When Can You Delete an LLM Call? Correction-Grounded Evaluation of Deterministic Offloading

Tarun Upaday · working paper v0.3, August 2026

Abstract

Production agent systems spend model calls on decisions that are rule application, not judgment. Replacing such a call with ordinary code is easy to propose and hard to trust. We present a validation protocol for call-level offloading and a case study on a live communications-automation system. The protocol replays candidate rules against decision traces and scores them against human-corrected outcomes rather than raw model outputs. It cuts validity at rule-authoring boundaries. It reports exact binomial error ceilings, with an explicit refusal to certify at small n. In the case study, six hand-authored routing rules covered 5.7% of email classification decisions with zero disagreements against corrected outcomes. On every covered decision the operator later corrected, the rules matched the corrected outcome. The original model output matched none of them. Within coverage, the model contradicted the operator's settled policy 23% of the time. This inverts the standard evaluation target: agreement with the model is the wrong objective when correction data exists. We also measure the offload ceiling. 72% of email decisions in this system end with no reply, no work item, and a filing action. That is an upper bound far above current rule coverage. We release the protocol as an evaluation recipe. We also report the instrumentation the audit trail needed before exact replay was possible.

1. Introduction

Agent pipelines route, gate, file, deduplicate, and verify. Much of this is deterministic given the same inputs. Each such decision made by a model costs tokens and latency. The cost recurs on every model family the system runs.

The harder problem is not writing the replacement rule. It is knowing the rule is safe to deploy.

Existing work approaches this from two directions. Trace-compilation systems mine recurring agent workflows into deterministic programs. Surrogate-routing systems train small classifiers on model outputs and gate deployment on agreement with the teacher. Both directions treat the deployed model's behavior as the target to reproduce.

Our case study shows why that target is wrong for systems with a human in the loop. Operators correct model decisions. Corrections are sparse, but where they exist they are ground truth. They systematically mark the places where the model disagrees with the operator's actual policy. A candidate that reproduces the model reproduces those errors. A candidate that encodes the corrected policy is penalized by an agreement metric on exactly the decisions where it is most valuable. Evaluation should therefore score candidates against post-correction outcomes, with raw model outputs as a fallback where no correction exists.

We contribute four things. First, a correction-grounded replay protocol for validating call-level deterministic offloads, with chronological validity cuts and exact small-sample error certification. Second, a case study on a production personal-automation system, including the negative results and the instrumentation gaps. Third, a measured example of model policy variance that deterministic enforcement removes. Fourth, an offload-ceiling estimate showing where the real savings sit.

2. Related work

Surrogate routing. TRACER trains lightweight classifiers on production LLM classification logs and gates deployment on agreement with the teacher. The shipped gate is stronger than the original paper's summary suggests: it selects thresholds on exact binomial lower bounds, checks them on a held-out slice, and refuses deployment when no threshold certifies. We verified this in the current source while contributing to the project.

Our protocol differs in three ways. Candidates are arbitrary code, not trained surrogates. The evaluation target is the corrected outcome, not teacher agreement. And certification is per predicted class, with a small-sample evidence floor and chronological validity cuts; a pooled gate can certify overall while a minority class runs below target, which we demonstrate in Section 4 and patched upstream.

Workflow compilation. TraceCompiler mines clusters of agent traces into mostly deterministic workflows and declines to compile flows with unobserved side effects. Agentc rewrites expensive calls at runtime. Both operate above the single call. FrugalGPT routes between models by cost; the offload candidate here is cheaper than any model.

Selective prediction. Learning-to-defer and selective classification study the coverage-risk frontier for models with a rejection option. Conformal risk control gives distribution-free guarantees for a chosen risk level. Our deterministic candidates have fixed coverage. The machinery therefore reduces to exact binomial certification on the covered region, plus temporal validity arguments those frameworks usually assume away.

3. Protocol

The developer supplies decision traces, a candidate implementation, and a field-level equivalence definition.

Resolve final outcomes. Traces record what the model decided. Corrections record what should have happened. Resolve each decision to its final post-correction state and mark corrected records as gold. Score candidates against finals; report agreement with raw model outputs separately. The gap between the two scores is the model's measured error rate inside coverage.

Replay with exact inputs. Candidates match on message attributes. If the audit trail does not persist those attributes, replay degrades to reconstruction and inherits its noise. In our case study the trail stored decisions richly but not sender domain or subject. Inputs were therefore adjudicated from decision free text, with a recorded basis per record; ambiguous records were excluded. The case study's equivalence definition is field-level: reply decision (must be none), work-item creation (must be empty), flag (must be false), and mailbox disposition class (archive matches archive-type actions, discard matches trash-type actions). Rules that retain model work count as partial offloads and are excluded from full-offload coverage. The first deployment change this protocol motivated was a schema change: persist the match inputs, and make the rule engine the single writer of its own verdict block.

Cut validity chronologically. Random splits leak. Rules are usually authored in response to observed traffic. Traffic before a rule's authoring date is therefore the only clean out-of-sample set. Record authoring timestamps and report pre-authoring performance separately.

Certify with exact bounds, or refuse. For a deterministic candidate, coverage is fixed. The question is the error rate on covered traffic. With x observed errors in n covered decisions, report the one-sided 95% Clopper-Pearson ceiling: the largest error rate p for which observing x or fewer errors in n has probability at least 5% (Clopper and Pearson, 1934). With zero errors this is roughly 3/n. A 5% ceiling needs 60 covered decisions; a 1% ceiling needs 300. Below the target, the correct output is a refusal: keep shadowing, with the n required and the expected date. Certification thresholds are the operator's risk budget, set per decision class.

