Who Verifies the Self-Modifier? Property Preservation in Agentic AI Systems

Deepinder Sidhu
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
sidhu@umbc.edu

***Abstract*—Recent advances in agentic AI systems and AI-assisted software development raise a fundamental question: if an autonomous system modifies its own software, algorithms, protocols, tests, or decision processes, how can it be established that the resulting system continues to satisfy its required properties and constraints? While much attention has been devoted to capability improvement and code generation, comparatively little attention has been devoted to preservation of correctness properties following autonomous modification. This note introduces the property preservation problem and argues that preservation of system properties and test adequacy may represent fundamental challenges for self-modifying agentic systems.**

I. Introduction

Recent reports from major AI laboratories indicate a rapid increase in AI-generated software. Table I summarizes publicly reported figures from Google and Anthropic. These observations suggest a trend toward increasing participation of autonomous systems in software generation, modification, testing, and maintenance.

TABLE I
Reported AI-Generated Code Percentages

Organization	Reported Code	Context
Google (2024)	$> 25\%$	New code generated by AI, then reviewed and accepted by engineers [1].
Google (2026)	75%	New code reported as AI-generated and approved by engineers [2].
Anthropic (2026)	$> 80\%$	Code merged into Anthropic's codebase reported as authored by Claude [3].

Decades of research in software engineering, protocol engineering, distributed systems, and formal methods have produced methodologies for designing systems that satisfy well-defined correctness properties [4]–[7]. Protocol testing research has shown that testing hierarchies can be constructed over behaviors of increasing complexity, yielding systematic exploration of system behavior and fault exposure [8]. Recent work extends these concepts to agentic AI systems through composition-based testing hierarchies constructed from behavioral primitives and prompt sequences [9].

The question addressed in this note is simple: if an agentic AI system modifies software, protocols, tests, or decision processes, who verifies that required properties continue to hold?

TABLE II
Direction of Increasing AI Participation in Software Modification

Development Stage	Human Oversight Question
Traditional development	Humans design, implement, test, review, and deploy software.
AI-assisted development	Humans define tasks and review AI-generated suggestions.
Agent-generated code	Humans supervise, review, approve, and integrate generated code.
Self-modifying systems	Who verifies that required properties and tests remain valid?

II. The Property Preservation Problem

Following Alpern and Schneider, let σ denote an execution and let P denote a property. We write

$$\sigma \models P$$

to indicate that execution σ satisfies property P.

Suppose a program that generates execution σ undergoes autonomous modification and subsequently produces execution σ'. If

$$\sigma \models P$$

the central question becomes whether

$$\sigma' \models P$$

still holds.

For safety, liveness, recurrence, convergence, authorization, trust, security, timing, resource, and other required properties, who verifies that

$$\sigma' \models P$$

remains true after autonomous modification?

In particular, preservation of Safety ($\Box P$), Liveness ($\Diamond P$), Recurrence ($\Box\Diamond P$), and Convergence ($\Diamond\Box P$) properties remains essential if autonomous systems are to maintain operational trustworthiness following modification.

TABLE III
THE PROPERTY PRESERVATION PROBLEM

Property or Constraint	Before Modification	After Modification
Safety ($\Box P$)	Verified	Must Revalidate
Liveness ($\Diamond P$)	Verified	Must Revalidate
Recurrence ($\Box\Diamond P$)	Verified	Must Revalidate
Convergence ($\Diamond\Box P$)	Verified	Must Revalidate
Security constraints	Verified	Must Revalidate
Authorization constraints	Verified	Must Revalidate
Trust assumptions	Verified	Must Revalidate
Timing constraints	Verified	Must Revalidate
Resource constraints	Verified	Must Revalidate
Test adequacy	Established	Revalidate

A modification that improves a local objective may nevertheless introduce global violations of required properties. Improved performance, lower cost, or greater capability do not imply preservation of safety, liveness, authorization, trust, security, timing, or other constraints.

III. THE TEST PRESERVATION PROBLEM

The same concern applies to testing. A test hierarchy T may be adequate for detecting violations of property P in executions of the original system. Following autonomous modification, the system may generate new behaviors, states, transitions, communication patterns, authorization paths, or failure modes that were not considered when T was constructed.

Consequently, a second question arises: who verifies that the original test hierarchy remains adequate for detecting violations of P in the modified system?

This observation is consistent with classical protocol testing theory. Testing hierarchies are constructed relative to the behavioral structure of the system under test. Changes to states, transitions, dependencies, interaction sequences, or execution semantics may invalidate assumptions underlying existing test sequences and coverage arguments [8]. Similarly, agentic testing hierarchies are constructed from behavioral primitives and admissible compositions; modifications that alter those primitives or their interactions may require regeneration of the corresponding hierarchy and associated test sequences [9].

IV. WHY EXISTING PRACTICES ARE INSUFFICIENT

Human review. Traditional software development assumes human review of modifications. As the percentage of AI-generated code increases, complete human review becomes increasingly difficult to sustain.

Regression testing. Regression testing demonstrates that previously selected tests continue to pass. It does not establish that the original testing hierarchy remains adequate for the modified system.

Performance metrics. A self-generated modification should not be considered acceptable merely because it improves a performance metric. The modification must also preserve the properties, constraints, and testing assumptions upon which correct operation depends.

V. CONCLUSION

As agentic systems increasingly participate in software generation, testing, protocol design, and autonomous modification, property preservation becomes a foundational assurance challenge.

The challenge is no longer merely whether autonomous systems can generate software. The challenge is whether autonomous systems can demonstrate that required properties, constraints, guarantees, and testing assumptions remain valid following modification.

This challenge can be viewed as a specific instance of the broader Assurance Gap: increasing autonomous capability without a corresponding ability to establish preservation of required properties and guarantees.

Property preservation may therefore become one of the defining problems of trustworthy autonomous systems.

Who verifies the self-modifier?

REFERENCES

[1] A. Inc., "Alphabet third quarter 2024 earnings call," 2024, google CEO Sundar Pichai stated that more than 25 percent of new Google code was AI-generated and then reviewed by engineers.

[2] Google, "Reported growth in ai-generated code at google," 2026, public reporting indicates increasing AI-generated code at Google beyond the 2024 level.

[3] Anthropic, "When ai builds itself," Anthropic Institute, 2026, discussion of recursive self-improvement and AI-assisted AI development.

[4] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Transactions on Software Engineering*, vol. SE-3, no. 2, pp. 125–143, 1977.

[5] B. Alpern and F. B. Schneider, "Defining liveness," *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, 1985.

[6] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, 2nd ed. MIT Press, 2018.

[7] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*. Springer, 1995.

[8] D. P. Sidhu, H. Moteller, and R. Vallurupalli, "On testing hierarchies for protocols," *IEEE/ACM Transactions on Networking*, vol. 1, no. 5, pp. 590–600, 1993.

[9] D. P. Sidhu, "Testing hierarchies of agentic ai systems: A composition-based framework," *Preprint*, 2026.