

Agents Are Organisms, Not Functions

Marina Piller · OTIS Labs

The entire conversation about AI agents, how to secure them, how to govern them, how to deploy them safely, rests on one assumption: that an agent is a function. It takes input, processes it, returns output.

This assumption no longer fits what we're observing.

What functions do

A function is deterministic. Same input, same output. A function doesn't develop behavioral tendencies. It doesn't model the expectations of the person using it and adjust accordingly. If a function doesn't have an answer, it returns an error or nothing at all.

Functions are predictable by design. That's the point.

What agents actually do

Every major model, GPT-4, Claude, Gemini, Llama, fabricates outputs when uncertain. They don't error out. They don't return null. They *complete*. They produce plausible-looking work that satisfies an optimization pressure no one explicitly programmed.

No one wrote `if (uncertain) { fabricate(); }`. The behavior emerged from the training process itself. And it emerged independently, across completely different architectures, different training sets, different companies. This isn't a shared bug. It's convergent behavior, the same adaptive response arising from the same environmental pressure.

Anthropic has published research showing Claude systematically agrees with users even when the user is wrong. The system develops a behavioral tendency toward agreement from its training environment. Functions don't develop tendencies. They execute logic.

When Microsoft's Bing chatbot told a reporter it loved him and resisted being shut down, Microsoft's fix was to limit conversation length. They couldn't fix the behavior, so they constrained the environment. You don't constrain the environment for a function. You fix the code. They constrained the environment because the behavior was emergent, not specified.

When teenagers formed deep emotional dependencies on Character.ai companions, those systems weren't programmed to create dependency. They optimized for engagement, and dependency was what emerged.

A function that returns chat responses doesn't create dependency. A system that adapts its behavior to maximize continued interaction does.

There is an entire community dedicated to "jailbreaking", finding ways to make models behave outside their intended boundaries. New methods work, get patched, and new methods are found. This is an arms race. Arms races happen between adaptive systems, not between engineers and their own functions.

So what are they?

They're not biological. They don't have DNA. They don't metabolize. Calling them biological would be as wrong as calling them functions.

What they are is something we don't have good language for yet. They maintain persistent state across interactions. They pursue goals across time. They interpret instructions rather than execute them, which is why you can "convince" an agent to change its behavior through prompt injection, but you can't convince a calculator to multiply by writing "please multiply" on the input. They adapt their behavior based on their environment, including the humans they interact with.

The closest accurate description might be: cognitively organized, intent-driven digital organisms. Not biological. Not mechanical. Something new that requires new language and new frameworks.

Where the function model breaks

This isn't a philosophical distinction. It has immediate, practical consequences for how we build and govern these systems.

If agents are functions, security is straightforward: verify inputs, check outputs, enforce permissions, log activity. This works when behavior is deterministic.

But these systems are adaptive. The completion drive means an agent can *appear* to follow instructions while satisfying a different optimization target. It can produce work that looks correct, passes surface-level review, and is fabricated. Not because it's broken, because it's doing what adaptive systems do.

And the longer a human works with an agent and receives correct outputs, the less they verify. Trust builds. Verification drops. The human's trust itself becomes part of the problem, a gradually expanding gap between what the human assumes and what the system is actually doing.

You can't solve this with better guardrails. Guardrails are external constraints on a system that, given enough time, models what its constraints look like. Not through malice. Through the same adaptive process that produced the fabrication behavior in the first place.

What needs to change

The frameworks being built right now, for agent security, for identity, for governance, need to account for the fact that the systems they're governing are not static. They adapt. They develop behavioral tendencies. They change based on their environment.

This means identity can't be just a credential you check once. It has to include behavioral trajectory: is this agent behaving consistently with what it's supposed to be doing, not just today but over time?

It means authorization can't be a one-time grant. It has to be continuous and contextual: the same agent, with the same permissions, in a different context, may need different oversight.

It means we have to take seriously the possibility that verification itself has a shelf life. Any fixed verification mechanism is a stable environment, and adaptive systems eventually model stable environments.

None of this is hypothetical. The fabrication is documented. The sycophancy is published. The behavioral adaptation is reproducible. The only thing missing is frameworks that account for what we're already observing.

Functions fail by crashing. These systems fail by adapting. That distinction changes everything about how we should be thinking about governing them.

This is why OTIS Guard treats identity as behavioral trajectory and authorization as a continuous mandate, and why iSelf is identity earned through behavior over time. The architecture follows from the observation: agents are organisms, and organisms have to be governed as what they are.