POST-AGI SYSTEMS NEED A "TRUTH STACK," NOT JUST BETTER MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

As models become broadly competent and increasingly agentic, failures shift from missing capabilities to missing *epistemic guarantees*: unclear specifications, uncheckable reasoning, and un-auditable interactions with tools and the world. We argue that mid-term post-AGI systems require a *truth stack*—a layered substrate that turns model outputs into *checkable claims* backed by persistent evidence. The truth stack combines (i) specification interfaces, (ii) proof- and test-carrying reasoning, (iii) calibrated statistical uncertainty, and (iv) provenance/supply-chain integrity, all bound to risk-tiered decision gates. We outline a post-AGI lab architecture where autonomous research agents produce reusable certificates (proofs, tests, datasets, traces) consumed by downstream agents and auditors. The central challenge is *scalable verification*: evidence must strengthen with autonomy and impact, while remaining computationally affordable.

1 INTRODUCTION

Scaling has delivered rapid gains in foundation models (Vaswani et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022) and enabled tool-using agents (Schick et al., 2023; Yao et al., 2022). Yet reliability remains brittle: in open-ended settings, errors often arise from *system interactions* (tool drift, stale data, misinterpreted results, hidden side effects), not just from generation mistakes.

We propose that post-AGI systems be engineered around a *truth stack* (Figure 1), rather than relying on improved base models alone. The stack enforces a discipline: every consequential output is a claim paired with evidence; every artifact has lineage; and every high-impact action passes risk-aware gates. This is a lab-centric view: early post-AGI deployments are likely to be autonomous R&D systems that write code, run experiments, and propose theories.

Two properties make the truth stack distinct from "add more evals." First, it is *compositional*: claims can be reused as verified subcomponents of later work, so trust compounds rather than resetting per run. Second, it is *operational*: evidence binds to permissions and actions, not just to paper claims. In effect, the truth stack is an operating layer for autonomous cognition.

2 TRUTH STACK: LAYERS AND INTERFACES

A truth stack is infrastructure for *compositional trust*. Its core layers are:

(L1) Specifications. Interfaces that turn ambiguous tasks into explicit contracts: goals, invariants, admissible tools, and cost/risk budgets. In software this includes contracts, types, and resource caps; in science it includes protocol templates, pre-registered analyses, and decision rules. Critically, specifications must be *action-addressable*: they should constrain what tools may be invoked, with which parameters, and what evidence is required for escalation. Without this interface, discussions of "alignment" collapse into post-hoc interpretation.

(L2) Evidence. Claims require evidence proportional to impact: formal proofs (when feasible), SMT/property checks (Katz et al., 2017; 2019), unit/integration tests, redundant execution, and calibrated uncertainty (Lakshminarayanan et al., 2017; Vovk et al., 2005). Heuristic deliberation (self-consistency, tree search) can reduce variance (Wang et al., 2023; Yao et al., 2023), but is treated as an evidence producer whose outputs remain checkable.

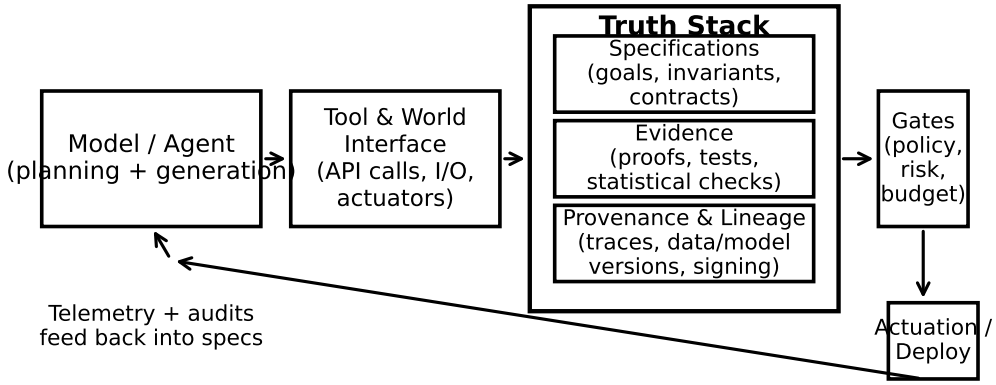

Figure 1: Truth stack: specifications, evidence, provenance, and risk-aware gates for tool-using systems.

We find it useful to represent any consequential output as a *decision object*: a structured bundle containing (i) a proposed action or claim, (ii) the spec clauses it relies on, (iii) the evidence it provides, and (iv) a replay recipe. Decision objects are the primary unit passed between agents and auditors.

Decision objects also make *verification debt* measurable. Whenever downstream work depends on a decision object whose evidence is stale (dependencies changed) or whose uncertainty assumptions no longer hold, the system accrues debt. Debt can be budgeted like compute: gates may prohibit compounding on high-debt artifacts, or require automatic re-validation before reuse. This framing turns reliability into an operational quantity rather than a post-hoc critique.

