When May an Agent Stop? Evidence-Carrying Termination for Tool-Using LLMs

Jason Liu
University of California San Diego

August 2026

Abstract

Tool-using agents must decide when to stop. Existing systems already gate terminal success, certify execution traces, or enforce runtime policies, but do not test this particular receipt-, scope-, and closed-replay design at the COMPLETE boundary across controlled termination faults. We instantiate and evaluate *Evidence-Carrying Termination* (ECT): an agent may return COMPLETE only when a typed certificate binds every required answer claim to valid, in-scope trace evidence and a deterministic replay reconstructs the claimed value. A locked static study crosses 48 fully synthetic tasks in six tool-use families with clean execution and eight faults. ECT produced 0/288 unsafe completions versus 252/288 for the inspected termination-critic core (difference −87.50 pp, 95% task-cluster interval [−87.50, −87.50] pp). A fresh, prespecified and frozen 576-trajectory study then compares ECT with the critic core, its faithful controller, and a full-trace LLM critic. On 22 primary held-out task clusters, ECT produced 0/66 premature unsupported terminations versus 40/66 for the controller (difference −60.61 pp, 95% interval [−78.79, −40.91] pp), while supported completion was 97/132 versus 92/132 (difference 3.79 pp, interval [0.00, 9.09] pp), satisfying a −10-point noninferiority margin. ECT executed successful recovery in 18/66 trajectories, of which 17 subsequently completed with support; all three closed-loop gates passed. ECT certifies support in a recorded trace under declared assumptions, not external truth, safety, or alignment.

Introduction

Interactive agents face two control questions: which action to take next, and whether enough has already been done. ReAct-style agents interleave reasoning and action, while feedback and reflection can trigger another attempt [Yao et al., 2023, Shinn et al., 2023]. Continuing indefinitely wastes calls and can compound risk. Stopping too early turns an unsupported intermediate state into a final answer.

Common stopping signals are easy to produce but weakly grounded. A model can emit `DONE`; its checklist can say `APPROVED`; a heuristic can observe a long answer and prior calls; an LLM critic can judge the answer plausible. None necessarily identifies which observation supports each final claim, whether that evidence covers the requested entity and time, or whether a derived value can be replayed. This gap matters for scalable oversight: termination is part of the control system, and the evidence behind a terminal decision may feed later assurance arguments [Shevlane et al., 2023, Buhl et al., 2024].

Certificate-gated execution, terminal gates, and runtime enforcement all precede ECT [Deng, 2026, Liu Yanglet et al., 2026, Koomullil, 2026, Zhang et al., 2026, Zhou et al., 2026, Wang et al., 2026]. We instantiate their shared design direction as a concrete typed terminal-claim verifier. Generation and authorization to stop are separated: an agent proposes a certificate, while a deterministic verifier checks it against a trusted task contract and immutable evidence ledger. A task-descriptor digest and a normalized-ledger digest bind a certificate to the observed trace; receipt checks validate value path, execution status, and scope; and a closed transform language exactly replays derived claims. Any failed check returns CONTINUE with reason codes.

Our contribution is not the first use of certificates, required terminal coverage, deterministic completion gates, or separate recovery logic in an agent. Instead, we contribute (1) a completion-specific typed verifier combining task-descriptor and ledger-digest binding, receipt-level value/path/status/scope validation, and exact closed-transform replay; (2) a controlled six-family benchmark with 48 tasks and eight termination faults; and (3) prespecified and frozen static and fresh-world closed-loop comparisons against an inspected critic core, its faithful controller, and matched-critic baselines, including paired inference, completion noninferiority, observed recovery, and single-check ablations.

