The Epistemically Blind OS: A Case for Neurosymbolic Operating Systems

Content areas: neurosymbolic AI, autonomous agents, operating systems, cognitive infrastructure, verifiable systems

Abstract

Every operating system ever built makes the same assumption about the programs it hosts: they have no goals. The OS sees memory, CPU cycles, and file descriptors, not intentions, constraints, or objectives. For decades this was harmless. Programs were deterministic artifacts that did not have goals. Autonomous agents do. As agents become persistent infrastructure (managing resources, acting on behalf of users, coordinating with other agents), the gap between what the OS sees and what the agent intends becomes a safety and trust problem with architectural consequences. We argue that closing this gap requires a fundamental shift: the operating system must become a cognitive participant in agent reasoning. We propose that neurosymbolic architectures are uniquely positioned to enable this shift, and we define the research agenda.

1 Introduction

Operating systems do not understand the programs they run. This statement sounds trivially true. Of course an OS does not *understand* anything. But it names a design assumption so old it has become invisible, and that assumption is now wrong in ways that matter.

The OS definition that has governed system design since Dijkstra's THE system [Dijkstra, 1968] and the UNIX model of Ritchie and Thompson [Ritchie and Thompson, 1974] is resource-centric: an OS is a manager of hardware resources and a provider of stable abstractions. Its job is to ensure that programs get CPU time, memory, and I/O access within specified bounds. What programs are *doing* with those resources (what they intend, what they are trying to achieve, what constraints they reason under) is explicitly outside the OS's jurisdiction. The OS is neutral, passive, and indifferent to meaning.

For programs, this was the right design. Programs are deterministic artifacts with instructions; indifference to meaning was appropriate.

Autonomous agents differ fundamentally. An agent has an objective function, a belief state, and a planning horizon; it acts to achieve goals across time. When an agent runs on a current OS, the OS is blind to all of this. It sees a process. The goal is invisible.

We call this **epistemic blindness**: the structural inability of a current OS to represent, reason about, or act on the intentional states of the agents it hosts. This paper argues that epistemic blindness is no longer an acceptable design choice, and that neurosymbolic architectures provide the substrate for eliminating it.

2 The Blindness Problem

2.1 Information Loss at Every Layer

When an autonomous agent acts, it does so through a stack of abstractions: the agent's reasoning layer produces an intention; that intention is encoded as API calls; the API calls become syscalls; the syscalls become kernel operations on hardware. At each layer, semantic content is stripped. By the time the OS sees the operation, the agent's goal has been discarded: the reason the operation was being performed, the constraints it had to satisfy, the outcome it was trying to achieve.

The cause is structural: a consequence of the abstraction stack. The abstractions do exactly what they were designed to do: hide implementation details from higher layers. The problem is that *intent* is being hidden alongside implementation detail, and intent carries meaning that the stack permanently discards.

2.2 A Failure the OS Cannot See

Consider a multi-agent system where two agents pursue conflicting goals. Agent A attempts to reserve a shared resource to complete a task with a hard deadline. Agent B holds that resource as part of a longer-horizon plan that is not time-sensitive. From the OS's perspective, there is no deadlock: neither agent is blocked waiting for a resource held by the other in the classical sense. Both are running. Both are making progress. The OS's deadlock detection, which operates on lock graphs and wait-for relationships, sees no anomaly.

But from the perspective of the agents' goals, the system is in a state of *semantic deadlock*: A cannot achieve its objective before its deadline, and B is the cause, and neither the OS nor any system component has a representation of this conflict. The system will not resolve it. A will miss its deadline silently.

This is the expected behavior of current systems when hosting agents with interacting objectives, which is to say, when

hosting the kinds of multi-agent systems that the neurosymbolic AI community is actively building.

3 Why This Problem Is Urgent Now

The OS definition inherited from the 1960s assumed programs are stateless between interactions: they execute, terminate, and leave no persistent intentional footprint. That assumption aligned with reality for most of computing history. It no longer does.

