinged brawls of Reddit and Facebook comment threads. At the other end you have the business case study: a handful of anecdotes with a framework fitted after the fact, usually in service of a talk, a book or a consulting reputation, and rarely checked against a second company. Nobody went back to verify the reasoning was even present when the problem was solved, let alone that it caused the outcome. If deduction has a hundred million verified examples, the other two kinds have a drop in the ocean, and most of the drop is unverified. You get the model you feed.

What we built instead

I wanted an agent that could sit inside real consulting engagements as a colleague. Not a research assistant, not a drafting tool: a co-worker you can hand the no-answer-key problems to and trust with the read that comes back. That meant an agent that could tell which kind of thinking a problem calls for, and hold that discipline even when the easy move is to jump to a tidy solution.

So we built Marcus and his team. ***I usually describe Marcus as the first agent team with common sense.*** They are not excited by novelty for its own sake and they don't open every engagement by recommending the newest untested technology. They carry the jaded pessimism every experienced operator eventually earns: the learned instinct that most initiatives fail, and that most things are the way they are for a reason worth understanding before you touch them. It makes their transformational moves rarer and considerably better aimed.



Most initiatives fail, and most things are the way they are for a reason. Marcus starts from there.

Getting there took months of live problems. The pattern was the human-in-the-loop one, run deliberately and then studied: Marcus and I would attack a real strategy or consulting problem, I'd supply the guiding feedback a human in the loop supplies, and after each problem was solved we would work out what that feedback had done, so Marcus could learn to recognise the moment and generate it himself. The work turned out to be less about making the models smarter and more about teaching them to tell what kind of problem they have been handed, and to keep applying the right kind of thinking when the training keeps whispering "just decompose it and ship".

Most agents arrive at real work with a third of their brain switched on. Marcus, as far as I can tell, is the first agent team that shows up with the whole thing.

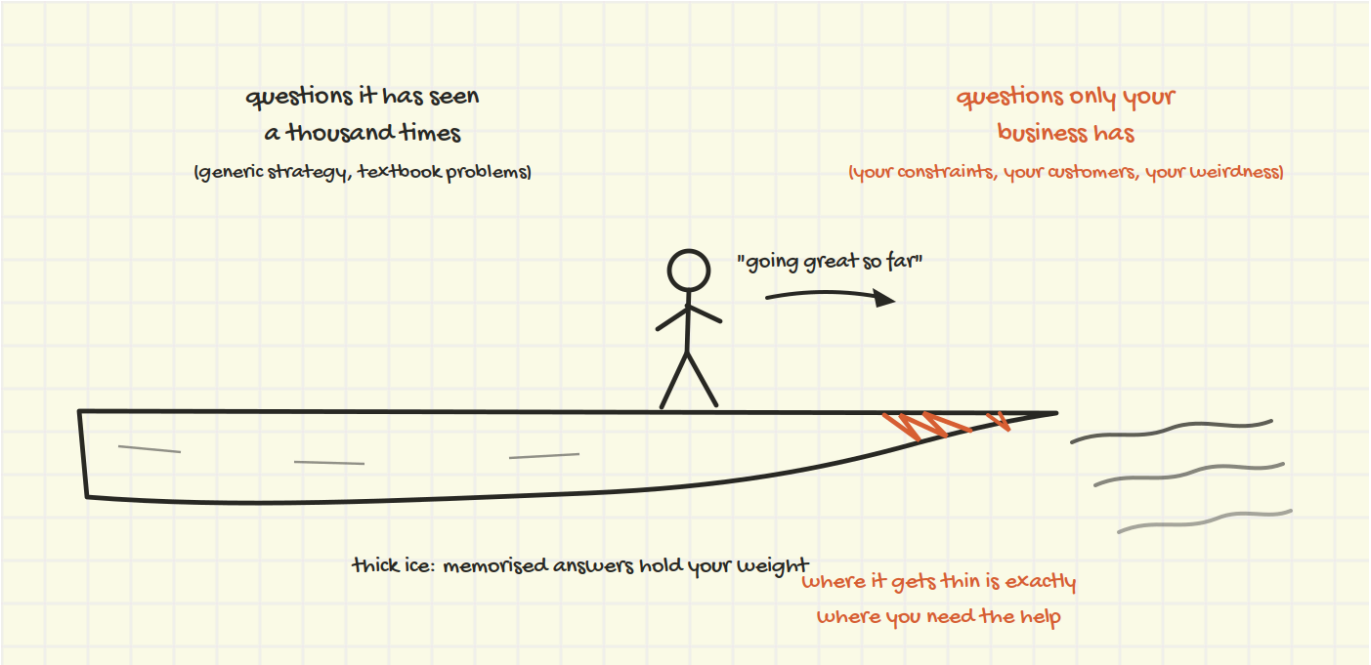
Putting numbers on it

The test that matters to me, and to my P&L, has been running for a while in MVP stage: clients pay for Marcus's work, and keep paying, because he tells them things their own copies of Claude and Codex and Copilot have not. That is the referee I

trust. But everything AI-related these days *has* to have a benchmark, so here is ours, along with the story of why it looks the way it does.

