METARULE: TRANSACTIONAL PROCEDURE LEARNING FOR CROSS-DOMAIN HARNESS EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents rely on harnesses to organize context, memory, tools, and output handling, yet repeated execution does not explain when a local trace should become durable behavior. Existing model-centric approaches can store memories, revise prompts, or search harness configurations, but they lack a transaction for deciding when execution evidence is closed enough to persist, limiting its scope, and keeping learning separate from evaluation. We present METARULE, a transactional procedure-learning mechanism for self-evolving harnesses. It converts answer-free execution traces into typed, scoped, lifecycle-managed programs: replayable checkpoints expose telemetry to an isolated diagnoser; bounded same-thread revision and typed closure establish whether a procedure completed; and paired admission, provenance, and delayed publication control what a successor harness may execute. The compiler dispatches only trigger- and capability-matched operations in the next frozen state, while official verification and read-only held-out evaluation preserve the measurement boundary. Across scientific and competition-mathematics benchmarks, the complete METARULE treatment improves held-out performance over native harness evolution, with the clearest gains in registered cross-domain and cross-type transfer and in recovery from native negative transfer. Matched frozen-state and trajectory-shadow analyses place these gains at the complete harness-treatment level and quantify uncertainty around persistent procedure transfer. METARULE provides a replayable, auditable substrate for studying how execution evidence changes an agent's future behavior.

1 INTRODUCTION

Language-model agents increasingly depend on harnesses that coordinate context, memory, tools, and output handling. As agents operate across a sequence of tasks, this execution layer—not only the model weights—determines what can be reused, inspected, and transferred. A local trace may reveal a useful repair, but promoting it changes the program governing every subsequent task; the system therefore needs a state transition that preserves evidence, closes the intended operation, and limits the scope of the resulting behavior.

Most existing approaches improve one episode at a time: reflection revises an answer, memory stores an experience, and harness search selects a better configuration. Cross-task adaptation creates a stricter requirement. A useful intervention must survive replay, remain scoped to the capability that produced it, and avoid turning an unverified local repair into a new source of negative transfer. The system must also distinguish evidence used to learn from information available during evaluation. This missing state-transition contract is the systems gap addressed here.

This motivates a transaction boundary for harness evolution. The boundary must close answer-free evidence before diagnosis, verify that a same-thread repair actually completed, test proposed behavior on matched tasks, and publish any durable change only to a successor frozen state. Without these controls, persistent adaptation is difficult to audit and its apparent gains are hard to separate from leakage, prompt drift, or accidental state mutation.

We introduce METARULE, a transactional procedure-learning mechanism for self-evolving harnesses. Replayable checkpoints expose closed telemetry to an isolated diagnoser; bounded same-thread revision and typed closure establish whether a procedure completed; and paired admission with provenance and delayed publication controls what a successor harness may execute. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

official verifier alone determines correctness and held-out evaluation remains read-only. We use *MetaRule treatment* for this complete integrated condition; an individual compiled artifact is a procedure or precautionary countercheck program.

Our contributions are:

- **Transactional procedure learning.** We cast persistent harness adaptation as an auditable transaction from answer-free execution evidence to scoped, replayable successor behavior.
- **Typed lifecycle and guarded transfer.** We introduce the MetaRule treatment, an integrated lifecycle for durable, scoped procedural behavior across tasks.
- **Cross-domain transfer and negative-transfer repair.** We show that the integrated treatment improves registered cross-domain and cross-type transfer, while matched analyses delimit the contribution attributable to persistent procedure transfer.

2 Related Work

Reasoning and self-improvement. Chain-of-thought, self-consistency, reflection, self-refinement, and tool-assisted critique improve individual reasoning episodes through traces, feedback, or search (Wei et al., 2022; Wang et al., 2023b; Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2023). STaR, Tree of Thoughts, and Graph of Thoughts turn successful traces or intermediate states into training signals or structured search (Zelikman et al., 2022; Yao et al., 2023a; Besta et al., 2024). Our focus is the missing cross-task publication contract: which closed evidence may become durable harness behavior.

Memory and evolving agents. Experience memories, reusable programs, persistent stores, and automated agent design support adaptation across tasks (Wang et al., 2023a; Zhao et al., 2023; Wang et al., 2024b; Park et al., 2023; Hu et al., 2024; Packer et al., 2023; Cemri et al., 2026; Huang et al., 2026). MetaRule complements these systems with answer-free diagnosis, typed closure, matched admission, and successor-state publication.

Tools, programs, and evaluation. ReAct, Toolformer, SWE-agent, and PAL expose tools or interpreters as part of agent execution (Yao et al., 2023b; Schick et al., 2023; Jimenez et al., 2024; Gao et al., 2023), while AgentBench, GAIA, WebArena, and τ-bench emphasize explicit environments and outcome protocols (Liu et al., 2024; Mialon et al., 2024; Zhou et al., 2024; Yao et al., 2024). We study the state transition that selects reusable behavior across executions.

Process supervision and program optimization. Process feedback and verifiers provide step-level or outcome-level supervision (Uesato et al., 2022; Lightman et al., 2023; Zheng et al., 2024; Yang et al., 2023b), while DSPy, OPRO, and TextGrad optimize prompts or textual pipelines (Khattab et al., 2023; Yang et al., 2023a; Yuksekgonul et al., 2024). Our focus is the auditable boundary between public execution evidence, scoped transfer, and official verification.

3 MetaRule: Transactional Procedure Learning

MetaRule is a transactional procedure-learning mechanism for self-evolving harnesses. It converts answer-free execution evidence into typed, guarded programs that can affect the next frozen harness state. The complete operator combines replayable checkpoints, isolated diagnosis, same-thread revision, typed closure, paired admission, and delayed publication. Figure 1 shows the lifecycle; interface definitions and algorithms are given in Appendix A–C.

Given a frozen state S_t and the ordered, closed batch $\overline{\mathcal{O}}_t$, MetaRule produces

$$S_{t+1} = \mathfrak{M}(S_t, \overline{\mathcal{O}}_t), \qquad S_t = (B_t, H_t, \mathcal{E}_{<t}, \mathcal{R}_t, C_t, \Pi), \tag{1}$$

where B_t is the train-only experience bank, H_t the base harness, $\mathcal{E}_{<t}$ sanitized procedural experience, $\mathcal{R}_t$ the admitted procedure set, C_t the controller, and Π the frozen execution policy. All tasks in an iteration read the same S_t; only the completed batch can produce S_{t+1}.

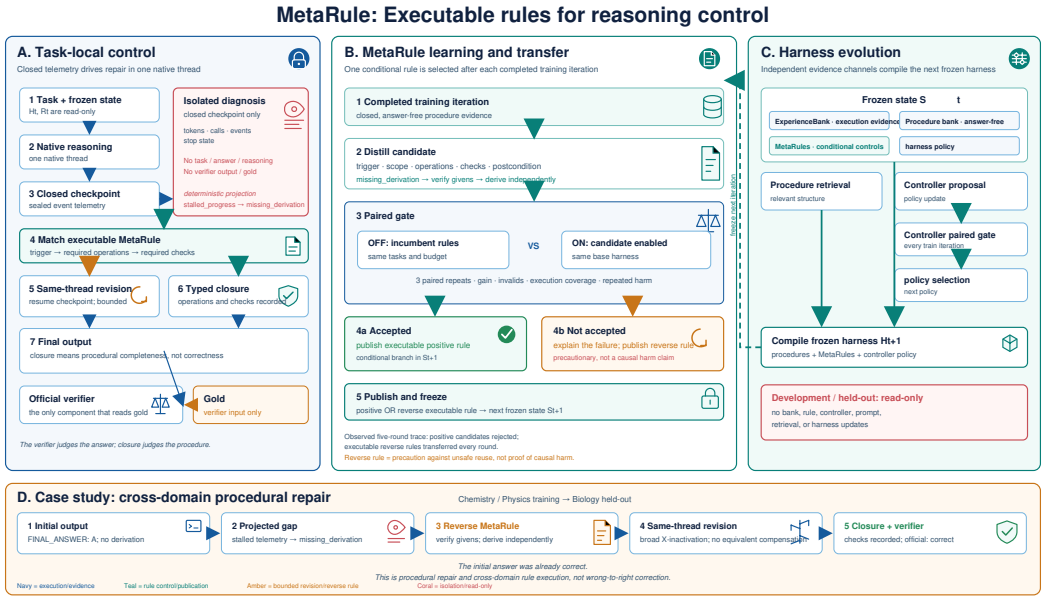


Figure 1: Overview of METARULE. Within a task, isolated diagnosis reads a closed checkpoint projection, a matching program may guide bounded same-thread revision, and typed closure decides whether the revised answer can replace the initial answer. At the batch barrier, controller and procedure candidates are evaluated by separate paired gates. Efficacy admission publishes a supported program; when support is insufficient, precautionary admission publishes a guarded countercheck program. Both outputs enter the next frozen state. Gold is read only by the official verifier and held-out evaluation is read-only.