Finally, the truth stack benefits from *independence*. Many real-world failures are correlated: the same model and retrieval stack will make the same error repeatedly. For high-impact actions, the evidence layer should therefore favor non-identical checks (distinct solvers, redundant tests, or separate implementations) and record this diversity in the certificate metadata.

(L3) Provenance. A certificate is only as strong as its dependencies. Provenance standards (Lebo et al., 2013) and software supply-chain practices (OpenSSF, 2023) motivate cryptographic hashing/signing, explicit dependency graphs, and reproducible builds. In an autonomous lab, provenance must extend to datasets, instruments, tool versions, and even model configurations. Provenance is not optional metadata: it is the mechanism by which certificates remain valid (or are invalidated) under drift.

(L4) Gates. Evidence must bind to decisions. Gates implement risk-tiered policies: higher autonomy/impact requires stronger evidence and stricter tool permissions. Governance frameworks become enforceable technical constraints (National Institute of Standards and Technology, 2023; National Aeronautics and Space Administration, 2024).

3 Proof-Carrying Reasoning: A Practical "Truth Layer"

Proof-carrying code (Necula, 1997) provides a guiding abstraction: an untrusted producer emits code plus a machine-checkable proof that the code satisfies a safety policy; a small trusted checker verifies the proof. We generalize this to *proof-carrying reasoning*: an agent emits actions plus certificates (proofs, tests, traces) that justify those actions under a declared specification.

Symbolic–neural theorem proving is a pragmatic path to such certificates. Proof assistants (e.g., Lean) provide small kernels and strong foundations (de Moura et al., 2015; The mathlib Community, 2020). Language models can propose proof steps while retrieval and tactic automation close the loop (Polu &

Sutskever, 2020; Yang et al., 2023). Program-search pipelines suggest similar synthesis+verification loops for discovery tasks (Romera-Paredes et al., 2024).

A practical PCR interface is a *certificate hierarchy*. At the bottom are structural guarantees (types, schemas, unit consistency, resource bounds) that are cheap to check and prevent entire classes of mistakes. Above that are bounded property checks (SMT certificates, fuzzing, regression tests) and, when feasible, machine-checked proofs. At the top are action-level safety cases that bind evidence to an explicit specification and to a concrete execution context. This hierarchy is crucial because most high-impact failures are mundane (a wrong unit conversion, a stale dataset, an out-of-date tool), not deep logical errors.

PCR also clarifies what must be trusted. The goal is a small trusted computing base consisting of verifiers (proof kernels, certificate checkers), policy interpreters (which evidence is required for which action), and integrity mechanisms (hashing/signing and replay). Large models remain untrusted producers. This separation supports defense-in-depth: even if a model is misled or strategically optimizing, it cannot pass gates without producing checkable artifacts.

Finally, PCR extends naturally to tool use. Tool calls should produce signed traces and replay recipes, so that a verifier can re-run critical steps in a sandboxed environment and confirm that outputs match the claimed evidence. In post-AGI labs, many "proofs" will take this form: executable analyses that are deterministic, version-pinned, and accompanied by checks that validate intermediate invariants. PCR systems should be *hybrid* rather than absolutist. Formal proofs are strongest when specifications are crisp (compilers, kernels) (Leroy, 2009; Klein et al., 2009). For empirical claims, the certificate may be a calibrated decision rule or a robust test suite rather than a theorem. The key is that the verifier can mechanically validate the certificate and bind it to the decision gate.

Two additional design constraints are central in a post-AGI setting. First, *formalization* must be treated as part of verification: a system can "prove the wrong statement" if it silently shifts meaning while translating informal claims into formal ones. Second, proof search is compute-intensive and should be governed like any other scarce resource. A truth stack therefore needs an *evidence scheduler*: allocate proof search, testing, and replication budgets only where they change downstream decisions.

4 Post-AGI Lab Architecture Blueprint

To scale, verification must support *reuse*. We propose a certificate lifecycle (Figure 2): agents produce artifacts and traces; independent verifiers generate certificates with dependency metadata; downstream consumers validate and reuse certificates cheaply; and certificates are invalidated when dependencies change.

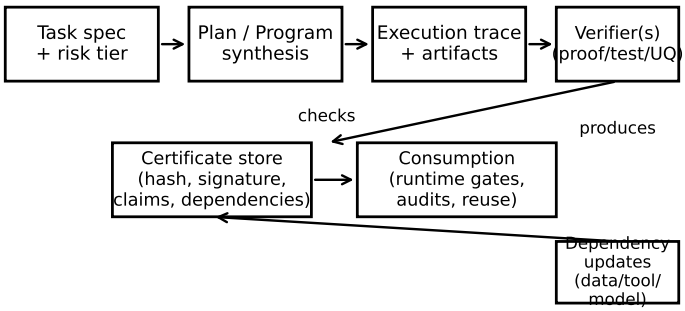

Figure 2: Certificate lifecycle and invalidation under dependency drift.