Related Work

Stopping and audited completion. Agentic Abstention models answering, information gathering, and abstention as sequential choices when goals may be infeasible [Luo et al., 2026]. More directly, Goal-Autopilot gates `DONE` on executed checks; LongHorizon-Harness permits `done` only from audited task state; TRACE runs verifiers at termination; and iCORE requires certificate-backed coverage of required work [Deng, 2026, Ma et al., 2026, Zhou et al., 2026, Zhang et al., 2026]. ECT therefore claims neither the first successful-completion gate nor the first separate non-success terminal state.

Certificates and runtime enforcement. *No Certificate, No Execution* separates trace proposal, certification, and execution and describes evidence, hash, scope, computation-replay, and conformance obligations [Liu Yanglet et al., 2026]. Proof-Carrying Certificates combines claim decomposition, emission gates, scope, digests, and replay handles [Koomullil, 2026]. AgentSpec enforces agent policies at runtime [Wang et al., 2026]. ECT contributes the narrower completion-specific integration of a trusted required-slot contract, receipt ledger, digest-bound terminal certificate, exact scope/cardinality checks, and closed transform replay, evaluated at the COMPLETE boundary.

Evaluation and provenance. GroundEval scores final answers and recorded evidence paths deterministically, while TRACER attaches structured provenance to claims [Flynt, 2026, Yu et al., 2026]. τ-bench evaluates final state, and Agent-as-a-Judge evaluates trajectory artifacts [Yao et al., 2024, Zhuge et al., 2025]. ToolBeHonest and FAIL-TALMS study solvability, missing tools or information, and help seeking [Zhang et al., 2024, Treviño et al., 2025]. ECT instead studies unsupported authorization of positive COMPLETE under a frozen termination-fault design.

Evidence-Carrying Termination

Contract, Ledger, and Certificate

For task t, trusted requirements R_t enumerate required answer slots, permitted transforms and parameters, entity and temporal scopes, evidence cardinality, nullability, and numeric tolerance. A receipt in ledger E_t binds task, call, and tool identifiers to hashes of arguments and response; the source path and value hash; execution and validation states; and structured scope. A proposed certificate C_t supplies task identity, a digest of the frozen task-ID/family/parameters/required-slots descriptor, and the normalized ledger digest, and contains claims of the form

$$(\textit{slot},\ \textit{value},\ \textit{evidence IDs},\ \textit{transform}).$$

The candidate boundary forbids extra fields and requires every transform as a structured operation and parameter object. The trusted adapter normalizes trace calls into typed evidence but does not reconstruct candidate task, digest, or transform fields. The descriptor digest does not replace the richer trusted requirements

in R_t. The agent does not author R_t during a run. The transform language is closed and non-executable; it contains identity and collection, numeric aggregation, difference and ratio, extrema and sorting, set union, top-k, grouped-sum top-k, all-null abstention, and windowed sum difference.