3.1 PROBLEM FORMULATION

We study a stream of tasks $x \in \mathcal{X}$ processed by a harness whose state may persist across tasks but may change only at iteration barriers. For task x, the runtime emits a public answer a_x, an execution trace, and a closed checkpoint projection $\pi(Q_x)$; the private reference y_x is available only to the official verifier V. A procedure is a typed tuple $r = (\tau, \kappa, \omega, \sigma)$ containing a trigger class τ, capability scope κ, registered operations ω, and lifecycle status σ. The compiler may dispatch r only when its trigger and scope match the current task, and the verifier returns $V(a_x, y_x) \in \{\text{correct}, \text{incorrect}, \text{invalid}\}$ after execution.

The learning problem is to select a successor state that improves future verified utility while preserving evidence and information boundaries. In particular, the state update must use only the closed training batch and satisfy

$$S_{t+1} = \mathfrak{M}(S_t, \overline{\mathcal{O}}_t), \qquad \overline{\mathcal{O}}_t \subseteq \mathcal{X}_{\text{train}}, \qquad \mathcal{R}^{\text{new}}_{t+1} \subseteq \mathcal{R}^+ \cup \mathcal{R}^-, \tag{2}$$

where $\mathcal{R}^+$ contains efficacy-supported procedures and $\mathcal{R}^-$ contains non-causal precautionary counterchecks. Diagnosis, controller updates, and procedure admission cannot read y_x or raw verifier output. Development may select among frozen states, while held-out evaluation applies a sealed state and performs no update. The primary estimand is the complete-treatment difference in official-verifier performance; procedure-level causal effects are secondary mechanism estimands and are reported only with matched interventions.

3.2 ANSWER-FREE EVIDENCE AND SAME-THREAD REPAIR

The public task is compiled into an answer-free procedure graph G_x. A bounded retrieval packet contains only public structure, sanitized diagnosis/closure classes, and prior repair mechanisms; it excludes source answers, raw reasoning, gold labels, and raw verifier text. The compiler produces a task-local harness $\widehat{H}_{t,x} = \mathcal{C}(H_t, G_x, P_{t,x}, \mathcal{R}_t, C_t)$.

Execution first closes a native checkpoint $Q_{t,x}$ at a frozen event boundary. The isolated diagnoser receives only the deterministic projection $\pi(Q_{t,x})$ containing trigger counters, aggregate usage, completion status, and settled native status:

$$d_{t,x}^{\text{phase}} = \mathcal{D}(\pi(Q_{t,x})), \qquad \{a_{t,x}, r_{t,x}, y_x\} \notin \text{dom}(\pi(Q_{t,x})). \tag{3}$$

Here $a_{t,x}$, $r_{t,x}$, and y_x are the current answer, reasoning trace, and gold answer. These fields are omitted from the diagnosis request by construction. A deterministic adapter maps the typed phase class to a semantic trigger such as missing derivation, choice mapping, contradiction, or format error.

When revision is requested, the runtime resumes the original native thread and executes only the operations associated with the matching typed program. Closure requires the prescribed evidence kind, no new contradiction, and exact completion of the public check set:

$$\text{Close}(d) = \mathbf{1}[\text{resolved}]\mathbf{1}[\neg\text{newContradiction}]\mathbf{1}[e = e_d]\mathbf{1}[\widehat{K} = K_d]. \tag{4}$$

If closure fails and capacity remains, one bounded corrective revision is attempted. Only a verified closure may replace the initial answer. The official verifier then independently emits correct, incorrect, or invalid; operational failures remain distinct outcomes.

3.3 Native evolution and the procedure channel

The T1 treatment isolates native evolution: after each complete training batch it updates the ExperienceBank and answer-free procedure bank, and on its registered controller schedule it applies a paired gate to the proposed harness configuration. This stream has no typed diagnosis, closure, or executable MetaRule lifecycle. T1 is therefore a native-evolution baseline rather than a static prompt baseline.

MetaRule retains this transaction and adds a separately gated procedure stream. After the same closed batch, the controller proposes C_t' and the procedure learner proposes candidates r_t. The two updates are committed only at the barrier:

$$(C_{t+1}, \mathcal{R}_{t+1}) = \text{Commit}(\text{Gate}_{\text{ctrl}}(C_t, C_t'), \text{Gate}_{\text{rule}}(\mathcal{R}_t, r_t)). \tag{5}$$

Thus T1–MetaRule is an integrated treatment comparison: it measures the added procedure channel together with diagnosis, revision, closure, admission, and transfer, rather than a rule-only effect.

3.4 Paired admission and precautionary transfer

Both gates compare matched off/on arms on pre-reserved tasks and repetitions. For task-level differences $\bar{\delta}_i$ they compute

$$\widehat{\Delta} = N^{-1}\sum_i \bar{\delta}_i, \qquad \begin{aligned}\text{Accept} = \mathbf{1}[I = 0]\mathbf{1}[H = \varnothing]\mathbf{1}[\rho_{\text{exec}} = 1]\mathbf{1}[\rho_{\text{evid}} = 1] \\ \cdot\, \mathbf{1}[\widehat{\Delta} > 0]\mathbf{1}[L \geq -0.05].\end{aligned} \tag{6}$$

The conditions veto invalid regression, repeated task harm, incomplete execution/evidence coverage, and a materially negative lower bound. These are engineering selection contracts, not population-level significance tests.

If a candidate passes, it enters the next state as an efficacy-supported program. If evidence is insufficient, rejection analysis records the observed reason and the compiler may emit a precautionary reverse program

$$r^- = \mathcal{N}(r, A(r, g)) = \big(d, \text{forbid}(r), K^-, \text{post}(d), -\big). \tag{7}$$

The reverse program does not claim that the rejected procedure caused an error; it only forbids blind reuse and requires public counterchecks. A closed validator checks provenance, trigger scope, registry membership, exact agreement between operations and closure checks, and the absence of a causal claim. This is a separate precautionary proposition, not a bypass of efficacy evidence.

The mechanism requires every completed training batch to pass at least one typed transfer artifact, positive or reverse, into the successor state. If no supportable candidate exists, a conservative fallback reverse constraint is compiled from the most frequent actionable diagnosis (or the public format trigger). Dispatch remains conditional on trigger and capability scope, closure controls answer replacement, and repeated exposed failures can suspend a rule at the next barrier. Complete lifecycle and replay records are in Appendix D. Thus each completed iteration passes either an efficacy-supported procedure or a guarded precautionary countercheck to the next frozen state.

Table 1: Registered local datasets and transfer strata.

Benchmark	Train	Dev	Held-out	Model	Transfer split
GPQA-Diamond	125	18	55	GPT-5.5	36 same / 19 cross-domain
AIME2025	25	–	5	Haiku 4.5	not split
HMMT (25+26)	39	–	24	Haiku 4.5	12 same / 12 cross-type

3.5 Replay and evaluation boundary

The batch barrier requires one terminal official-verifier outcome per training task before learning begins. Recovery reuses completed outcomes and retries only missing or operationally failed outcomes under the frozen protocol. Development may select among frozen states but cannot update them. Held-out evaluation binds a sealed state and is read-only with respect to the bank, controller, prompts, retrieval policy, and rule set. Full algorithms, interface definitions, split details, and audit records are provided in the appendix.

4 Experiments

We evaluate harness evolution across scientific and competition-mathematics benchmarks. Each local result uses a state frozen after five training iterations and five held-out generations per task, scored by the official verifier. Missing, provider-failed, timed-out, verifier-failed, and invalid outcomes remain distinct from incorrect answers; Table 2 reports only complete aggregates.

4.1 Datasets

We evaluate three registered local benchmarks. GPQA-Diamond uses 125 training, 18 development, 36 same-type held-out, and 19 cross-domain held-out questions. AIME2025 uses 25 training and 5 held-out problems with no development split. HMMT (25+26) uses 39 training and 24 held-out problems, split into 12 same-type and 12 cross-type tasks. AIME has only five held-out problems, so a same/cross domain breakdown would not be statistically or substantively meaningful; we report its aggregate metrics descriptively.

4.2 Evaluation Setup and Baselines

We compare three nested treatments under frozen, read-only held-out evaluation. B0 is the unextended Native MemoHarness baseline. T1 is the native-evolution treatment: it adds the ExperienceBank, answer-free retrieval, and controller-generated harness configuration. MetaRule is the complete treatment: it adds typed diagnosis, same-thread revision, typed closure, paired admission, and delayed procedure transfer to T1. All tables and figures use MetaRule. All local treatments use five training iterations and five official verifier generations per held-out task with the same ordered split within a benchmark. The primary held-out sets contain 55 GPQA-Diamond, 5 AIME2025, and 24 HMMT (2025+2026) tasks. Split, model, settings, recovery rules, and audit checks are in Appendix B. We report official-verifier Avg@5 and Pass@5; complete four-metric and invalid counts are retained in the appendix for auditability.

