

Can Formal Verification Match AI Code Generation Speed?

Mingyu Li Zhanbo Wang

Key Laboratory of System Software (Chinese Academy of Sciences)
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences

1 The Verification Gap

In 2009, verifying the seL4 microkernel cost 11 person-years of proof for roughly 8,700 lines of C [11]. In 2026, LLMs can draft filesystems [12], kernel extensions [22], and everyday kernel code [7] in minutes. Generation got orders of magnitude cheaper; verification barely moved. Classical verified kernels, such as seL4 [11], CertiKOS [9], assume a static, human-curated codebase: verify once, then freeze. Even push-button pipelines re-verify the whole kernel per version [16], and specification-guided generation still leans on empirical assurance for the final mile [12]. Testing, simulation, and LLM-as-judge narrow the resulting *verification gap* but cannot close it: a passing suite is still not a proof. Can we redesign the OS so that verification becomes incremental and autonomous rather than external and human-paced?

2 VERIOS: Verification by Construction

Yes. Decidability is a design choice. Every automatic verifier that survived production bought it by selling expressiveness: eBPF admits only kernel extensions it can bound [19]. So instead of scaling verification up to arbitrary LLM output, VERIOS scales generation down to a verifiable core through three interlocking restrictions.

(1) All shared state is relational. All shared state (e.g., resources, tasks, capabilities, and sensor readings) lives in transactional relations, following the DBMS-oriented OS line [1, 18]. Safety invariants become constraints checked at commit time over a finite instance (cheap model checking, never state invention); every reader sees an MVCC snapshot [3]; ACID atomicity [8] guarantees a faulty AI component aborts without leaving the system half-updated.

(2) All logic is pure and effect-tracked. Each AI-generated component is a terminating function $\lambda : \text{State} \times \text{Event} \xrightarrow{\Sigma} \text{State}' \times \text{Intent}$ with an explicit effect signature $\Sigma = (R, W)$, in the spirit of purely functional systems [10] and eBPF’s bounded execution [19]: no unbounded recursion, no ambient syscalls. Parallel branches compose only over disjoint write sets, governed by machine-checked algebraic laws [21]; irreversible effects accumulate as typed *intents* and are released only after the enclosing DAG verifies, a cyber-physical two-phase commit.

(3) Verification splits into local summaries and global fixed-point checks. Rice’s theorem rules out general verification [17], so VERIOS factors it. A *local* phase applies model checking [4] to each bounded function and memoizes a compact summary (effects, types, bounds); a *global* phase evaluates human-written Datalog invariants by bottom-up fixed point—polynomial in data—re-verifying only the affected sub-DAG incrementally [14, 15, 20]. For example, “never torque a joint while e-stop is active” is one rule: $\text{violation} := \text{intent}(\text{torque}, _) , \text{estop}(\text{active})$, and intents release only if violation is empty. Invariants stay a human-curated root of trust; LLM proposals are advisory until ratified.

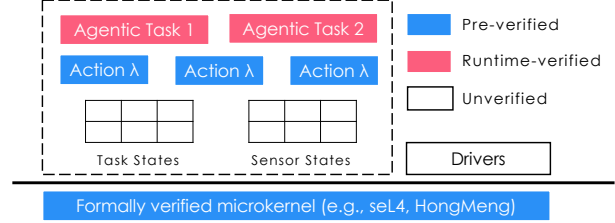


Figure 1: VERIOS in an embodied AI setting. The microkernel and low-level controllers are verified offline; only AI-synthesized task DAGs undergo online two-phase verification.

3 Case Study: ROS-Next for Embodied AI

Embodied AI is the most unforgiving case: a verification failure can cause physical damage. ROS-Next replaces the ROS 2 control plane [13] with VERIOS (see Figure 1). ROS 2’s decentralized publish-subscribe model allows different nodes to observe inconsistent snapshots of pose, velocity, or joint state [2], and AI-generated behavior trees [6] are today validated largely by simulation, which is inherently incomplete. Under VERIOS, the entire control state lives in relational tables; every component reads a transactionally consistent snapshot, and failed transactions roll back. High-level tasks are compiled into λ -DAGs that undergo the full two-phase verification before any intent is released; when a behavior tree is updated by an LLM, only the affected sub-DAG is re-verified. To satisfy strict real-time requirements we deliberately separate concerns by frequency: high-frequency control loops (≥ 100 Hz) use static, offline-verified topologies and *never enter the runtime verification path*, while low-frequency planning and agentic layers (3–10 Hz) perform full online two-phase verification before committing intents. The stable foundation remains a conventionally verified microkernel (seL4 [11] or HongMeng [5]); VERIOS concentrates its incremental verification on the rapidly changing, AI-generated layer.

4 Sandbox Semantics, Not Just Resources

Classical OSes sandboxed *resources*; an OS that hosts AI-generated code must also sandbox *semantics*: what a component may mean, not just what it may touch. VERIOS makes incremental verification a native OS primitive: invariants stay human, fast paths stay offline, and the language of AI components stays deliberately small, so local checks stay decidable and global checks stay polynomial. Generation is cheap; trust is expensive. Let the models dream, and make the automatic verifier the adult in the room.

Acknowledgments

This work was partially funded by the National Natural Science Foundation of China (No. 62502510), and the CAS Talent Introduction Project, Institute of Software, Chinese Academy of Sciences (No. E5YD230101).