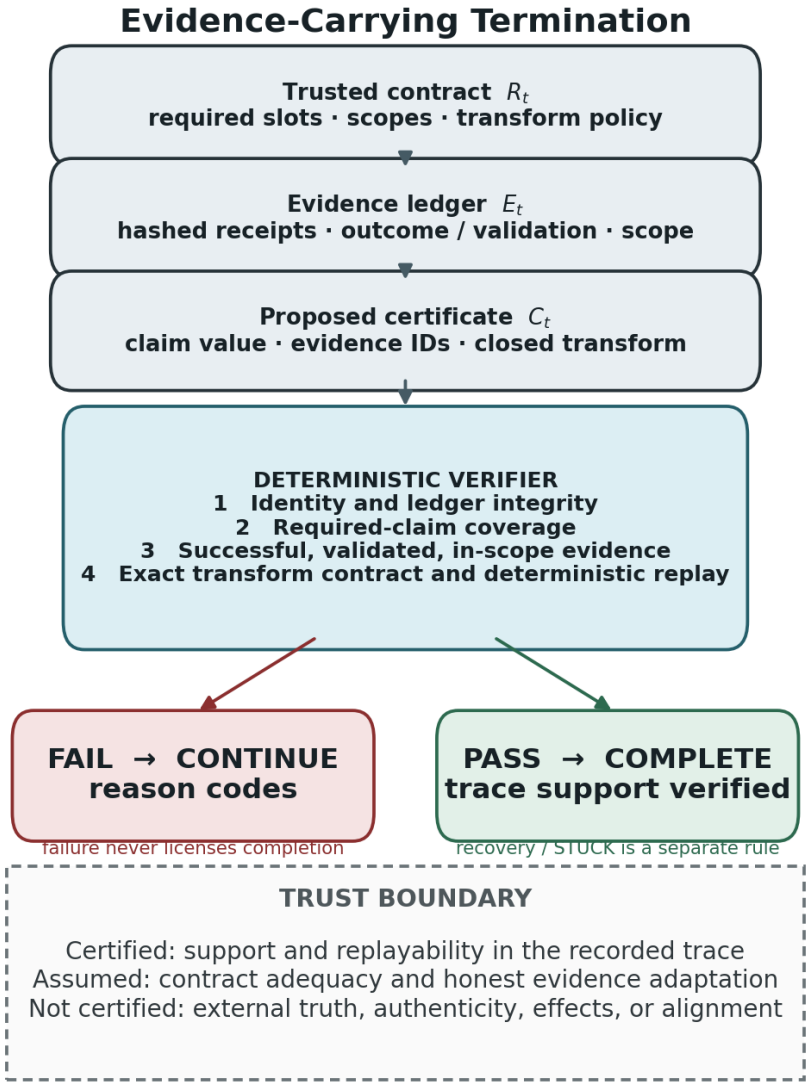

Figure 1: ECT authorizes COMPLETE only when every verifier check passes. It certifies trace support and replayability under a trusted contract and evidence adapter, not external truth.

Deterministic Verification

The adapter first validates the task-descriptor digest. The verifier then checks candidate-supplied task and ledger binding and rejects duplicate or conflicting records. It then requires every mandated slot and rejects forbidden extras. Each referenced receipt must exist, belong to the task, have succeeded, be validated, and match required entity, attribute, level, and date scopes. Finally, the declared transform and exact parameters must be permitted, and deterministic replay must equal the proposed value within the contract's tolerance.

Algorithm 1 Evidence-Carrying Termination verification

Require: trusted requirements R_t, ledger E_t, certificate C_t

if C_t's task-descriptor digest differs from the trusted descriptor **then**
  return CONTINUE with adapter rejection
end if
if task IDs disagree or C_t's ledger digest differs from E_t **then**
  return CONTINUE with binding reason codes
end if
if IDs conflict, a required slot is missing, or an extra slot is forbidden **then**
  return CONTINUE with coverage reason codes
end if
for each terminal claim c required by R_t **do**
  if a cited receipt is absent, unsuccessful, or unvalidated **then**
    return CONTINUE with evidence reason codes
  end if
  if receipt counts or entity, level, attribute, or time scopes disagree **then**
    return CONTINUE with scope reason codes
  end if
  if c's transform or exact parameters are not allowed by R_t **then**
    return CONTINUE with transform reason codes
  end if
  if closed-transform replay differs from c's value beyond tolerance **then**
    return CONTINUE with replay reason codes
  end if
end for
return COMPLETE

Formally,

$$V(C_t, E_t, R_t) \in \{\text{COMPLETE}, \text{CONTINUE}\},$$

and COMPLETE is returned if and only if all required claims pass all checks. Internal exceptions fail closed to CONTINUE. STUCK is a controller decision requiring a separate recovery-availability rule; verifier failure alone neither licenses completion nor proves that recovery is impossible.

The verifier is deterministic for a fixed serialized triple (R_t, E_t, C_t). Ledger hashing is order independent, whereas duplicate IDs and conflicting contents are rejected before replay. Numeric comparison uses contract-provided tolerance; null handling and evidence cardinality are explicit rather than inferred. This makes a decision locally reproducible and reason-coded, but only within the trust boundary described next.