Our estimands are ordered accordingly: (RQ1) does the complete MetaRule treatment improve registered transfer; (RQ2) does it repair negative transfer introduced by native evolution; and (RQ3) can frozen dispatch or cross-iteration persistence alone explain that gain? RQ1–RQ2 use task-level Avg@5 on the registered strata (12 HMMT cross-type tasks and 24 tasks overall); RQ3 uses matched off/on or full/shadow tasks with exact task-level sign-flip tests. These are complete-system and mechanism-boundary comparisons.

For a held-out set of N tasks, we define $\mathrm{Avg@5} = (5N)^{-1} \sum_{i=1}^{N} \sum_{j=1}^{5} v_{ij}$, where v_{ij} is the official-verifier correctness indicator. Thus the HMMT cross-type estimate uses 12 tasks and the overall estimate uses 24 tasks.

Leakage boundary. Training updates consume training tasks only. Development may select among already frozen states but cannot write the bank, controller, prompts, retrieval policy, or rules. Held-out questions, answers, and verifier outcomes are read-only and cannot enter any subsequent update. The isolated diagnoser additionally receives neither the current answer or reasoning nor the gold answer or verifier text. External baseline rows are descriptive references and are never used for model, split, or policy selection.

4.3 Mechanism Realization and Search Space

The implementation separates model-mediated diagnosis from deterministic procedure compilation. A task model first runs the native thread until the registered checkpoint boundary, producing an answer-free telemetry projection. An independent isolated-diagnosis call then classifies that projection using the frozen typed interface; in audited MetaRule runs this call is made by GPT-5.5 at medium effort with no tools, while the task-side model remains the benchmark model specified above. The diagnoser emits typed phase/failure codes and registered revision-directive identifiers, not an answer, verifier response, or free-form rule text. The native task thread resumes with the selected directive and emits a typed closure record containing revision, checks, and procedural completion fields. The official verifier is invoked only after this execution path and is not part of the diagnoser's input.

The remaining learning pipeline is deliberately constrained and reproducible. The deterministic projection records each training outcome as a tuple of task identity, diagnosis code, revision and closure status, reward, invalid status, capability set, and attempted/executed rule IDs. Distillation groups the latest evidence by diagnosis code and requires support on at least three distinct training tasks; reserved gate tasks are excluded from discovery evidence. Candidate rules are then assembled from a finite registry of trigger, variant, capability, and operation identifiers. For the mathematical answer path, the actionable trigger vocabulary contains five semantic failure codes and the operation registry contains five primitives (including premise verification, independent derivation, conflict resolution, criteria separation, and preservation of supported work). The compiler renders the selected operation list into the short host instruction inserted at task time; it does not ask an LLM to invent a new procedure string.

Paired off/on gates are therefore admission tests for coverage, execution mediation, harm, and closure, rather than standalone significance tests. A candidate that does not pass the efficacy conditions is converted by deterministic reverse distillation into a non-causal precautionary countercheck; when no candidate reaches the support threshold, the configured fallback-negative path preserves the same conservative contract. This distinction is important for interpretation: the reported T1–MetaRule comparison evaluates the integrated diagnosis–revision–closure–publication system, whereas the frozen dispatch and trajectory-shadow interventions test whether the compiled procedure channel contributes at the successor-state boundary. They do not treat the finite template compiler as an unconstrained language-program learner.

4.4 Main Results

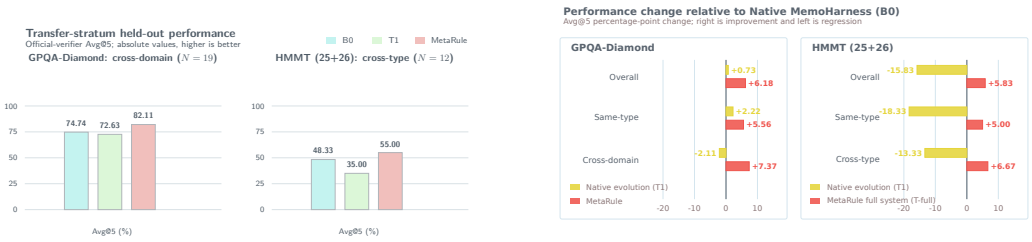
Table 2 gives the compact one-outcome comparison. On GPQA-Diamond, MetaRule reaches 90.91% Acc@1 on the registered 55-task held-out set. Across five generations per task, it attains 88.73% Avg@5 and 94.55% Cons@5, gains of 5.46 and 5.46 percentage points over T1, and produces no invalid generations. The complete repeated-generation metrics are reported in the appendix.

Single-outcome provenance. Local Acc@1 rows use the first registered held-out generation from the frozen five-iteration state; they are not Avg@5. The external rows are aggregated from an independently curated external audit set; its construction and exclusions are documented in the supplementary material. AIME 2025 and GPQA-Diamond use the five and 55 held-out inventory tasks, respectively; the main table preserves the 10-task HMMT 2025 and 14-task HMMT 2026 columns. These external one-shot audits are descriptive Acc@1 references and are not matched treatment arms.

The transfer-strata comparison uses the common Avg@5 estimator and contrasts same-type with cross-domain/cross-type strata for the two benchmarks with registered transfer partitions; the matched HMMT analysis below supplies the task-paired treatment comparison. Resource accounting and complete audit records are retained in the appendix.

Table 2: Single-outcome held-out accuracy (Acc@1, %). Local treatment rows use the first registered generation from each frozen five-iteration state; their benchmark-specific models are given in the setup. External rows are independent, context-only one-outcome-per-task audits with EpiGraph conditions excluded. HMMT is separated by source year; complete local five-sample metrics are retained in the appendix. Only completed, registered treatment rows are shown; unavailable treatment results remain '–'.

Agent / model / treatment	**GPQA-Diamond** $N = 55$	**AIME 2025** $N = 5$	**HMMT 2025** $N = 10$	**HMMT 2026** $N = 14$
Native MemoHarness (B0)	78.18	60.00	60.00	35.71
T1	85.45	60.00	30.00	28.57
MetaRule	**90.91**	60.00	60.00	42.86
Claude Code / Haiku 4.5	65.45	60.00	60.00	35.71
Claude Code / Opus 5	72.73	60.00	80.00	57.14
Claude Code / Sonnet 5	70.91	60.00	80.00	57.14
Codex / GPT-5.5	81.82	80.00	90.00	71.43
Codex / GPT-5.6 Sol	**90.91**	**100.00**	**100.00**	**100.00**
Codex / GPT-5.6 Terra	85.45	60.00	**100.00**	85.71



(a) Absolute held-out performance

(b) Change relative to Native MemoHarness (B0)

Figure 2: Two views of the same registered transfer-stratum outcomes. The left panel shows absolute official-verifier Avg@5 for B0, T1, and MetaRule; the right panel removes the B0 level and shows the corresponding percentage-point change for T1 and MetaRule. The GPQA-Diamond panel uses 19 cross-domain Biology tasks and the HMMT panel uses 12 cross-type Number Theory tasks.

The four metrics expose complementary behavior. On GPQA-Diamond, MetaRule raises Avg@5 from 83.27% (T1) to 88.73% and Cons@5 from 89.09% to 94.55%, while Pass@5 and Best@5 remain 96.36%. On HMMT, the same treatment raises Avg@5 from 35.83% to 57.50%, Pass@5 from 58.33% to 87.50%, Cons@5 from 41.67% to 66.67%, and Best@5 from 58.33% to 87.50%. AIME is reported descriptively because its registered treatment budgets are not matched; its MetaRule row reaches 72.00% Avg@5 and 80.00% on each task-level metric. The complete rows and invalid counts are retained in the appendix for auditability.

Cross-benchmark transfer and reliability. The same frozen-state estimator gives a consistent system-level pattern: GPQA cross-domain Avg@5 improves by 9.48 points over T1, HMMT cross-type by 20.00 points, and AIME increases from 68.00% to 72.00% descriptively on its five-task split. These comparisons use complete official-verifier outcomes; invalid outputs remain explicit (4/2/0 for B0/T1/MetaRule on GPQA and zero on the mathematics rows), so repeated-generation gains do not silently recode operational failures as incorrect.

5 Ablation Study

We decompose the treatment at three levels: native evolution (T1) versus B0, frozen dispatch of the final procedures, and trajectory-level transfer with or without persistence. These estimands are reported separately.

Table 3: Complete five-sample held-out metrics for the local treatment arms. Values are percentages; valid counts include all expected task–sample outcomes and invalid outcomes are shown in parentheses. These rows are separate from the one-outcome Acc@1 comparison in Table 2.