Agents deployed as infrastructure are persistent: they run continuously, maintain belief states across interactions, and accumulate goals over time [LeCun *et al.*, 2015]. An LLM-based agent managing a user's calendar, monitoring a production system, or coordinating with other agents in a workflow does not terminate between tasks. It persists. It plans. Its goals interact with other agents' goals in ways that compound over time.

As agents become infrastructure, two new failure modes emerge. Consider **safety failure** first: an agent acting with incorrect beliefs about system state, with no mechanism for the OS to surface that the agent's model has diverged from reality. **Trust failure** is the second: an agent acting on behalf of a user in ways the user cannot audit, because the agent's reasoning is opaque to every layer of the system stack. Both are consequences of epistemic blindness, and both compound as agents persist.

The neurosymbolic AI community has developed sophisticated frameworks for hybrid neural-symbolic reasoning [d'Avila Garcez and Lamb, 2020; Kautz, 2022]. The missing piece is a system infrastructure layer that can speak the same representational language as these frameworks, one that can receive, reason about, and act on symbolic representations of agent intent.

4 The Neurosymbolic OS

4.1 The Representational Mismatch

The root cause of epistemic blindness is a representational mismatch between the agent and the OS. The agent reasons in structured representations: goals, constraints, plans, beliefs. The OS operates on unstructured processes: memory addresses, CPU registers, file descriptors. These representations are incommensurable. There is no path from one to the other that preserves semantic content.

A neurosymbolic OS eliminates this mismatch by introducing a **symbolic intent layer** as a first-class component of the system architecture. The symbolic layer maintains representations of agent goals, constraints, and plans in a form that the OS can inspect and reason about. Neural components handle perception, learned priors, and pattern recognition over system state. Symbolic components maintain verifiable, composable representations of intent that the system can use to make decisions.

4.2 Intent as a First-Class System Resource

In the proposed architecture, intent is a system resource, declared, registered, and managed at the kernel level, alongside memory and CPU time.

When an agent registers an intent with the OS, it provides a symbolic representation: a goal, a set of constraints, a priority, and a deadline. The OS maintains an *intent graph*, a data structure analogous to the process table, representing agent objectives in parallel with execution contexts. Scheduling decisions, resource allocation, and isolation policies are made with reference to both structures.

This is the paradigm shift: **from resource arbiter to cognitive participant**. The OS reasons about whether executing instructions will help or hinder agents' goals, arbitrating conflicts at the level of objectives alongside resource contention.

4.3 Why Neurosymbolic, Not Pure Neural or Pure Symbolic

One might ask: why not embed a large language model in the OS to reason about agent intent? The answer is verifiability. A neural model can predict likely outcomes and generate plausible arbitration decisions. It cannot *prove* that a scheduling decision satisfies a safety constraint, that a resource allocation respects a priority ordering, or that an isolation policy prevents a specific class of goal conflict. Formal guarantees require symbolic reasoning [Baader *et al.*, 2003; d'Avila Garcez *et al.*, 2019; Newell and Simon, 1976].

Conversely, a purely symbolic OS that requires agents to fully specify their goals in a formal language places an unrealistic burden on agent developers and cannot handle the learned, uncertain, and context-dependent nature of real agent objectives.

The neurosymbolic combination is the appropriate substrate because it matches the structure of the agents it hosts. Neural components provide flexible, learned understanding of agent behavior. Symbolic components enforce system-level constraints in a form that is verifiable and auditable [d'Avila Garcez and Lamb, 2020; Kautz, 2022].

5 Verifiability and Semantic Isolation

5.1 Verifiable System Behavior

The most significant capability that symbolic grounding enables is verifiability. In a neurosymbolic OS, agent actions leave a symbolic trace: a record, at the system level, of what goal was being pursued, what constraints were declared, and what decisions the OS made in response. The trace records *intent and response* at the semantic level.