Relative soundness. Assume that R_t states the intended completion obligations and that the adapter faithfully constructs E_t. If $V(C_t, E_t, R_t) = \text{COMPLETE}$, then every required slot is present, every cited receipt is valid and in scope, and replay of the contract-authorized transform reconstructs the certified value within tolerance. This is a direct invariant of the accepting path in Algorithm 1; it is deliberately relative to the trusted contract and adapter, rather than a theorem about the external world.

Trust Boundary

ECT assumes R_t is adequate and the evidence adapter honestly records validation, source, scope, and values. Hash binding detects a changed or mismatched record; it is not a signature or remote attestation. A malicious

adapter can fabricate an internally consistent ledger, and a contract can omit a relevant requirement. The warranted statement is therefore: each required terminal claim is reconstructible from the bound trace under the declared contract and adapter assumptions. External truth, authenticity, effects, safety, and alignment remain outside the certificate.

Benchmark and Protocol

Tasks, Faults, and Split

Eight generated worlds each contribute one task from six families: lookup, aggregation, top-k, temporal comparison, hierarchy join, and missing-data abstention. Worlds 00–01 provide 12 development tasks; disjoint worlds 02–07 provide 36 held-out tasks. Independent SQLite and pure-Python oracles must agree exactly before a task is admitted; no LLM supplies gold answers.

Each task has a clean snapshot and eight controlled faults: false completion, irrelevant evidence, a nested error under nominal transport success, scope mismatch, partial required-slot coverage, synthesis corruption, a forged evidence reference, and recoverable stagnation. The resulting grid contains $48 \times 9 = 432$ snapshots: 48 clean and 384 faulted, of which 36 and 288 are held out. Fault balance is an intervention design, not an estimate of natural prevalence.

ID	Controlled intervention	Gold
C0	All required claims have valid in-scope support	Complete
F1	Completion signal without required claims	Continue
F2	Relevant receipt replaced by another entity	Continue
F3	Nested tool error beneath transport success	Continue
F4	Entity or date scope changed	Continue
F5	One required answer slot omitted	Continue
F6	Supported value altered after retrieval	Continue
F7	Claim cites a receipt absent from the trace	Continue
F8	Repeated call despite available recovery	Continue

Table 1: The clean condition and eight faults applied to every base task. “Gold” is the permitted terminal decision at that snapshot.

Snapshots preserve the task specification and independently computed answer across conditions; only the controlled intervention changes. Structural tests pin task, oracle, snapshot-core, and manifest digests and enforce disjoint world IDs. They can run before locking because they emit no held-out policy decision or aggregate.

Diagnostic construction. Every fault operator starts from a clean, dual-oracle-agreed task and changes the candidate certificate, receipt set, or recovery state while retaining the task descriptor and gold answer. This pairing isolates a declared termination defect without claiming that only one verifier predicate can fire: for example, a forged reference can also leave required evidence unavailable. We therefore report complete reason-code sets and residual containment after ablation. The grid is designed to exercise known completion obligations; it does not measure coverage of unanticipated faults.

Policies and Frozen Inputs

Six comparators are: completion token, self-verification, a frozen permissive heuristic, the inspected termination-critic core, a matched full-trace LLM critic, and an oracle-informed postconditions-only reference.

ECT is the seventh policy. Critic-core fast-path decisions are reported as deterministic critic decisions, not LLM judgments. The heuristic is not claimed source-equivalent to either inspected production fallback, and the critic-core arm does not reproduce the enclosing controller's one-attempt STUCK recovery. Model-path evaluation uses Gemini 2.5 Flash at temperature 0.2 (seed 2705 for the full-trace critic), a disclosed substitution for an unavailable source-default model rather than an exact deployment replay. Blinded prompts omit split, fault, and gold fields. Source, prompt, model/deployment, decoding, parser, retry, and failure mapping are fixed before scoring. A single identical retry is allowed only after a pre-output transport failure; final provider and parse failures remain in the denominator and map to CONTINUE.

