Arcgentica: Solving the hardest benchmark by letting models code

We have spent the last two years obsessing over "Chain of Thought"—the idea that if an LLM talks to itself in English, it gets smarter. Symbolica's latest work on the ARC-AGI-2 benchmark suggests a different, harder path: the best way for a model to reason isn't to talk. It's to code.

The ARC-AGI Problem

ARC-AGI is widely considered the only benchmark that actually matters for reasoning. Unlike the bar exam or coding tests, you can't memorize it. It consists of visual grid puzzles that require discovering a novel rule from a few examples and applying it. Most LLMs fail miserably at it because they rely on pattern-matching training data, whereas ARC requires on-the-fly abstraction.

The "Arcgentica" Breakthrough

Symbolica's new post details how they used their **Agentica** framework to hit a massive 85.28% score on ARC-AGI-2 (using Opus 4.6). But the interesting part isn't the score; it's the architecture.

Standard agentic workflows use "tool calling"—stateless API hits where the model sends a JSON packet and gets a result. Symbolica argues this is broken for reasoning. Instead, they gave the agent a **persistent Python REPL**.

This allows the model to:

- **Maintain state:** Keep objects in memory and mutate them (like a human working on a whiteboard).
- **Interleave thinking and doing:** Write a line of code, run it to see if the hypothesis holds, and then iterate.
- **Compose abstractions:** Define functions that become new tools for the rest of the session.

Why "State" is the missing link

This reinforces a trend I've been tracking: the move from *probabilistic generation* to *verified construction*. When a model writes English, it has no way to know if it's hallucinating until it's too late. When a model writes code in a REPL, the interpreter acts as a reality check. If the code crashes, the model knows it's wrong and tries again.

Symbolica calls this a move towards "Category Theoretic" architectures, but you don't need to understand the math to see the engineering truth: giving models a reliable, stateful workspace turns them from predictors into solvers.

Contributor: Alessandro Linzi