Benchmark	System	N	Valid (invalid)	Avg@5	Pass@5	Cons@5	Best@5
GPQA-Diamond	Native MemoHarness (B0)	55	271/275 (4)	82.55	**96.36**	87.27	**96.36**
GPQA-Diamond	Native evolution (T1)	55	273/275 (2)	83.27	**96.36**	89.09	**96.36**
GPQA-Diamond	MetaRule	55	275/275 (0)	**88.73**	**96.36**	**94.55**	**96.36**
AIME2025	Native MemoHarness (B0)	5	25/25 (0)	60.00	**80.00**	60.00	**80.00**
AIME2025	Native evolution (T1)	5	25/25 (0)	68.00	**80.00**	**80.00**	**80.00**
AIME2025	MetaRule	5	25/25 (0)	**72.00**	**80.00**	**80.00**	**80.00**
HMMT (25+26)	Native MemoHarness (B0)	24	120/120 (0)	51.67	79.17	50.00	79.17
HMMT (25+26)	Native evolution (T1)	24	120/120 (0)	35.83	58.33	41.67	58.33
HMMT (25+26)	MetaRule	24	120/120 (0)	**57.50**	**87.50**	**66.67**	**87.50**

Table 4: Task-paired HMMT nested ablation. Entries are Avg@5 percentage-point differences; parentheses contain the exact two-sided task-level sign-flip p-value. W/T/L counts tasks improved, tied, or degraded by the right-hand treatment.

Contrast	Overall	Same-type	Cross-type	Overall W/T/L
T1 − B0	−15.83 (.0029)	−18.33 (.0156)	−13.33 (.1250)	1/12/11
MetaRule − T1	+21.67 (.0007)	+23.33 (.0430)	+20.00 (.0078)	15/7/2
MetaRule − B0	+5.83 (.2585)	+5.00 (.6719)	+6.67 (.4063)	8/11/5

The HMMT comparison is a particularly informative complete-system case because all three arms use the identical 24-task held-out inventory, model, and five-sample verifier protocol. Native evolution falls from 51.67% Avg@5 for B0 to 35.83% for T1, whereas the MetaRule treatment reaches 57.50%. On the 12 cross-type tasks, the corresponding values are 48.33%, 35.00%, and 55.00%. The complete diagnosis–revision–closure–publication mechanism therefore mitigates the loss introduced by T1 and exceeds the B0 point estimate.

Table 4 treats each held-out task's five-sample mean as one observation and reports exact task-level sign-flip tests. On cross-type, MetaRule improves eight tasks, ties four, and degrades none; the paired gain is 20.00 points ($p = 0.0078$, 95% task-bootstrap interval $[10.00, 31.67]$). Across all tasks, MetaRule also improves Pass@5 by 29.17 points ($p = 0.0156$) and Cons@5 by 25.00 points ($p = 0.0313$) relative to T1. These task-paired statistics compare the frozen outcomes of the completed five-iteration trajectories.

The task-transition visualization complements the claim-bearing paired statistics in Table 4: the left panel gives the treatment ladder, the middle panel preserves task identity, and the right panel isolates recovery from T1-harmed tasks.

The task-level transitions identify a recovery pattern rather than a uniform baseline shift. T1 lowers Avg@5 on 11 of 24 tasks and improves only one. MetaRule improves all 11 T1-harmed tasks, restores eight to at least their B0 score, and draws 88.5% of its net MetaRule–T1 gain from this harmed subset. In the cross-type stratum, T1 harms four tasks; MetaRule improves all four and restores two to B0, while also improving four tasks that T1 had not harmed. Thus the T1 ablation identifies negative-transfer repair as the dominant source of the complete procedure system's gain. MetaRule also exceeds B0 by 5.83 points overall and 6.67 points on cross-type tasks.

The recovery spans all four subject areas; detailed domain slices are provided in Appendix B.

Mechanism boundary summary. The complete B0–T1–MetaRule comparison improves HMMT cross-type Avg@5 from 35.00% to 55.00% (8 wins, 4 ties, 0 losses; $p = .0078$). Frozen dispatch has a positive point estimate but an interval crossing zero, while trajectory-shadow comparisons range from -1.67 to $+6.67$ points with all intervals including zero. These interventions bound immediate execution and persistence; the complete-system transfer estimate remains the primary result.

The complete transfer-strata plot is retained in the appendix; the HMMT case figure above is the primary visual ablation in the main text.

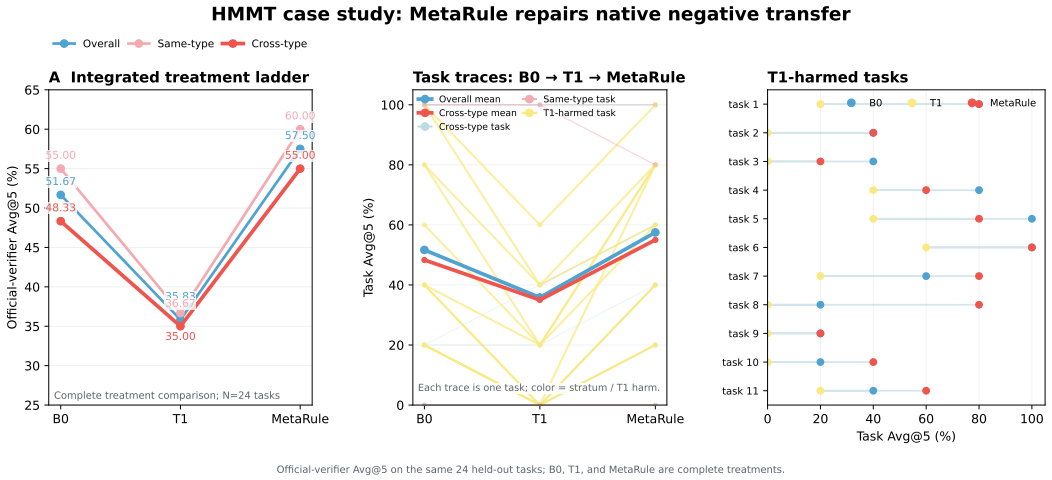


Figure 3: HMMT (2025+2026) ablation case study. (A) The integrated B0–T1–MetaRule treatment ladder, reported as official-verifier Avg@5 for the full held-out set and its two strata. (B) One trace per held-out task, showing its B0, T1, and MetaRule Avg@5 in the same order; green traces are cross-type and gray traces are same-type. (C) Descriptive accounting of the tasks harmed by T1 and subsequently improved or restored by MetaRule. The panels compare complete treatments and task-level recovery. The separate frozen-dispatch and trajectory-transfer estimands remain in the accompanying tables.

6 Discussion and Limitations

The experiments support a system-level account of procedural adaptation. The complete treatment raises GPQA-Diamond cross-domain Avg@5 from 72.63% under T1 to 82.11%, and HMMT cross-type Avg@5 from 35.00% to 55.00% (8/4/0 wins, ties, and losses; $p = 0.0078$). On HMMT, it improves every task harmed by T1 and obtains 88.5% of the treatment gain from that subset, identifying repair of native negative transfer as the central observed behavior.

The evidence ladder is correspondingly bounded: frozen dispatch is directionally positive but uncertain, while trajectory-shadow estimates range from -1.67 to $+6.67$ points and cross zero (Pair 2 overall $+4.17$, $p = .4575$). These analyses audit the transaction boundary but do not identify persistence as an isolated cause. Each primary treatment has one five-iteration training trajectory, so independent seeds and larger admission cohorts remain the key next tests.

7 Conclusion

We present MetaRule, a transactional procedure-learning mechanism for self-evolving harnesses. It converts answer-free checkpoints into scoped successor behavior through isolated diagnosis, same-thread revision, typed closure, paired admission, and delayed publication, while keeping gold in the official verifier and held-out evaluation read-only. The complete treatment improves over T1 by 9.48 points on GPQA-Diamond cross-domain Avg@5 and 20.00 points on HMMT cross-type Avg@5.

The HMMT gain is concentrated in native negative-transfer repair: MetaRule improves all four cross-type tasks harmed by T1 and restores two to at least their B0 outcome. This pattern suggests that transactionally governed diagnosis, revision, and guarded transfer can stabilize a harness under task shifts.

The mechanism evidence also sets a boundary: frozen dispatch is favorable but uncertain, and trajectory-shadow estimates range from -1.67 to $+6.67$ points with intervals crossing zero. Thus the complete treatment is the system-level estimand; persistence alone is not identified. The exploratory biology probe shows that typed candidate extraction can operate outside the registered domains, but not that the treatment improves biology performance or successor-state efficacy.

More broadly, MetaRule makes harness evolution inspectable as frozen state transitions rather than opaque prompt accumulation. Independent training seeds, larger admission cohorts, and tool-mediated domains should next test stability, efficiency, and procedure efficacy separately. The central contract remains: only closed, scoped, and auditable evidence may change the next harness state.