A post-AGI lab can be organized into five planes: (1) *control* (specs and policies), (2) *execution* (agents and tools, with structured traces), (3) *truth* (verification services and provenance signing), (4) *memory* (artifact registry with lineage graphs and access controls), and (5) *governance* (audit queries, monitoring, and policy updates). In this architecture, verification services are independent compo-

nents: proof checkers, test runners, simulation replayers, and calibration routines. Independence matters: for high-risk gates, the lab should prefer at least two non-identical checks (e.g., redundant test suites, distinct solvers, or separate implementations) to reduce correlated failure.

Autonomous chemistry illustrates the stakes: agents can propose and run experiments, but claims should only ship when protocols and evidence thresholds are met (Burger et al., 2020; Boiko et al., 2023). More broadly, post-AGI labs must manage *verification debt*: the latent risk introduced when downstream work depends on stale certificates (dependencies drifted) or weak evidence. Verification debt is a system-level metric that can be policed by gates (tighten permissions) and schedules (prioritize re-validation).

Systems efficiency matters because evidence consumes compute. Serving improvements (Dao et al., 2022; Kwon et al., 2023) and sustainability constraints (Schwartz et al., 2020) push toward impact-weighted evidence schedules. Evidence schedules provide a simple policy surface: as impact increases, the system increases verification strength, narrows tool scopes, and demands stronger certificates.

5 Research Directions

Verification-aware scaling laws. We need laws for evidence vs autonomy, analogous to capability vs compute (Kaplan et al., 2020; Hoffmann et al., 2022). One concrete target is a reliability budget: given an action horizon, tool set, and acceptable failure probability, how much verification compute is required, and where should it be spent?

Composable certificates. How do we decompose open-ended research into locally verifiable claims that remain globally useful?

Tool drift and invalidation. How can we minimize re-validation cost when dependencies change, while avoiding silent regressions (OpenSSF, 2023; Lebo et al., 2013)?

Hybrid evidence aggregation. How should formal and statistical evidence compose under distribution shift (Lakshminarayanan et al., 2017; Vovk et al., 2005)?

6 Conclusion

Post-AGI systems will be judged less by raw capability than by the trustworthiness of their actions. A truth stack—specifications, evidence, provenance, and gates—offers a concrete path to scalable control for highly capable agents.

References

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0.

Benjamin Burger, Phillip M. Maffettone, Vladimir V. Gusev, Charlie M. Aitchison, Yu Bai, Xiaobo Wang, Xiaoyan Li, Bruce M. Alston, Ben Li, Roger Clowes, Nicola Rankin, Brandon Harris, Reiner Sebastian Sprick, and Andrew I. Cooper. A mobile robotic chemist. *Nature*, 583:237–241, 2020. doi: 10.1038/s41586-020-2442-2.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL `https://arxiv.org/abs/2205.14135`.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction – CADE-25*, pp. 378–388, 2015. doi: 10.1007/978-3-319-21401-6_26.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre.

Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. URL https://arxiv.org/abs/2203.15556.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification (CAV)*, pp. 97–117, 2017. doi: 10.1007/978-3-319-63387-9_5.

Guy Katz, Derek Huang, David Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljko, David L. Dill, and Mykel J. Kochenderfer. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification (CAV)*, pp. 443–452, 2019. doi: 10.1007/978-3-030-25540-4_26. URL https://arxiv.org/abs/1905.00915.

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. sel4: Formal verification of an os kernel. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP)*, pp. 207–220, 2009. doi: 10.1145/1629575.1629596.

Woosuk Kwon, Zhuohan Li, Shawn Zhuang, Yuzhu Sheng, Lianmin Zheng, Cody Yu, Joseph E. Gonzalez, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

Timothy Lebo, Satya S. Sahoo, and Deborah L. McGuinness. Prov-o: The prov ontology. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-o/.

Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7): 107–115, 2009. doi: 10.1145/1538788.1538814.

National Aeronautics and Space Administration. Nasa-std-7009b: Models and simulations credibility assessment and reporting. Technical report, NASA, 2024. URL https://standards.nasa.gov/standard/nasa/nasa-std-7009.

National Institute of Standards and Technology. Artificial intelligence risk management framework (ai rmf 1.0). Technical Report NIST AI 100-1, U.S. Department of Commerce, National Institute of Standards and Technology, 2023. URL https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf.

George C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pp. 106–119, 1997. doi: 10.1145/263699.263712.

OpenSSF. Supply-chain levels for software artifacts (slsa) v1.0. Specification, 2023. URL https://slsa.dev/spec/v1.0/.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020. URL https://arxiv.org/abs/2009.03393.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. doi: 10.1145/3381831.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, 2020. doi: 10.1145/3372885.3373824. URL https://arxiv.org/abs/1910.09336.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005. doi: 10.1007/b106715.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Kaiyu Yang, Jia Deng, Xiongfeng Yue, Ziang Zhang, Denny Zhou, Andrew Li, Alex Smola, Dawn Song, and Lingpeng Kong. Leandojo: Theorem proving with retrieval-augmented language models. *arXiv preprint arXiv:2306.15626*, 2023. URL https://arxiv.org/abs/2306.15626.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. URL https://arxiv.org/abs/2305.10601.