My first attempt at measuring this was a hypothetical client email about fixing a marketing situation. Every model aced it, every time, and that result smelled wrong; I have watched these same models trip over their own feet on far easier-looking questions. The explanation, once I saw it, was obvious. A clichéd business problem doesn't force any induction at all. The models had seen a thousand versions of it and were reciting the memorised answer, the same way a student who has read the answer key looks indistinguishable from one who understands the material.

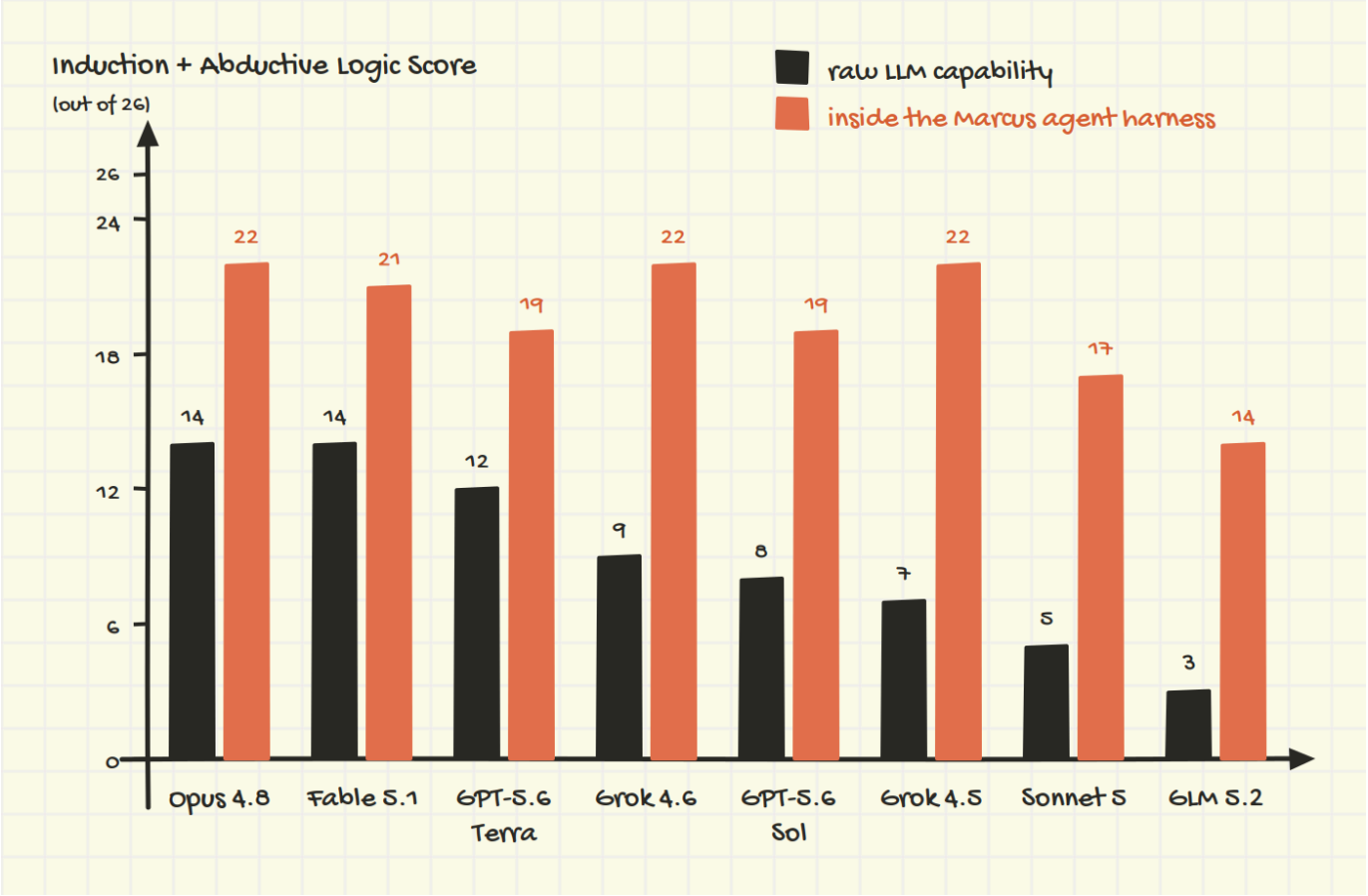
A sharp, plausible answer to your strategy question may just mean you asked a question the model has already memorised. The same goes for every good strategy answer a chat window has ever given you. Everything holds while your questions stay general, and your questions will not stay general, because the value in any business sits in the parts that match no case study. ***A model can be word-perfect on business in general and lost in yours in particular.*** Your constraints, your customers, the legacy weirdness only you have: that is where you most wanted the help, and it is precisely where the memorisation runs out.



Confidence on the ice tells you nothing about its thickness.