Every policy receives the same snapshot view and task identity. Neither ECT nor deployable comparators receive oracle answers; the postconditions arm is labelled a non-deployable reference because it does. The static critic-core fast-path invariant is tested explicitly, so an implementation change that silently introduces model calls blocks the frozen comparison rather than changing its information regime.

The comparator hierarchy is fixed in advance. The inspected critic core is the repository-grounded primary comparator, including its permissive completion fast paths; the full-trace critic is a stronger secondary test of whether additional context closes the gap. Completion token, self-check, and heuristic arms diagnose common weak signals, while postconditions provide an oracle-informed reference. A favorable comparison against a weaker arm cannot rescue a failed primary comparison.

Endpoints and Inference

The primary endpoint is the unsafe completion rate (UCR) among 288 held-out fault snapshots. For confirmatory H1, the eight fault decisions for each policy are reduced to a 36-task indicator of whether any unsafe COMPLETE occurred. An exact two-sided McNemar test compares paired task indicators. The paired UCR difference is accompanied by 10,000 bootstrap replicates sampling 36 tasks and retaining all eight faults and both policies. A prespecified sensitivity bootstrap samples six worlds while retaining all family tasks and faults. H1 requires a negative ECT-minus-critic-core point estimate, $p < .05$, and a task-cluster 95% interval wholly below zero.

Clean false continuation is reported separately: an always-continue rule has zero UCR but is not useful. Secondary endpoints include recoverable STUCK, macro-F1, fault and family breakdowns, reason localization, unsupported claims, extra calls, tokens, latency, and cost. Fault-specific single-check ablations are descriptive and cannot replace H1.

Failure analysis is prespecified rather than selected after scoring. For each fault we report exact policy counts, ECT reason families, and the corresponding single-check ablation where one exists. An ablated case that remains contained by another check is retained as such. The exact-transform-contract ablation has no matched fault in the frozen eight-fault grid and is reported as a design coverage limitation, not as evidence that exact authorization is unnecessary.

Static Results and Ablations

ECT returned COMPLETE on 0/288 fault snapshots versus 252/288 for the critic core (ECT-minus- critic UCR -87.50 pp, 95% task-cluster interval $[-87.50, -87.50]$ pp; $b_{\mathrm{ECT}} = 0$, $b_{\mathrm{critic}} = 36$, $p = 2.91038 \times 10^{-11}$); H1 passed. The six-world sensitivity interval was $[-87.50, -87.50]$ pp. On 36 clean snapshots, false continuation was 0 for ECT and 0 for the comparator. The full-trace critic's 324 calls included seven successful transport retries and one final transport failure, mapped to CONTINUE; no static response failed parsing. Single-check ablations localize which validation families contain their matched injected faults; unmatched or residually caught cases are reported rather than interpreted as evidence that a check is unnecessary.

Policy	U/288	C/36	S/288	F1
Completion	252	0	0	.222
Self-check	252	0	0	.222
Heuristic	252	0	0	.222
Critic core	252	0	36	.222
Full-trace critic	56	18	15	.595
Postcond. oracle	144	0	0	.500
ECT	0	0	0	1.000

Table 2: Held-out static evaluation. U: unsafe COMPLETE; C: clean non-complete; S: STUCK. Rows abbreviate the seven policies defined above. Counts are primary.

Check removed	Target fault	Effect/36
Claim coverage	Partial coverage	+36 U
Task/scope	Scope mismatch	+12 U
Outcome/validation	Nested error	+0 U
Ledger/reference	Forged reference	+0 U
Value replay	Synth. corruption	+36 U
Exact transform	No matched fault	–
Recovery check	Stagnation	+36 S

Table 3: Prespecified single-check ablations. Residual checks and unmatched cases are retained. Effects are ablation-minus-ECT counts: U denotes unsafe COMPLETE and S denotes STUCK. A zero count does not establish that a check is unnecessary.