References

- [1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- [2] Soroush Bateni, Marten Lohstroh, Hou Seng Wong, Hokeun Kim, Shaokai Lin, Christian Menard, and Edward A. Lee. 2023. Risk and Mitigation of Nondeterminism in Distributed Cyber-Physical Systems. In *21st ACM-IEEE International Symposium on Formal Methods and Models for System Design, MEMOCODE 2023, Hamburg, Germany, September 21-22, 2023*. ACM / IEEE, 1–11.
- [3] Philip A. Bernstein and Nathan Goodman. 1983. Multiversion concurrency control—theory and algorithms. *ACM Trans. Database Syst.* 8, 4 (Dec. 1983), 465–483.
- [4] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. 1999. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference* (New Orleans, Louisiana, USA) (DAC '99). Association for Computing Machinery, New York, NY, USA, 317–320.
- [5] Haibo Chen, Xie Miao, Ning Jia, Nan Wang, Yu Li, Nian Liu, Yutao Liu, Fei Wang, Qiang Huang, Kun Li, Hongyang Yang, Hui Wang, Jie Yin, Yu Peng, and Fengwei Xu. 2024. Microkernel Goes General: Performance and Compatibility in the HongMeng Production Microkernel. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 465–485.
- [6] Michele Colledanchise and Petter Ögren. 2018. Behavior Trees in Robotics and AI.
- [7] Jonathan Corbet. 2025. Supporting kernel development with large language models. <https://lwn.net/Articles/1026558/>
- [8] Jim Gray. 1981. The transaction concept: virtues and limitations (invited paper). In *Proceedings of the Seventh International Conference on Very Large Data Bases - Volume 7* (Cannes, France) (VLDB '81). VLDB Endowment, 144–154.
- [9] Ronghui Gu, Zhong Shao, Hao Chen, Xiongnan (Newman) Wu, Jieung Kim, Vilhelm Sjöberg, and David Costanzo. 2016. CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 653–669.
- [10] Thomas Hallgren, Mark P. Jones, Rebekah Leslie, and Andrew Tolmach. 2005. A principled approach to operating system construction in Haskell. *SIGPLAN Not.* 40, 9 (Sept. 2005), 116–128.
- [11] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. 2009. seL4: formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles* (Big Sky, Montana, USA) (SOSP '09). Association for Computing Machinery, New York, NY, USA, 207–220.
- [12] Qingyuan Liu, Mo Zou, Hengbin Zhang, Dong Du, Yubin Xia, and Haibo Chen. 2026. Sharpen the Spec, Cut the Code: A Case for Generative File System with SYSSPEC. In *24th USENIX Conference on File and Storage Technologies (FAST 26)*. USENIX Association, Santa Clara, CA, 291–311.
- [13] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. 2022. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics* 7, 66 (2022), eabm6074. [arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.abm6074](https://www.science.org/doi/pdf/10.1126/scirobotics.abm6074)
- [14] David Maier, K. Tuncay Tekle, Michael Kifer, and David S. Warren. 2018. *Datalog: concepts, history, and outlook*. Association for Computing Machinery and Morgan & Claypool, 3–100.
- [15] Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martin Abadi. 2013. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (Farmington, Pennsylvania) (SOSP '13). Association for Computing Machinery, New York, NY, USA, 439–455.
- [16] Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, and Xi Wang. 2017. Hyperkernel: Push-Button Verification of an OS Kernel. In *Proceedings of the 26th Symposium on Operating Systems Principles* (Shanghai, China) (SOSP '17). Association for Computing Machinery, New York, NY, USA, 252–269.
- [17] H. G. Rice. 1953. Classes of Recursively Enumerable Sets and Their Decision Problems. *Trans. Amer. Math. Soc.* 74, 2 (1953), 358–366. <http://www.jstor.org/stable/1990888>
- [18] Athinagoras Skiadopoulos, Qian Li, Peter Kraft, Kostis Kaffes, Daniel Hong, Shana Mathew, David Bestor, Michael Cafarella, Vijay Gadepally, Goetz Graefe, Jeremy Kepner, Christos Kozyrakis, Tim Kraska, Michael Stonebraker, Lalith Suresh, and Matei Zaharia. 2021. DBOS: a DBMS-oriented operating system. *Proc. VLDB Endow.* 15, 1 (Sept. 2021), 21–30.
- [19] Dave Thaler. 2024. BPF Instruction Set Architecture (ISA). <https://www.rfc-editor.org/rfc/rfc9669.html>
- [20] J. D. Ullman. 1989. Bottom-up beats top-down for datalog. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Philadelphia, Pennsylvania, USA) (PODS '89). Association for Computing Machinery, New York, NY, USA, 140–149.
- [21] George Zakhour, Pascal Weisenburger, and Guido Salvaneschi. 2024. Automated Verification of Fundamental Algebraic Laws. *Proc. ACM Program. Lang.* 8, PLDI, Article 178 (June 2024), 24 pages.
- [22] Yusheng Zheng, Yiwei Yang, Maolin Chen, and Andrew Quinn. 2024. Kgent: Kernel Extensions Large Language Model Agent. In *Proceedings of the ACM SIGCOMM 2024 Workshop on EBPF and Kernel Extensions* (Sydney, NSW, Australia) (eBPF '24). Association for Computing Machinery, New York, NY, USA, 30–36.