Measure the ceiling. Count decisions whose final outcome is trivial: no reply, no work, no flag, a filing disposition. This outcome-shaped set is an upper bound on rule-coverable traffic. It locates the classes worth authoring rules for next. Expanding coverage toward the ceiling also accelerates certification, since n grows with matched traffic.

4. Case study

System. A personal communications-automation system classifies every inbound message across email and

four chat channels into five decision axes: reply, operational work, mailbox disposition, attachments, and conversation state. Classification runs in shadow mode on two frontier-model families. The operator maintains six deterministic routing rules for known notice classes. They cover expense-platform reminders the operator has permanently waived, routine finance-status notices, generic payment-status notices, card-activity notices, a weekly community digest, and airline remittance advices requiring body markers. Before this work, a skill instruction told the model to load the rules file and apply matching rules. The rules were policy executed stochastically.

Corpus. 631 unique classification decisions from 2026-08-13 11:09 to 2026-08-16 20:01 UTC, the system's entire shadow history (3.4 days). A decision is unique by stable message identity: channel, account, and provider message id. Repeat classifications of the same message deduplicate to the first record, then resolve to the last correction. 229 decisions are inbound email, the population the rules address. 66 decisions (10.5%) carry operator corrections.

Coverage and errors. The six rules matched 13 decisions: 5.7% of inbound email, 2.1% of all traffic. Twelve are full offloads. One retains model work: the digest rule files deterministically but keeps a read-and-summarize step. Against final outcomes the rules made zero errors. The pooled 95% ceiling is 20.6%, so the protocol's verdict is refusal: keep shadowing until n reaches 60. Six of the thirteen matches predate their rule's authoring timestamp and are clean out-of-sample evidence: all five card-activity matches (authored on the corpus's final day; matches span the prior two days) and the single digest match.

Gold labels invert the target. Three covered decisions were later corrected by the operator. On all three, the rule output equals the corrected outcome and the original model output does not. Two were card preauthorization notices the model held for review against the operator's standing file-and-archive policy. One was the weekly digest, which the model routed to trash against a policy of read, report, then archive. Within coverage, the model's error rate against settled policy was 3/13 (23%); the rules' was 0/13. An agreement-based gate would have counted the rules' three best decisions as failures.

Policy variance, observed. The model archived a $1,066 card preauthorization silently on one day. Two days later it flagged a $251 preauthorization from the same sender class as material. The distinction appears in no operator policy and is not even monotone in amount. The model invented a criterion and applied it inconsistently. Code cannot do this on either model family the system runs.

Boundary behavior. The rules correctly declined the dangerous near-misses. These were a message from the expense platform's domain delivering bank documents for verification, a human email confirming a $3,670 deposit in payment-adjacent language, and a wallet-provisioning security notice. Subject-pattern scoping and fail-open body-marker handling carried the safety load.

The ceiling. 165 of 229 email decisions (72%) are outcome-trivial. Current rules cover 13 of those 165. The uncovered set includes a uniform five-observation family of trade-fill receipts. At the observed rates (3.8 covered decisions per day now, 48 per day at full ceiling), and assuming traffic is stationary, the certified-5% date is twelve days out at current coverage and one day at full ceiling coverage.

Deployment. Both protocol-motivated changes shipped on 2026-08-16. The audit schema now persists sender domain, subject, and a versioned verdict block written only by a deterministic prefilter script. The prefilter decides matched traffic. The classifier records its own agreement or disagreement on every matched decision, and its decision controls only when a rule's declared non-mechanical exception applies. Disagreements and operator corrections accumulate as paired certification evidence. Model-call savings begin when certified classes skip model review entirely.

5. Discussion

For surrogate systems. Correction-grounded scoring drops into any teacher-agreement gate. Where corrections exist, they should override the teacher. Surrogate-teacher disagreements on corrected records should count for the surrogate, not against it. A calibration routine that silently picks minimum coverage on a ten-example class is answering a question the data cannot support; the honest output is a refusal with the n required. Both changes are concrete in TRACER's open issues on per-class thresholds and small calibration sets.

Cross-model stability. A rule enforced in code beneath both model families removes a variance term that prompt retuning per family cannot reach. It also survives vendor model refreshes unchanged.

Limitations. One system, 3.4 days, n=13 covered decisions: the certification machinery exists precisely because this is not enough evidence. The headline gold-label result rests on three corrections. The audit trail does not attribute decisions to model families, so all results are pooled; per-family comparisons are future work. Rule-match inputs were reconstructed by a single adjudicator rather than replayed exactly. The schema change removes this for future runs but cannot repair the past. Corrections exist only where the operator noticed an error. Model error inside coverage is therefore a lower bound, and uncorrected model errors in uncovered traffic are unmeasured. Exception patterns were screened against subject text only, an optimistic direction we flag rather than hide.

6. Conclusion

Deleting an LLM call is a claim about safety, not about capability. The evidence that supports it is already lying in production systems: decision traces, operator corrections, and rule files that models are being asked to execute by hand. The protocol here turns that evidence into either a certified replacement or an honest refusal.

References

Clopper, C. J., and Pearson, E. S. The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika, 1934. TRACER: trace-based adaptive cost-efficient routing for LLM classification. arXiv:2604.14531. TraceCompiler: skill-guided mining and compilation of LLM agent traces. arXiv:2608.02680. Chen, Zaharia, Zou. FrugalGPT: using LLM APIs more efficiently. 2023. Geifman, El-Yaniv. Selective classification for deep neural networks. NeurIPS 2017. Madras, Pitassi, Zemel. Predict responsibly: learning to defer. NeurIPS 2018. Angelopoulos, Bates. Conformal risk control. 2022. Agentc: JIT optimization runtime for multi-step LLM agent workloads. github.com/AveryClapp/Agentc.