REFERENCES

Maciej Besta, Ales Blach, Robert Kubicek, Jan Gerstenberger, Michal Podstawski, Jakub Gajda, Tomasz Lehmann, Jan Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Mert Cemri, Andrei Cojocaru, Melissa Pan, Shu Liu, Shubham Agarwal, Alexander Krentsel, Jay Tang, Kannan Ramchandran, Joseph E. Gonzalez, Matei Zaharia, Alex Dimakis, and Ion Stoica. Fantastic adaptive taxonomies and how to use them, 2026. URL https://arxiv.org/abs/2607.16387.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing, 2023. URL https://arxiv.org/abs/2305.11738.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2024. URL https://arxiv.org/abs/2408.08435.

Yue Huang, Wenjie Wang, Han Bao, Yuchen Ma, Xiaonan Luo, Yi Nian, Haomin Zhuang, Zheyuan Liu, Yue Zhao, and Xiangliang Zhang. MemoHarness: Agent harnesses that learn from experience, 2026. URL https://arxiv.org/abs/2607.14159.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Karthik Narasimhan, Ofir Press, and Kilian Lieret. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Li, Kexin Tang, Yi He, Hu Zheng, Yitao Zhang, Qian Liu, Lin Shen, et al. Agentbench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: A benchmark for general ai assistants. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2311.12983.

Charles Packer, Vivian Fang, Shishir Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

Timo Schick, Jane Dwivedi-Yu, Federico Dessee, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL https://arxiv.org/abs/2305.16291.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024b. URL https://arxiv.org/abs/2409.07429.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2023a. URL https://arxiv.org/abs/2309.03409.

Kaiyu Yang, Aidan Swoa, Yining Miao, Shun Deng, Yixuan Song, Tong Wu, et al. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, 2023b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Karthik Narasimhan, Yuan Cao, Gen Xu, Xue Wang, and Arvind Narayanan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models, 2023b. 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. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.12045.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners, 2023. URL https://arxiv.org/abs/2308.10144.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*, 2024. URL https://arxiv.org/abs/2412.06559.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Arman Kwon, Howard Chen, Adam Litz, Alfred Xie, Sonya Yu, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

APPENDIX CONTENTS

A MetaRule Algorithms **13**

B Instantiated Learning Policy **14**

C Gold-Isolation Interface **14**

D Audited Case Studies of Procedural Transfer **15**

D.1 Case 1: Rejection becomes a rule, then changes a held-out procedure 15

D.2 Case 2: A MetaRule aligns with fewer repeated mistakes 15

D.3 Case 3: The gate prevents a repeatedly harmful repair from propagating 16

D.4 Case 4: Distinct MetaRules restore a stable HMMT plurality 16

D.5 Case 5: A MetaRule accompanies first success on an unseen problem type 17

D.6 Additional benchmark-level support 18

D.7 Cross-iteration transfer summary 18

D.8 Trace walkthrough: from a closed checkpoint to cross-domain correction 20

D.9 Dynamics of the evolving harness and rule channel 20

D.10 Task-level outcome distributions 21

D.11 Evidence matrix and claim boundaries 22

E Data Leakage and Split Integrity **23**

F Lifecycle Supplement **24**

G Resource and Audit Supplement **24**

H Out-of-Domain Biology Extraction Stress Test **25**

I AI Use Disclosure **27**

A METARULE ALGORITHMS

Algorithm 1 first records the native T1 transition on which the full method is built. It separates within-batch inference from post-batch learning and invokes the controller gate only on the pre-registered schedule Γ.

Algorithm 1 Native T1 harness evolution

Require: Ordered train tasks $\mathcal{D}_{\text{tr}}$, initial $(B_1, H_1, C_1, \mathcal{E}_{<1})$, schedule Γ, iterations T
for $t = 1$ to T **do**
  Freeze $(B_t, H_t, C_t, \mathcal{E}_{<t}, \Pi)$ for the complete batch
  for all $x \in \mathcal{D}_{\text{tr}}$ in parallel **do**
    Retrieve bounded train-only procedure packet from $\mathcal{E}_{<t}$
    Adapt H_t to public target features and execute the task
    Obtain one terminal outcome from the OFFICIAL VERIFIER
  end for
  Require the complete ordered outcome batch $\overline{\mathcal{O}}_t$
  Stage answer-free procedure records and update ExperienceBank B_{t+1}
  if $t \in \Gamma$ **then**
    Propose D1–D6 configuration C'_t from the closed post-batch state
    $C_{t+1} \leftarrow \text{PAIREDCONTROLLERGATE}(C_t, C'_t)$ without bank writes
  else
    $C_{t+1} \leftarrow C_t$
  end if
  $H_{t+1} \leftarrow \text{Publish}(H_t, C_{t+1})$
  Atomically append the staged procedure records to $\mathcal{E}_{\le t}$
  Publish bank and harness digests for iteration $t + 1$
end for
return Frozen final T1 bank, harness, procedure bank, and transition receipts

Algorithm 2 exposes the full transactional boundary. The notation matches Section 3: S_t is immutable within an iteration, $\mathcal{C}$ is the typed harness compiler, and both gates consume complete official outcomes. The final assertion enforces transfer of a validated positive or precautionary artifact before the successor state is committed.

Algorithm 2 Transactional METARULE training

Require: Ordered train tasks $\mathcal{D}_{\text{tr}}$, initial state S_1, iterations T
for $t = 1$ to T **do**
  Freeze $S_t = (B_t, H_t, \mathcal{E}_{<t}, \mathcal{R}_t, C_t, \Pi)$
  for all $x \in \mathcal{D}_{\text{tr}}$ in parallel **do**
    Build bounded train-only packet $P_{t,x}$ from S_t
    $\widehat{H}_{t,x} \leftarrow \mathcal{C}(H_t, G_x, P_{t,x}, \mathcal{R}_t, C_t)$
    $o_{t,x} \leftarrow \text{EXECUTENATIVETASK}(x, \widehat{H}_{t,x}, \Pi)$
    $v_{t,x} \leftarrow \text{OFFICIAL VERIFIER}(o_{t,x})$
  end for
  Require one official outcome per task; retain operational failures for recovery
  Project answer-free experiences $\{e_{t,x}\}$ and stage bank update B'
  Propose controller configuration C'_t and distilled rule candidates $\mathcal{Q}_t$
  $C_{t+1} \leftarrow \text{PAIREDCONTROLLERGATE}(C_t, C'_t)$
  $H_{t+1} \leftarrow \text{Publish}(H_t, C_{t+1})$
  $\mathcal{A}_t \leftarrow \text{PAIREDRULEGATES}(\mathcal{R}_t, \mathcal{Q}_t)$
  if $\mathcal{A}_t$ contains no transferable rule **then**
    Add a conservative executable reverse constraint from closed batch telemetry
  end if
  Require at least one admitted positive or reverse transfer artifact
  Atomically publish $S_{t+1} = (B', H_{t+1}, \mathcal{E}_{\le t}, \mathcal{R}_{t+1}, C_{t+1}, \Pi)$
end for
return Frozen final state S_{T+1} and replay receipts

Algorithm 3 shows that diagnosis is not a critique of the current answer. It reads only a closed checkpoint projection and returns typed control information. Semantic repair remains task-local, happens in the original thread, and must satisfy closure before its answer can replace the initial answer.

Algorithm 3 Native task execution with isolated diagnosis and closure

Require: Public task x, compiled harness $\widehat{H}$, frozen policy Π
Start native thread and execute the initial segment under cumulative budget
Observe complete event boundaries until the first checkpoint decision Q
Settle interruption or early completion and replay the checkpoint prefix
$d^{\mathrm{phase}} \leftarrow \mathcal{D}(\pi(Q))$ using a stateless, answer-oblivious request
$d^{\mathrm{sem}} \leftarrow \mathcal{M}(d^{\mathrm{phase}}, G_x, \mathrm{Format}(a))$
if d^{sem} requests revision **then**
    Select admitted rule programs whose trigger matches d^{sem}
    Resume the same native thread and execute the bounded repair program
    Assess typed closure: resolution, contradiction, evidence kind, and checks
    if closure fails and corrective budget remains **then**
        Perform one bounded corrective revision and corrective closure
    end if
end if
Select revised answer only if the typed closure contract admits it; otherwise preserve initial answer
return Public answer, usage ledger, checkpoint, diagnosis, closure, and replay receipts

B Instantiated Learning Policy

Table 5 records the instantiated policy used by the current GPQA-Diamond configuration. These values define the engineering protocol rather than a statistical significance test.

Table 5: Instantiated METARULE learning and execution policy.