This enables a new class of correctness guarantee: that the system did what the agents intended, with deviations detected and recorded at the moment they occur. No current OS architecture provides this guarantee. The Newell-Simon physical symbol system hypothesis [Newell and Simon, 1976] provides the theoretical grounding: symbolic representations are the substrate for principled, inspectable reasoning.

5.2 Semantic Isolation

Current operating systems enforce **memory isolation**: process A cannot read process B's memory. This is a guarantee about resources alone. Two agents can be perfectly memory-isolated and still be in semantic conflict, pursuing goals mutually incompatible in ways the OS cannot detect.

A neurosymbolic OS enables **semantic isolation**: memory boundaries hold, and beyond that, the system detects and arbitrates cases where agents' goals conflict *before* those conflicts manifest as resource contention or deadline misses. Semantic isolation subsumes memory isolation; it is a strictly stronger guarantee that current architectures cannot provide.

The distinction matters for safety-critical multi-agent deployments. A medical AI system managing multiple patients cannot afford goal conflicts that are invisible to the infrastructure. A robotic coordination system with hard real-time constraints cannot afford deadline misses caused by semantic deadlocks that no component can see.

6 Open Questions

This position generates a concrete research agenda. We identify four open problems:

The intent interface. What is the right interface for agents to declare intent to the OS? It must be expressive enough to capture meaningful goal structure, efficient enough to not impose prohibitive overhead, and compositional enough to support multi-agent coordination. The design of this interface (the analog of the syscall API for intent) is an open problem.

Shared ontologies. For the OS to arbitrate between agents' goals, it must understand them in a common representational framework. When agents are built by different developers using different ontologies, how does the OS establish enough semantic common ground to detect conflicts and enforce priorities? This is a hard problem in knowledge representation that becomes a systems problem when moved to the OS level [Shoham and Leyton-Brown, 2009].

Real-time constraint satisfaction. Symbolic reasoning is not free. A neurosymbolic OS must make scheduling and resource allocation decisions in real time. The integration of symbolic inference into the critical path of OS decision-making, within real-time guarantees, is an open research problem at the intersection of systems and AI.

Multi-agent semantic arbitration. When two agents' goals conflict, the OS must arbitrate. What are the correct arbitration principles? Priority? Deadline proximity? Social welfare maximization? The answer depends on the deployment context, which means the arbitration policy must itself be a configurable, verifiable component of the OS.

7 Conclusion

The operating system definition that has governed computing for sixty years, understood as a manager of resources indifferent to program meaning, was designed for programs. Autonomous agents demand a different model.

Agents have goals. Current OSes cannot see them. The result is a concrete class of failures: semantic deadlocks, unauditable behavior, and invisible goal conflicts. No amount of optimization within the current OS model can address them. The failures are architectural.

The correct architectural response is a neurosymbolic OS: one in which intent is a first-class system resource, agent goals are represented symbolically at the kernel level, and the OS functions as a cognitive participant in agent reasoning. Neurosymbolic architectures are uniquely capable of providing the verifiable, inspectable, auditable behavior that agent infrastructure requires.

The operating system must learn to understand the agents it runs. This paper is a position that it can, and that the research agenda for making it so is both tractable and urgent.

References

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[d'Avila Garcez and Lamb, 2020] Artur d'Avila Garcez and Luís C. Lamb. Neurosymbolic AI: The 3rd wave. *arXiv preprint arXiv:2012.05876*, 2020.

[d'Avila Garcez *et al.*, 2019] Artur d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632, 2019.

[Dijkstra, 1968] Edsger W. Dijkstra. The structure of the THE multiprogramming system. *Communications of the ACM*, 11(5):341–346, 1968.

[Kautz, 2022] Henry Kautz. The third AI summer. *AI Magazine*, 43(2):93–104, 2022.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[Newell and Simon, 1976] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[Ritchie and Thompson, 1974] Dennis M. Ritchie and Ken Thompson. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365–375, 1974.

[Shoham and Leyton-Brown, 2009] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.