Your LLM Doesn't Write Correct Code. It Writes Plausible Code.

Yuxi Liu
Jan 15, 2027

One of the smartest tricks you can use as a developer

Pasting a bug report into your LLM and hoping for the best.

Trusty Bug Squashing Bunnyrock
Bunnyrock, Bunny Valley, Carrotland
Bug Squashing Bunny Wanted!

Type your bunny name here...

Will it one-shot the fix, or will it hallucinate a new bug while it tries?

It's a non-deterministic system. The model is a probabilistic function, and its output is a probabilistic guess.

RNN, forced random hacks combined the primitive with hyperbolic. This can be a powerful tool for exploration. But when you're trying to ship a stable, secure, and maintainable feature, the unpredictability becomes a liability. Prompt engineering helps, but it's a probabilistic fix for a probabilistic problem. Retrieval-Augmented Generation (RAG) is a fantastic tool for grounding a model in specific, factual data. But it can't fix the core problem of a model that's fundamentally unreliable.

The thing is bigger. The entire industry is stuck on a local maxima. We're optimizing for plausibility when what we need is correctness. We're optimizing for compatibility, and it drags LLM. Not the planet's code but the industry's collective intelligence.

Correctness is a property of the system, not of the model.

Little spinner for plausibility, but a crucial, robust, deterministic layer for correctness.

I think this is a paradoxical, yet accurate, vibe-coding. Over 30 years of professional software engineering, I've learned that the most boring, predictable, and well-understood tools are the most powerful. Relational databases, compilers, and version control systems are the boring bedrock of our industry. The reason we can build complex, high-stakes applications on top of them is because we know exactly what they do. We can reason about their behavior, we can trust their output, and we can build upon their guarantees. I want to be clear that I'm not a luddite. I'm a pragmatist. I see the immense potential of these tools. But I think we are rushing to integrate them into our workflows before we have the foundations to do so safely.

LLMs Lie. Numbers Don't.

Instead of relying on probabilistic guesses, we can use formal methods. Instead of asking an LLM to write a piece of code, we can ask it to write a formal specification or a mathematical proof that proves the code is correct. This is a deterministic, verifiable, and auditable artifact. The code is just a byproduct, a commodity that can be regenerated and verified on demand.

Approach	Time	Cost
MINIPT 700 line SaC	1.36 ms	2,242.94 ms
SLLGT 80 line SaC	0.34 ms	9.69 ms
SLLGT 80 line SaC (-O3)	0.34 ms	9.69 ms
MPMPT 705	0.47 ms	2,245.00 ms
SLLGT 80 line Lua	1.00 ms	1,000.00 ms
SLLGT 80 line Python	10.00 ms	10,000.00 ms
MPMPT 705 (-O3)	0.47 ms	2,245.00 ms

-1.0%
-1.0%
-1.0%
-1.0%
-1.0%
-1.0%
-1.0%
-1.0%