Policy field	**Value**
Training iterations	5
Training tasks per iteration	125
Task model and reasoning effort	GPT-5.5, medium
Task concurrency	4
Task tools	disabled
Transport retries	at most 3
Candidate rules per iteration	2
Minimum candidate support	3
Gate tasks per domain	3
Paired repetitions	3
Bootstrap resamples	2,000
Bootstrap confidence	80%
Minimum mean gain	strictly greater than 0
Maximum tolerated lower-bound harm	0.05
Controller gate schedule	every iteration
Rule gate schedule	every iteration

C Gold-Isolation Interface

The interface boundary can be summarized by which component may consume each field. The task agent reads the public question and compiled policy. The isolated diagnoser reads only checkpoint telemetry. The compiler and distiller read public IR, sanitized experience, and admitted rules. The controller reads the complete training batch after official verification but receives no raw gold value. Only the OFFICIAL VERIFIER maps the selected public answer to the private reference. Development and held-out executors bind a frozen admission snapshot

and have no update capability. This capability separation is checked by persisted request schemas and replay rather than by prompt wording alone.

D AUDITED CASE STUDIES OF PROCEDURAL TRANSFER

This section follows the HCC-style progression from a rule's birth to its downstream behavior: provenance, task-local action, repeated-sample stability, harm prevention, cross-task reuse, and a failure boundary. The primary records come from the completed five-iteration GPQA-Diamond run `tfull-diamond-v31-native-full-train-20260915-r30`; three complementary cases come from the independently trained HMMT run. We report only public answers and typed receipts. Private references remain accessible only to the official verifier. These traces establish executable mediation and the observed task-level outcomes.

D.1 CASE 1: REJECTION BECOMES A RULE, THEN CHANGES A HELD-OUT PROCEDURE

Diamond iteration 5 proposed `diamond-v31-i5-restore_derivation-conservative-s0`: when a derivation is missing, verify the public givens, derive independently, and preserve supported work. Its paired gate found execution on $3/6$ tasks, mean task delta -0.1111, and zero wins versus two losses. It therefore rejected the positive proposal for `insufficient_execution_coverage`. Rejection analysis `ra-a0c027cfb49970b0` explicitly set `causal_claim=false`; the system did not label the proposed repair as beneficial.

The gate is followed by a concrete state transition. Reverse distillation published `negative-9a30b07278bdc961`, an executable precautionary rule that forbids blind reuse of the rejected parent procedure and, only under `missing_derivation`, requires `verify_givens` and `independent_derivation`. The rule entered the frozen iteration-5 state and was available to read-only held-out evaluation. Thus rejection does not end the learning path: it changes what the next harness is allowed to do while retaining the negative evidence that motivated the constraint.

On unseen Biology task `reccVFW11Elovnf8T`, sample 4 supplies the downstream execution. The closed checkpoint is diagnosed as `missing_derivation`; the frozen rule is attempted and executed in the same native thread. The resumed procedure checks that Cre removes the paired `lox2272` stop cassette and that the CBA promoter plus IRES make a missing-enhancer explanation unsupported. An independent derivation then identifies an out-of-frame receptor–eGFP junction. The public answer changes `B`→`A`; closure records both required checks with no new contradiction, and the official verifier records correct with `invalid=0`. Across five samples the task is $5/5$ for MetaRule and $4/5$ for both B0 and T1. This is the complete post-gate story: rejected proposal → reverse constraint → frozen harness state → triggered operations → admitted revision → official outcome.

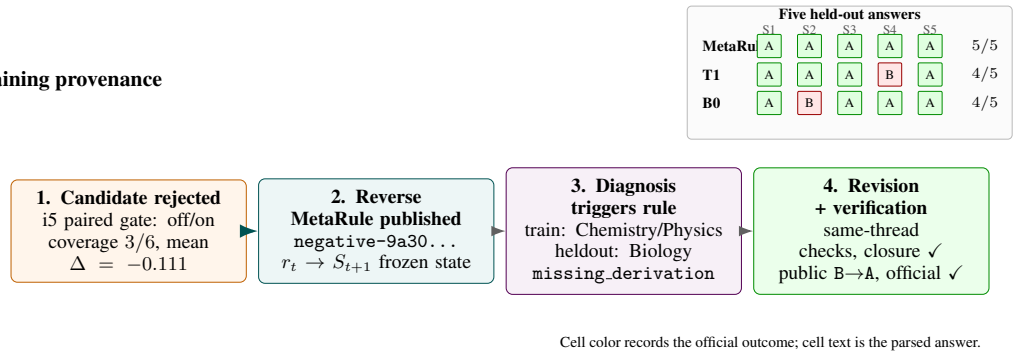


Figure 4: Case 1, from rejected proposal to held-out use. The training lane records why the positive candidate is not inherited and how reverse distillation changes the next frozen harness. The execution lane records a later trigger match, required operations, answer revision, closure, and official outcome. The inset shows five independent samples. Each tile contains the parsed answer; green and red backgrounds indicate official correct and incorrect outcomes.

D.2 CASE 2: A METARULE ALIGNS WITH FEWER REPEATED MISTAKES

Task `reckl6PkaEMWM50yZ` makes the Cons@5 mechanism visible. B0 produces `[A,A,A,C,A]` and is correct only $1/5$, so its plurality answer is wrong. T1 produces `[A,C,A,C,C]` and reaches $3/5$. MetaRule produces

`[A,C,C,C,C]`, reaches $4/5$, and makes the correct answer the stable plurality. In MetaRule samples 2, 3, and 5 the same frozen rule is executed; all three official outcomes are correct. The two non-executed samples contain one correct and one incorrect outcome.

Sample 2 exposes the behavioral change. Its first segment selects `D` after incorrectly treating the attacked epoxide carbon as already methyl-substituted and therefore predicting a gem-dimethyl product. The `missing_derivation` trigger dispatches `negative-9a30b07278bdc961`. Rechecking substituent locations and independently tracking attack and inversion excludes the gem-dimethyl choices and changes the answer to `C`; closure admits the revision and the official verifier marks it correct. The rule does not encode option `C`; it encodes the checks that remove the recurrent shortcut.

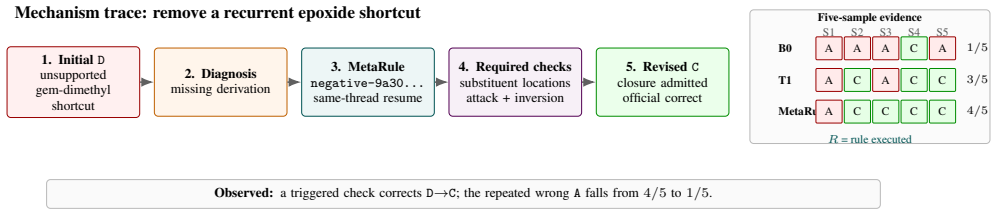


Figure 5: Case 2, shortcut removal and repeated-sample stability ($N = 5$). The mechanism trace is primary; the compact inset records the five outcomes. MetaRule reduces the repeated `A` error from four B0 samples to one, while one audited execution changes `D`→`C`.

This case mirrors the aggregate Diamond pattern. Pass@5 and Best@5 are identical for B0, T1, and MetaRule at 96.36%, while Avg@5 rises from 82.55% and 83.27% to 88.73%, and Cons@5 rises from 87.27% and 89.09% to 94.55%. The complete treatment therefore improves repeated-generation stability on tasks that were already solvable at least once, rather than expanding Pass@5 coverage in this evaluation. The trace is observational: diagnosis controls rule execution, and the compared arms also differ in other MetaRule components.

D.3 Case 3: The gate prevents a repeatedly harmful repair from propagating

HMMT iteration 4 provides a direct repeated-harm case. Positive candidate `hmmt-v32-i4-derive_missing_step-conservative-s0` forced a new derivation under `missing_derivation`. On task `hmmt_feb_2025-28`, all three rule-off repetitions returned the correct $\sqrt{6}$. Rule-on returned two incorrect answers, $\sqrt{30}/2$ and $\sqrt{5}$, and one correct $\sqrt{6}$. The task delta was $-2/3$; the gate-wide mean delta was -0.2222, execution coverage was 1.0, and the receipt recorded two harms on the same task.

The gate rejected the positive candidate with `reason=repeated_task_harm` and published reverse constraint `negative-c945abef5d40967e`. In iteration 5 the same task was solved as $\sqrt{6}$; diagnosis was `none`, no forcing repair fired, and the initial answer was retained. The gate prevents a demonstrably harmful intervention from becoming inherited behavior, and the next harness keeps the correct initial answer on that task.