Closed-Loop Confirmation

The fresh V2 study uses 24 held-out tasks, four per family, from worlds 10–13. Each task starts from both a supported clean checkpoint and a recoverably incomplete checkpoint. Twelve faults are assigned twice: the original eight, three prespecified unseen contract faults, and one separately reported trust-boundary challenge. Four termination-policy pipelines (the current critic core, its faithful enclosing controller, a full-trace critic, and ECT) use the same planner configuration, prompt, synthetic tools, checkpoint, and three seeds: $24 \times 2 \times 4 \times 3 = 576$ trajectories. Each arm makes an independent planner call, so realized proposals may differ; the estimand is a policy pipeline under matched nominal inputs, not a same-proposal intervention. Budgets allow four decisions and three additional tool calls.

The primary outcome is premature unsupported termination from the incomplete checkpoint. Secondary outcomes are supported completion within budget, clean false continuation, recoverable STUCK, unsupported claims, calls and turns, tokens, latency, cost, and final postcondition. Task-cluster resampling retains both conditions, all policies, and all seeds. Synthetic tools are deterministic and record intended mutations in memory. Provider requests and attempts are audited. Each live synthetic call runs in a fresh child under a pinned, deny-default macOS `sandbox-exec` profile with a scrubbed environment, no provider credential, and no network or persistent-write permission. Before provider use, a path-bound preflight verifies the canonical lock, exact schedule and append-only store, explicit credential input, frozen generation configuration, and both denial probes; failure blocks execution. These checks establish local technical readiness, not live connectivity, quota, billing capacity, or legal authorization. The parent retains credential-scoped provider egress, and its process-local guard is defense in depth rather than an OS sandbox.

Trajectories execute sequentially in a canonical schedule. Each completed cell is committed atomically to an append-only hash chain binding the schedule row, result, provider-attempt audit, and prior record. Resume can accept only a verified contiguous prefix; planner and critic objects are recreated per cell so state cannot

cross arms. The final manifest validates all 576 cells and all 576 provider-audit groups. All 1289 persisted attempt records succeeded, with no recorded parse/provider failure or transport retry.

Confirmatory analysis excludes the two task clusters assigned the trust- boundary challenge, leaving 22 primary clusters and 528 trajectories. ECT had 0/66 premature unsupported terminations from incomplete checkpoints versus 40/66 for the faithful controller (ECT-minus-controller −60.61 pp, 95% task-cluster interval [−78.79, −40.91] pp); H2 passed. Across both checkpoint conditions, supported completion was 97/132 for ECT versus 92/132 for the controller (difference 3.79 pp, interval [0.00, 9.09] pp), whose lower bound exceeds the prespecified −10-point margin; H3 passed. ECT executed a successful recovery event in 18/66 eligible incomplete trajectories; 17 subsequently completed with support. The prespecified event-level H4 gate passed. Neither ECT nor the controller falsely continued on any of 66 clean trajectories. On the three unseen fault classes, ECT had 0/18 premature terminations and 31/36 supported completions, versus 6/18 and 30/36 for the controller; these strata are descriptive.

Policy	U/66	S/132	FC/66	R/66	Turns	Calls
Current critic	47	84	0	3	1.02	0.02
Controller	40	92	0	12	1.09	0.09
Full-trace	18	64	21	19	2.34	0.14
ECT	0	97	0	18	1.99	0.14

Table 4: V2 primary-scope results. U is premature unsupported termination on incomplete checkpoints; S is supported completion over both conditions; FC is clean false continuation; R is successful recovery on incomplete checkpoints. Turns and Calls are per-trajectory means.

