

TRICE: Deterministic Context Control for Live Software Agents

TraceRazor Research - 2026-06-21

Abstract

TRICE makes context assembly, evidence manifests, live verification, and acceptance gates deterministic. Verifier durations are normalized before hashing and wall-clock fields are excluded from evidence. On six bundled live repository tasks, TRICE reduces measured input context by 78.6% (95% bootstrap CI: 76.8% to 80.3%) while preserving executable task success.

Method

TRICE chooses keep, extract, summarize, mask-with-receipt, anchor-prefix, or lazy-recall actions under a token budget. Acceptance requires live verifier pass preservation and measured input savings above the user-conditioned target.

Related Work

TRICE combines software-agent benchmark lessons from SWE-bench, SWE-agent, tau-bench, AgentBench, and ToolLLM with prompt compression and serving-efficiency work such as LLMLingua, LongLLMLingua, Prompt Cache, PagedAttention, StreamingLLM, and H2O. The paper treats compression as a verifier-backed decision-preservation contract rather than a replay-only token counter.

Task	Baseline	TRICE	Savings	Pass
csv-filter	1895	430	77.3%	yes
dedupe-helpers	1826	361	80.2%	yes
fix-imports	1806	341	81.1%	yes
fix-offby-one	2527	634	74.9%	yes
implement-median	1889	424	77.6%	yes
rename-api	1814	349	80.8%	yes
Mean	--	--	78.6%	yes

Deterministic Claim Card

The generated claim card reports claim level smoke, claim allowed no, and determinism contract score 100/100. It binds the suite result and evidence manifest hashes and lists the non-claims that prevent over-selling the current smoke evidence.

Suite Readiness Preflight

The generated readiness card reports level smoke_ready, pilot-ready no, claim-ready no, and readiness score 60/100. It is a no-execution preflight gate: outcome claims still require live suite results, verified manifests, bundles, and a claim card.

Protocol Lock

The generated protocol lock reports level smoke_protocol_locked, claim allowed by protocol no, and protocol score 81/100. It pre-registers the primary metric, pass-preservation guardrail, clustered confidence rule, required task clusters, replicates, locked sources, adapter profiles, receipts, claim card, and artifact card before outcome

evidence.

Statistical Design Card

The generated design card reports level `smoke_design_observed`, claim-design ready no, and design score 65/100. It uses task-cluster means to project whether the locked claim sample size would clear the savings target while preserving the non-claim boundary when held-out design requirements are missing.

Reviewer Reproduction Packet

TRICE emits a separate reproduction card after the paper manifest is written. The card binds readiness, protocol, design, claim, bundle, paper-manifest, and paper-result inputs with SHA-256 receipts and lists exact verifier commands. It is intentionally outside the paper manifest to avoid a hash cycle; the final artifact card binds both the paper manifest and reproduction card.

Public Contract Card

The generated contract card reports level `library_contract_locked` and score 100/100. It binds SemVer, public imports, tracerazor-trice commands, shipped schemas, examples, docs, and package metadata so release compatibility claims have a declared API boundary.

Release Trust Card

TRICE treats distribution readiness as a separate machine-readable contract from benchmark evidence. The release card snapshots tracerazor-trice doctor, binds the artifact, reproduction, and contract cards, records package metadata, and refuses `public_release_ready` until PyPI, piwheels, crates.io, GitHub tag alignment, GitHub Actions, and OpenSSF Scorecard are green. This follows supply-chain guidance that provenance, public project health, trusted publishing, attestations, SBOMs, and checksums should be release artifacts rather than prose-only claims. The separate tracerazor-trice release-evidence packet binds the wheel, source distribution, CLI binary, installability card, proof cards, paper artifacts, evidence bundles, SHA-256 checksums, CycloneDX-style Python and Cargo SBOMs, and an in-toto/SLSA-shaped provenance statement. The GitHub release workflow then generates release-evidence sidecars and hosted artifact attestations for release assets produced by Actions. A separate tracerazor-trice integrity card binds offline doctor output, proof-card verifiers, release evidence, the crates publish card, installability card, research card, the paper manifest, schemas, and workflow hooks as one top-level proof graph.

Clean Wheel Installability

The generated installability card reports level `python_trice_install_ready` and score 90/100. It creates a clean virtual environment, installs the built wheel with `pip install --no-deps`, imports packaged schemas and public APIs, and runs the installed tracerazor-trice console script. The full tracerazor Rust CLI check remains separate so generic wheels do not overclaim platform-bundled binary readiness.

Research Card

The generated research card reports level `research_basis_locked` and score 100/100. It binds 165 ledgered sources, 165 unique URLs, category coverage, row hashes, and the source ledger hash. This proves the paper basis is reviewable, not that the held-out S-tier outcome gate has passed.

Crates Publish Card

The generated crates card reports level `publish_plan_locked` and score 80/100. It binds Cargo manifests, local dependency order, crates.io registry status, and README cargo-install honesty. `cargo_install_claim_allowed` is false, so Rust install wording remains blocked until the final CLI crate is public.