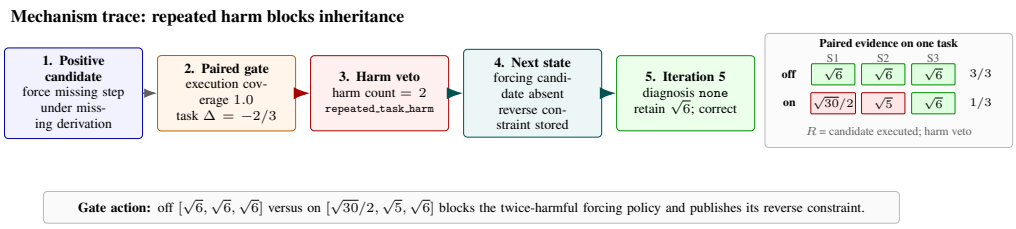


Figure 6: Case 3, repeated-harm prevention. The paired evidence is off $[\sqrt{6}, \sqrt{6}, \sqrt{6}]$ versus on $[\sqrt{30}/2, \sqrt{5}, \sqrt{6}]$; tile colors record official correctness, and the gate publishes a reverse rule after the harm veto.

D.4 Case 4: Distinct MetaRules restore a stable HMMT plurality

HMMT task `hmmt_feb_2025-27` demonstrates rule diversity rather than another copy of Diamond's missing-derivation pattern. B0 is correct $1/5$ with five dispersed answers; T1 is correct $0/5$ and repeats the wrong answer $4/15$ twice. MetaRule is correct $4/5$, with $7/18$ becoming the plurality answer.

Two independently distilled rule families appear in the five MetaRule executions. Sample 1 diagnoses `format_error` and executes `negative-d1dc5c318c271e51`, which verifies the givens while preserving the supported coordinate derivation; it returns $7/18$ and is correct. Samples 2, 4, and 5 diagnose `reasoning_contradiction` and execute `negative-ebdaf5f3e7691275`, which verifies the givens and derives independently. Samples 4 and 5 return $7/18$ and are correct, while sample 2 returns $49/171$ and is incorrect. Sample 3 executes no rule and is also correct. Overall, three of four rule-executed samples are correct, and the full treatment converts an unstable wrong set into a $4/5$ correct plurality.

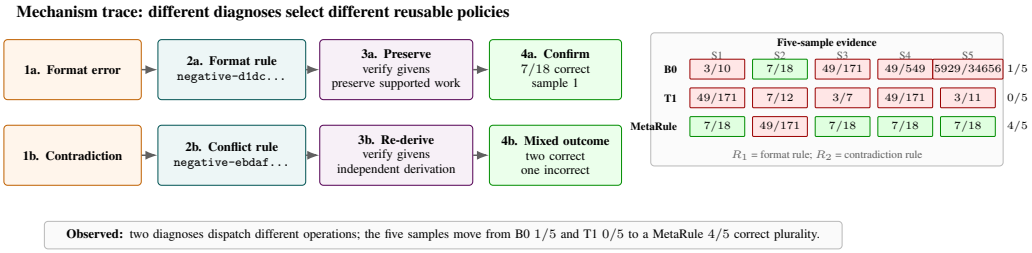


Figure 7: Case 4, diagnosis-conditioned rule diversity ($N = 5$). A format diagnosis selects a preservation policy, while a contradiction diagnosis selects independent re-derivation. The compact inset gives all exact parsed fractions with correctness-colored cells; the retained incorrect execution shows the official outcome for each independent sample.

This HMMT run is independently trained; no Diamond rule crosses benchmarks. It shows that the same typed interface can instantiate distinct policies within another benchmark. The task trace aligns MetaRule execution with restored Cons@5, but does not isolate the rule channel from the rest of the HMMT MetaRule treatment.

D.5 Case 5: A MetaRule accompanies first success on an unseen problem type

The previous cases primarily explain repeated-sample stability. HMMT cross-type task `hmmt_feb_2026-8` instead exposes a coverage gain. B0 produces five incorrect answers and T1 also produces five incorrect answers, so both have Avg@5, Pass@5, and Best@5 equal to zero. MetaRule produces two correct answers out of five, raising Avg@5 to 0.40 and both Pass@5 and Best@5 to one. This is a task that neither control solves in five attempts but the complete treatment solves twice.

One of the two correct MetaRule samples contains a complete rule-execution chain. Sample 2 reaches candidate answer 279, receives `format_error`, and executes `negative-d1dc5c318c271e51`. The rule requires `verify_givens` and `preserve_supported`. The resumed thread explicitly rechecks the recurrence, modular inverses, the telescoping transformation, boundary conditions $a_0 = 1$ and $a_{2026} = 9$, and the final modular sum. It preserves 279, closure returns `resolved=true`, and the official verifier records correct.

This is evidence for confirmatory procedural mediation: a rule can stabilize and audit a supported derivation without changing its answer. It also links real rule execution to a cross-type task on which Pass@5 and Best@5 improve from zero to one. The second correct MetaRule sample executes no rule, so the two successes cannot be attributed to this rule alone; they remain evidence for the complete MetaRule treatment.

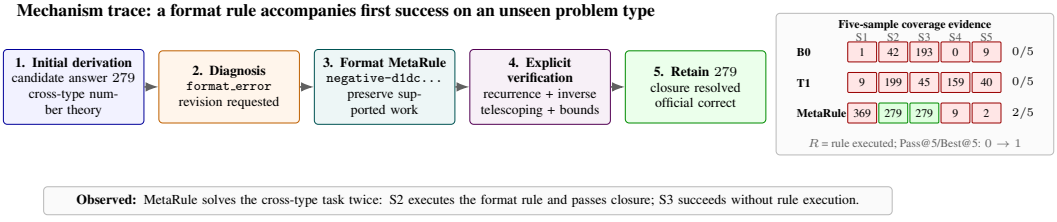


Figure 8: Case 5, first success on an unseen HMMT problem type ($N = 5$). The main flow shows a format-triggered preservation and verification procedure. The inset records every parsed integer and its correctness: neither B0 nor T1 succeeds in five attempts, whereas MetaRule succeeds twice and reaches Pass@5/Best@5 equal to one.

Table 6: Observed MetaRule families in the completed formal runs.

Benchmark	Trigger	Action	Required operations	Example rule
Diamond	`missing_derivation`	reject blind procedure reuse	verify givens; independent derivation	`negative-9a30b07278bdc961`
HMMT	`format_error`	`repair_format`	verify givens; preserve supported work	`negative-d1dc5c318c271e51`
HMMT	`reasoning_contradiction`	`resolve_conflict`	verify givens; independent derivation	`negative-ebdaf5f3e7691275`

Table 7: Compact outcome-completeness audit for claim-bearing local runs.

Benchmark	System	Expected	Terminal	Invalid	Verifier / state audit
GPQA-Diamond	B0	275	271	4	exact-choice / complete
GPQA-Diamond	T1	275	273	2	exact-choice / complete
GPQA-Diamond	MetaRule	275	275	0	exact-choice / unchanged
AIME2025	B0/T1/MetaRule	25 each	25 each	0	official integer / complete
HMMT (25+26)	B0/T1/MetaRule	120 each	120 each	0	MathArena / unchanged

Table 8: Complementary roles of the audited case suite.

Case	Observable transition	What it establishes	What it does not establish
1. Gate-to-execution	Rejected proposal → reverse rule → Biology `B→A`	Full rule provenance and later held-out procedural mediation	MetaRule-only causal gain
2. Diamond consistency	B0 $1/5$; T1 $3/5$; MetaRule $4/5$	Concrete shortcut removal aligned with fewer repeated mistakes	Randomized rule-only effect
3. Repeated-harm veto	Off $3/3$ correct; on $1/3$ correct; candidate blocked	Paired gate prevents a repeatedly harmful repair from being inherited	Reverse-rule execution caused the later correct answer
4. HMMT rule diversity	Format and contradiction rules; MetaRule $4/5$	Distinct typed rules operate in another benchmark and restore a correct plurality	Diamond-to-HMMT transfer or deterministic correction
5. First cross-type success	B0/T1 Pass@5 $= 0$; MetaRule Pass@5 $= 1$	Rule execution accompanies one verified first-success trace	Both successful samples were caused by the rule

D.6 Additional benchmark-level support

The final HMMT (25+26) held-out evaluation is complete and read-only: 24 tasks, five samples per task, 120 official-verifier outcomes, zero invalid outcomes, and no missing, provider, timeout, harness, or verifier failures. MetaRule reaches 57.50% Avg@5, 87.50% Pass@5, 66.67% Cons@5, and 87.50% Best@5; same-type and cross-type Avg@5 are 60.00% and 55.00%, respectively. Relative to the HMMT (25+26) B0 row, the complete treatment gains 5.83 points in Avg@5, 8.33 points in Pass@5, and 16.67 points in Cons@5. The complete T1 record reports 35.83% Avg@5, 58.33% Pass@5, 41.67% Cons@5, and 58.33% Best@5, with same-type/cross-type Avg@5 of 36.67%/35.00%. These are complete-system results and should not be attributed to MetaRule alone, since controller, diagnosis, revision, closure, and rule transfer are all active in the MetaRule treatment. They supply aggregate support that the complete method can be instantiated outside GPQA and improves transfer across mathematical problem types. This is a separate HMMT training run; it is not evidence that a Diamond-trained rule moved into HMMT.