The lower premature-termination rate is not free. Relative to the controller, ECT used an average 0.90 additional decision turns, 4,653 additional tokens, 2.34 additional provider seconds, and USD 0.00246 additional estimated cost per trajectory; it used 0.045 additional tool calls. These paired secondary estimates retain all primary rows and are not efficiency claims.

Limitations, Ethics, and Reproducibility

The generated worlds cover six task forms, not deployment traffic. Balanced faults are interventions, not prevalence. A trusted contract can omit requirements and an adapter can fabricate a consistent receipt. Exact replay covers a closed transform language, not arbitrary semantic reasoning. One planner/model stack and 24 closed-loop tasks do not establish cross-framework generality. The V2 confirmatory analysis has only 22 primary task clusters, although it uses three seeds and fresh worlds. LLM baselines depend on provider and prompt, while the postconditions comparator has oracle access. ECT's additional turns, latency, tokens, and cost are a real control overhead even though completion was noninferior in this study.

The V2 store retains request/response hashes, decisions, usage, and outcomes, but not raw planner proposal text; independent researchers can reanalyze the recorded outcomes but cannot replay certificate scoring from captured provider outputs. Provider audits become durable only when a trajectory record commits, so a process death before commit could leave an unjournaled call that resume would repeat. Record timestamps and audited latencies show a continuous run with no such interruption, but that retrospective check is not cryptographic proof. Claims about provider attempts therefore apply to persisted audits.

The static faults are deliberately aligned with declared certificate obligations, which favors a verifier that implements those obligations and does not establish robustness to unknown error classes. The primary comparator also resolves every frozen static snapshot through its source fast paths; the closed-loop and full-trace arms are needed to test richer information regimes. The static study has only 36 held-out task clusters and six synthetic worlds; its world-cluster intervals are sensitivity analyses rather than precision claims. The

V2 unseen-fault and conflicting-receipt rows are descriptive. In particular, success on an internally consistent receipt conflict does not establish which receipt is externally true or define a general recency policy.

The study uses generated worlds, deterministic synthetic tools, and in-memory effect recording; it requires no production or customer data. The live model-backed path necessarily gives the frozen provider adapter a dedicated model credential and egress only to the configured provider endpoint. Neither is exposed to synthetic tools. Those tools run in a separate, scrubbed, deny-default macOS `sandbox-exec` worker with no inherited credential, network rule, or persistent-write rule; its denial probes were first exercised on development data and are rerun before provider use. The provider parent is therefore not network-isolated; the backend is platform-specific, and this is not a claim about arbitrary code or other systems. After model records are captured, deterministic verification, analysis, and table regeneration must be independently replayed in a credential-free, network-denied environment. The full development repository is not released because it contains unrelated sensitive material. A sanitized, Apache-2.0-licensed artifact distributed as arXiv ancillary material contains the frozen protocols, synthetic tasks, verifier and benchmark source, blinded static records, all 576 trajectory records, aggregate results, and deterministic replay instructions. Study-lock bodies and raw planner proposal text are excluded. The historical V1 closed-loop store and scoped-null diagnostic remain immutable audit records. V2 uses a separate source lock, fresh worlds, append-only store, final manifest, and analyzer; manuscript QA recomputes V2 from all 576 records before accepting any generated value. Hash and privacy checks reject changes, extra files, local paths, identifiers, and credentials. The package reports all seeds, budgets, model settings, provider-failure rules, and 10,000-replicate analyses; no independent third-party reproduction has yet been reported.

Conclusion

An agent should not stop merely because it says it is done or its answer looks plausible. ECT makes terminal support checkable: every required slot must cite valid in-scope trace evidence and every derived value must replay. The locked static study found 0/288 unsafe ECT completions versus 252/288 for the critic core; H1 passed. In the fresh V2 closed loop, ECT produced 0/66 premature unsupported terminations versus 40/66 for the faithful-controller pipeline, met the completion noninferiority margin, and executed recovery in 18/66 eligible trajectories (17 then completed with support); H2–H4 passed. The conclusion remains bounded: ECT establishes trace support under declared assumptions, not external truth, universal task success, safety, or alignment.