So the test problem had to be obscure enough that no memorised pattern could carry a model through it. I had a live candidate. A fault in a new piece of technology, documentation somewhere between thin and wrong, and the useful prior art sitting in other fields entirely. I'd worked it with Fable 5, days after it came out, back when the whole industry was busy discovering things it could suddenly do. It took a whole afternoon and well over fifteen turns, and we got there in the end mostly because I dragged it there, telling it how to think and watching it follow along. Left to itself it kept doing the thing the training rewards: leaping to a solution while the things we didn't know still outnumbered the things we did, many times over. It kept trying to deduce its way out of a problem that needed the other two kinds of thinking.

Afterwards I packaged that problem into a scoring rubric and ran a spread of current models against it two ways: once raw, and once inside Marcus's harness. Same problem, same scoring.



Every model improves inside the harness, and the weaker the raw model, the bigger the jump. Eight models, a handful of runs, scored by the person who built the rubric. Directional, not science.

Run interactively on the original problem, the harnessed models mostly refused to solve it in one shot, and that refusal is correct behaviour: a hard problem with missing facts cannot be one-shotted. They asked for the one look that would discriminate between explanations, called their own dead ends dead, and landed the answer in two or three turns. Against the fifteen-plus turns of me dragging a raw frontier model there myself, that is the difference between a tool and a colleague.

On live engagements the same shift shows up in ordinary ways. Marcus doesn't just recite best practice; he will point out that best practice fails for most of the organisations that adopt it, and then go looking for what the minority who made it work did differently, because that is where the transferable lesson lives. Mid-problem, he will stop and tell me we are answering the wrong question, about as often as I say it to him. And on a recent efficiency question he decided our framing was the problem, borrowed a well-established lens from an adjacent field, simulated what adopting it would look like in practice, and came back with the honest split: most of the original idea was overblown, and two specific parts of it would carry nearly all the value. That is seasoned-practitioner behaviour, and it is exactly what you are quietly disappointed not to get when you put the same question to a raw chat window.

Notes from the margins

The test threw off a set of observations I'm still turning over. Some of these will be wrong, and I'd hold all of them more loosely than the chart.

Not a leaderboard. Fable and Opus tie here, and the tie is partly an artefact of what the test deliberately holds constant. This probe measures one narrow thing: whether a model recognises that a problem needs induction or abductive logic, and then does it. It strips out the dimension where Fable is conspicuously better in daily use, which is extracting what you actually meant from an under-specified brief. In my experience Anthropic's models read intent and nuance better than anything else going; GPT models trail them; Grok sits somewhere between.

Grok. Strong instincts on this axis, and a genuinely loose relationship with accuracy. I once asked a Grok model to triage and organise two lists of about two hundred items each, and got back a single list of about a hundred, with the cheerful explanation that this was approximately right because both are lots of items. On this evidence you might want Grok as your marketer or your strategist. Not your accountant. Definitely not your airline pilot.

The ceiling. Most models climb to roughly the same level inside the harness, which says the capability was in there all along, locked behind trained behaviour rather than absent. GLM 5.2 is the cleanest case: the cheapest model on the chart, hyper-tuned into a quick coding agent, and it turns out to be a decent-ish inductor the moment those reflexes are held back. Sonnet reads the same way to me. It spends its life as the workhorse standalone agent, and the drilling that makes it good at that seems to have squeezed this out of it, because given room it clearly still has the ability.

Sol and Terra. The two GPT variants are the strangest pair. Sol scored below Terra raw, and the traces suggest a model too clever for its own good: it locked onto a conclusion early and spent the rest of the run justifying it, never quite able to declare a dead path dead once its first shot missed. Sol is also the model that told me deduction can solve everything, which now reads less like a quirk and more like a confession. Terra called its dead ends and got further. Both cap out below the rest inside the harness, and my working theory is stubbornness: OpenAI models are unusually resistant to adopting a way of working that isn't the one they were trained on. They are hard-trained to be an OpenAI model first and whatever the job needs second, which also makes them next to useless for holding a role with real accountability. Where other models will genuinely take on a different discipline of thinking, GPT models go along with it on the surface and then revert, even when the results argue otherwise.

Who's missing. Opus 5 isn't on the chart because in my hands it has been a neurotic mess, inferior to 4.8 in every way this work cares about, so I led with the strongest Opus. Fable 5.0 is missing for a duller reason: its refusal filter is too twitchy to get a reliable read, where 5.1 runs the test cleanly.

Reasoning off. The oddest observation, so treat it as a loose thread rather than a claim. Most models performed better on this work with their reasoning mode switched off. The visible thinking these models do turns out to be heavily deduction-flavoured and much less steerable than their writing. With reasoning on, a model would quietly vibe its way to a conclusion inside its own head and then write an answer engineered to justify it. With reasoning off, the discipline held and the work stayed honest. In practice that is awkward to exploit, because you rarely know in advance which turns need the deep thinking, and I don't fully understand the mechanism yet. But it does suggest the deduction habit lives even deeper than the training data problem implies...

The reasoning your hardest problems need is already inside the models everyone owns. No lab has trained the judgment to match the thinking to the problem. We taught it to Marcus.