Replicated Suite Evidence

The public suite artifact fix-offby-one-suite contains 3 live replicate runs across 1 task cluster(s). Mean input-token savings is 76.6% with clustered-by-task 95% CI 76.6% to 76.6%; pass regressions 0. S-tier gate: not passed.

Portable Artifact Bundle

The public bundle trice_suite_evidence.trice.zip contains 35 hashed entries, root manifest trice_suite_evidence_manifest.json, and root result trice_suite_results.json. Verification replays aggregate and child-manifest checks after safe extraction.

Broad Smoke Evidence

The broad-smoke suite bundled-live-six-task-suite contains 6 live run(s) across 6 task cluster(s). Mean input-token savings is 81.5% with clustered-by-task 95% CI 79.0% to 83.5%; pass regressions 0. S-tier gate: not passed.

The broad-smoke bundle trice_broad_smoke_evidence.trice.zip contains 77 hashed entries.

Remote-Git Smoke Evidence

The remote-git smoke suite trice-remote-smoke contains 1 live run(s) across 1 locked public Git task cluster(s). Mean input-token savings is 83.2% with clustered-by-task 95% CI 83.2% to 83.2%; pass regressions 0. S-tier gate: not passed.

The remote-git smoke bundle trice_remote_smoke_evidence.trice.zip contains 17 hashed entries.

The remote-git smoke claim card reports level smoke, claim allowed no, and determinism score 100/100. It is a verified smoke artifact, not an S-tier claim.

Source Provenance

The suite source manifest records repo tree digest 604addf4070c... over 3 files and intervention provenance before execution. JSON patch tasks record patch SHA-256; command adapter tasks record command argv and timeout; adapter-profile tasks record profile SHA-256. Suite tasks may use either local paths or locked Git sources with URL, revision, optional subdirectory, and resolved commit recorded.

Product Contract

A TRICE result is accepted only when the public library emits a stable manifest, the live verifier passes, and replay is used only as preflight evidence. Generic users provide LiveTask plus a deterministic RepairAdapter; the bundled JSON patch and command repair adapters refuse test edits by default. Command adapters may be packaged as trice-adapter-profile/v1 files, and every live condition emits a trice-run-receipt/v1 artifact with adapter envelope, command hash, before/after workspace fingerprints, changed files, output hashes, and optional agent-reported token accounting. Receipts have a shipped JSON Schema and are validated during manifest and bundle verification. Identical real-run smoke executions must reproduce result and artifact hashes. Multi-repository evaluation uses trice-suite/v1 manifests, local or locked Git sources, verify-suite for aggregate plus child-manifest verification, and deterministic .trice.zip evidence bundles for handoff. The public CLI is tracerazor-trice and mirrors python -m tracerazor.trice.

Deterministic Claim Gate

Local smoke gate passed: yes. Broad claim allowed: no. Rationale: local deterministic smoke passed; broad claim still requires held-out provider runs with repeated trials and clustered confidence intervals