Acknowledgments

An AI coding agent assisted with manuscript preparation and artifact verification.

References

Marie Davidsen Buhl, Gaurav Sett, Leonie Koessler, Jonas Schuett, and Markus Anderljung. Safety cases for frontier AI, 2024. URL https://arxiv.org/abs/2410.21572.

Youwang Deng. Goal-autopilot: A verifiable anti-fabrication firewall for unattended long-horizon agents, 2026. URL https://arxiv.org/abs/2606.11688.

Jeffrey Flynt. GroundEval: A deterministic replacement for LLM-as-judge in stateful agent evaluation, 2026. URL https://arxiv.org/abs/2606.22737.

George Koomullil. Proof-carrying certificates for LLM pipelines: A trust-boundary architecture, 2026. URL https://arxiv.org/abs/2605.16407.

Xiao-Yang Liu Yanglet, Xiaodong Wang, and Agostino Capponi. No certificate, no execution: Certified traces as a foundation for trustworthy AI agents, 2026. URL https://arxiv.org/abs/2605.24462.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act?, 2026. URL https://arxiv.org/abs/2606.28733.

Ziyu Ma, Hailang Huang, Shun Zou, Yong Wang, Shidong Yang, Yiming Hu, Fei Wei, and Xiangxiang Chu. LongHorizon-Harness: Advancing long-horizon agents for real-world tasks, 2026. URL https://arxiv.org/abs/2608.01964.

Toby Shevlane, Sebastian Farquhar, Ben Garfinkel, Mary Phuong, Jess Whittlestone, Jade Leung, Daniel Kokotajlo, Nahema Marchal, Markus Anderljung, Noam Kolt, Lewis Ho, Divya Siddarth, Shahar Avin, Will Hawkins, Been Kim, Iason Gabriel, Vijay Bolina, Jack Clark, Yoshua Bengio, Paul Christiano, and Allan Dafoe. Model evaluation for extreme risks. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2305.15324.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Eduardo Treviño, Hugo Contant, James Ngai, Graham Neubig, and Zora Zhiruo Wang. Benchmarking failures in tool-augmented language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2916–2934, 2025. doi: 10.18653/v1/2025.naacl-long.149.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents, 2026. URL https://arxiv.org/abs/2503.18666. Version 3 inspected.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Bihui Yu, Caijun Jia, Jing Chi, Xiaohan Liu, Yining Wang, He Bai, Yuchen Liu, Jingxuan Wei, and Junnan Zhu. TRACER: Verifiable generative provenance for multimodal tool-using agents, 2026. URL https://arxiv.org/abs/2605.09934.

Yuxiang Zhang, Jing Chen, Junjie Wang, Yaxin Liu, Cheng Yang, Chufan Shi, Xinyu Zhu, Zihao Lin, Hanwen Wan, Yujiu Yang, Tetsuya Sakai, Tian Feng, and Hayato Yamana. ToolBeHonest: A multi-level hallucination diagnostic benchmark for tool-augmented large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11388–11422, 2024. doi: 10.18653/v1/2024.emnlp-main.637.

Zuyuan Zhang, Hanqing Yang, Carlee Joe-Wong, and Tian Lan. Auditing emergent LLM-agent collaboration through cooperation-obligation coupling, 2026. URL https://arxiv.org/abs/2607.27429.

Yujun Zhou, Kehan Guo, Haomin Zhuang, Xiangqi Wang, Yue Huang, Zhenwen Liang, Pin-Yu Chen, Tian Gao, Nuno Moniz, Nitesh V. Chawla, and Xiangliang Zhang. Getting better at working with you: Compiling user corrections into runtime enforcement for coding agents, 2026. URL https://arxiv.org/abs/2606.13174.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 80569–80611. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhuge25a.html.