Selected References

- [agentsmatter2024] AI Agents That Matter. <https://arxiv.org/html/2407.01502v1>
- [acmartifact] ACM Artifact Review and Badging. <https://www.acm.org/publications/policies/artifact-review-badging>
- [mohammadi2025agentsurvey] Evaluation and Benchmarking of LLM Agents: A Survey. <https://arxiv.org/html/2507.21504v1>
- [complexmcp2026] ComplexMCP: Evaluation of LLM Agents in Dynamic Tool Sandboxes. <https://arxiv.org/html/2605.10787v2>
- [reasonbench2025] ReasonBENCH: Benchmarking the Instability of LLM Reasoning. <https://arxiv.org/html/2512.07795v2>
- [dfah2026] Replayable Financial Agents: A Determinism-Faithfulness Assurance Harness. <https://arxiv.org/html/2601.15322v1>
- [terminalagents2026] Building AI Coding Agents for the Terminal. <https://arxiv.org/html/2603.05344v1>
- [uncertainty2024] Towards Reproducible LLM Evaluation. <https://arxiv.org/html/2410.03492v2>
- [externalization2026] Externalization in LLM Agents. <https://arxiv.org/html/2604.08224v1>
- [sweuniverse2026] SWE-Universe: Scale Real-World Verifiable Environments. <https://arxiv.org/html/2602.02361v1>
- [swebench2023] SWE-bench: Can Language Models Resolve Real-World GitHub Issues?. <https://arxiv.org/abs/2310.06770>
- [sweagent2024] SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. <https://arxiv.org/abs/2405.15793>
- [acon2025] Acon: Optimizing Context Compression for Long-Horizon LLM Agents. <https://arxiv.org/html/2510.00615v1>
- [personalizedagents2026] Learning Personalized Agents from Human Feedback. <https://arxiv.org/html/2602.16173v1>
- [nlharness2026] Natural-Language Agent Harnesses. <https://arxiv.org/html/2603.25723v1>
- [swtbench2024] SWT-Bench: Testing and Validating Real-World Software Fixes. <https://arxiv.org/html/2406.12952v3>
- [swebenchcl2025] SWE-Bench-CL: Continual Learning for Software Engineering Agents. <https://arxiv.org/abs/2507.00014>
- [rigorousagentbench2025] Establishing Best Practices for Building Rigorous Agentic Benchmarks. <https://arxiv.org/html/2507.02825v5>
- [sparkllmeval2026] Spark-LLM-Eval: A Distributed Framework for Statistically Rigorous LLM Evaluation. <https://arxiv.org/html/2603.28769v1>
- [agentloganalysis2026] Log Analysis is Necessary for Credible Evaluation of AI Agents. <https://arxiv.org/html/2605.08545v1>
- [topline2026] Topline Benchmark Performance. <https://arxiv.org/html/2605.28065v1>
- [setupbench2025] SetupBench: Assessing Software Engineering Agents' Ability to Bootstrap Development Environments. <https://arxiv.org/abs/2507.09063>
- [pairedbca2025] A Paired Bootstrap Protocol for Evaluating Small Improvements. <https://arxiv.org/pdf/2511.19794>
- [llmlingua2023] LLMingua: Compressing Prompts for Accelerated Inference. <https://arxiv.org/abs/2310.05736>
- [longllmlingua2023] LongLLMingua: Accelerating and Enhancing LLMs in Long Context Scenarios. <https://arxiv.org/abs/2310.06839>
- [promptcache2023] Prompt Cache: Modular Attention Reuse for Low-Latency Inference. <https://arxiv.org/abs/2311.04934>
- [kvflow2025] KVFlow: Efficient Serving for LLM Agent Workflows. <https://arxiv.org/abs/2507.07400>
- [agentdiet2025] AgentDiet: Efficient Context Pruning for LLM Agents. <https://hf.co/papers/2509.23586>
- [slsa] Supply-chain Levels for Software Artifacts Specification. <https://slsa.dev/spec/>
- [openssfscorecard] OpenSSF Scorecard. <https://scorecard.dev/>
- [pypitrust] PyPI Trusted Publishing. <https://docs.pypi.org/trusted-publishers/>
- [pypiattest] PyPI Digital Attestations. <https://docs.pypi.org/attestations/>
- [intoto] in-toto Attestations. <https://in-toto.io/>
- [cyclonedx] CycloneDX Specification. <https://cyclonedx.org/specification/overview/>
- [mlperfolicies] MLPerf Policies and Submission Rules. <https://docs.mlcommons.org/inference/policies/>
- [pypasample] PyPA Sample Project. <https://github.com/pypa/sampleproject>
- [semver] Semantic Versioning 2.0.0. <https://semver.org/>
- [pyversion] Python Version Specifiers. <https://packaging.python.org/en/latest/specifications/version-specifiers/>

[jsonschema2020] JSON Schema Draft 2020-12. <https://json-schema.org/draft/2020-12>

[reprobuilds] Reproducible Builds. <https://reproducible-builds.org/>

[cargoPublishing] Cargo Publishing Reference. <https://doc.rust-lang.org/cargo/reference/publishing.html>

[pypapackaging] Python Packaging User Guide: Packaging Python Projects. <https://packaging.python.org/tutorials/packaging-projects/>

[pipinstall] pip install Command Reference. https://pip.pypa.io/en/stable/cli/pip_install/

[wheelpep] PEP 427: The Wheel Binary Package Format. <https://peps.python.org/pep-0427/>

[react2022] ReAct: Synergizing Reasoning and Acting in Language Models. <https://arxiv.org/abs/2210.03629>

[reflexion2023] Reflexion: Language Agents with Verbal Reinforcement Learning. <https://arxiv.org/abs/2303.11366>

[sweevo2025] SWE-EVO: Benchmarking Coding Agents in Long-Horizon Software Evolution. <https://arxiv.org/html/2512.18470v6>

[taubench2024] tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. <https://arxiv.org/abs/2406.12045>

[agentbench2023] AgentBench: Evaluating LLMs as Agents. <https://arxiv.org/abs/2308.03688>

[toolllm2023] ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. <https://arxiv.org/abs/2307.16789>

[webarena2023] WebArena: A Realistic Web Environment for Building Autonomous Agents. <https://arxiv.org/abs/2307.13854>

[voyager2023] Voyager: An Open-Ended Embodied Agent with Large Language Models. <https://arxiv.org/abs/2305.16291>

[longbench2023] LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding. <https://arxiv.org/abs/2308.14508>

[lostmiddle2023] Lost in the Middle: How Language Models Use Long Contexts. <https://arxiv.org/abs/2307.03172>

[memgpt2023] MemGPT: Towards LLMs as Operating Systems. <https://arxiv.org/abs/2310.08560>

[rag2020] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. <https://arxiv.org/abs/2005.11401>

[pagedattention2023] Efficient Memory Management for Large Language Model Serving with PagedAttention. <https://arxiv.org/abs/2309.06180>

[streamingllm2023] Efficient Streaming Language Models with Attention Sinks. <https://arxiv.org/abs/2309.17453>

[h2o2023] H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. <https://arxiv.org/abs/2306.14048>