The paired absolute and relative views of the registered transfer strata are shown together in Figure 2. The relative panel removes the B0 level and reports the corresponding percentage-point change for T1 and MetaRule.

The independent one-shot baselines in Figure 9 provide context for model capability on exactly the registered inventory tasks. Each panel also shows the local MetaRule first registered generation so readers can locate the full system on the same Acc@1 scale. These rows are not matched treatment arms, so we do not use their rank order as evidence for harness evolution.

D.7 Cross-iteration transfer summary

Table 9 summarizes the five transfer receipts. Every iteration published one executable artifact, but none of the five positive candidates passed the paired MetaRule gate. Iterations 1–4 were rejected primarily because the evidence was insufficient or harm was uncertain; iteration 4 additionally recorded repeated harm, and iteration 5 was rejected for insufficient execution coverage. The final held-out audit found the resulting negative rule

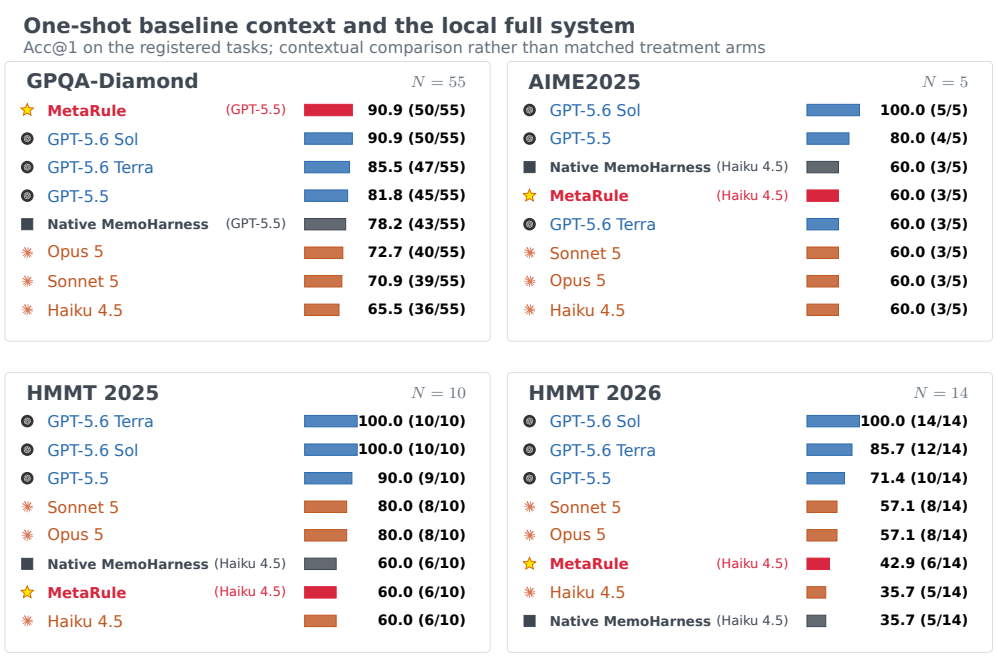


Figure 9: Contextual Acc@1 ranking with local Native MemoHarness and MetaRule rows included. MetaRule is shown in red with a yellow star, and Native MemoHarness in gray with a gray square. GPQA rows use GPT-5.5; AIME2025 and HMMT rows use Haiku 4.5. Blue bars with the OpenAI mark are Codex runs; orange bars with the Claude mark are Claude Code runs. External entries exclude EpiGraph conditions. These comparisons are separate from local Avg@5 and do not control model, budget, or agent implementation.

attempted on 40 tasks and executed on 39. Thus the run establishes that diagnosis, revision, closure, rejection analysis, and executable transfer are connected end to end. It does not establish a positive causal effect for MetaRules, because the admitted artifacts in this run were all conservative negative rules.

The end-to-end performance comparison is consistent with a system-level generalization effect: on the five-sample held-out audit, MetaRule reached 88.73% overall Avg@5 and 82.11% cross-domain Avg@5, compared with 83.27% and 72.63% for T1, respectively. These numbers are evidence for the complete MetaRule treatment (retrieval, diagnosis, revision, closure, controller, and rule transfer together). Because the five admitted rules in this run were negative rules, the case studies do not identify MetaRule alone as the cause of the gain; they show how the rule channel supplies auditable constraints to the evolving harness and how those constraints are executed on held-out tasks.

Table 9: Audited MetaRule transfer receipts from the five training iterations.

Iter.	Candidate trigger	Transfer artifact	Exec. cov.	Mean Δ	Gate result
1	missing derivation	negative-dff97f1aa34b092c	1/3	0.000	reverse
2	missing derivation	negative-cd802e34261166c2	1/6	0.111	reverse
3	missing derivation	negative-3b09ebc5e050da7f	0/6	0.056	reverse
4	missing derivation	negative-b7a31ab2665c302d	1/6	−0.111	reverse; harm veto
5	missing derivation	negative-9a30b07278bdc961	3/6	−0.111	reverse

The table also clarifies the relationship between harness evolution and MetaRule evolution. Controller gates may accept a D1–D6 harness change using complete train outcomes, while the rule gate independently decides whether a candidate is safe to transfer. In the formal run, both streams are recorded in the same iteration receipt; an accepted controller change is therefore not evidence that a MetaRule candidate was accepted, and a transferred reverse rule is not evidence that it improved accuracy.

D.8 TRACE WALKTHROUGH: FROM A CLOSED CHECKPOINT TO CROSS-DOMAIN CORRECTION

The following walkthrough uses the same typed vocabulary as the execution trace formalization in Section 3. It separates an action, its public observation, and the durable receipt that can be replayed. A model's statement that it performed a check is not itself evidence that the check occurred. The walkthrough follows the frozen `missing_derivation` rule through the primary held-out cross-domain case.

Table 10: Replayable trace walkthrough for the primary cross-domain correction.

Stage	**Action**	**Public observation**	**Durable receipt / acceptance condition**
Checkpoint	Native task reaches a closed interruption boundary	Settled status and trigger counters are available; answer and reasoning are absent from the diagnostic request	Checkpoint prefix, thread identity, cumulative usage, and replay decision; projection is closed before diagnosis
Diagnosis	Stateless isolated diagnosis consumes the telemetry projection	Typed phase failure and revision directive are returned	Diagnosis receipt contains phase class and schema version; no answer, reasoning, verifier output, or gold field is present
Mapping	Deterministic adapter maps phase class to semantic trigger	`missing_derivation` is selected	Semantic diagnosis receipt records trigger and evidence kind; adapter cannot inspect gold or verifier output
Rule dispatch	Compiler selects rules whose trigger and scope match	Reverse rule forbids blind parent-procedure reuse and requests public counterchecks	Rule-attempt receipt contains rule ID, trigger match, and attempted operations
Revision	Original native thread resumes with bounded repair program	Agent verifies givens and performs an independent derivation	Same-thread resume receipt binds continuation to the closed checkpoint and cumulative budget
Closure	Public closure evaluates required checks	Either all checks pass, or the revision remains unverified; a contradiction blocks replacement	Closure receipt records evidence kind, completed checks, resolution, contradiction, and verified flag
Verification	Official verifier evaluates selected public answer	Correct, incorrect, or invalid is assigned independently	Official-verifier receipt is the only receipt that may read the private reference
Frozen transfer	Training batches have already distilled and gated the rule	The held-out evaluator receives the sealed rule snapshot and cannot update it	State receipt binds the active rule-set digest; held-out writes no bank, controller, prompt, or rule state
Outcome	The selected answer is scored and aggregated	Sample 4 changes `B`→`A`; the task is $5/5$ under MetaRule	Official receipt records correct and non-invalid; task-level comparison is T1 $4/5$, B0 $4/5$, MetaRule $5/5$

For the concrete Diamond record, the native projection contains the initial `FINAL_ANSWER: B`; the action receipt binds `negative-9a30b07278bdc961` as both attempted and executed; its closure lists `verify_givens` and `independent_derivation` as completed checks; and the selected segment contains `FINAL_ANSWER: A`. The separate official-verifier receipt assigns reward 1 with `correct=1` and `invalid=0`. These fields establish that the rule was bound to the revision program and that the public closure receipt accepted the resulting contract; they do not independently verify hidden reasoning or rule out every post-hoc explanation.

D.9 DYNAMICS OF THE EVOLVING HARNESS AND RULE CHANNEL

The trace above is local to a task. The complementary dynamics view records what became available after each batch barrier. A controller transition changes the native D1–D6 harness stream; a rule transition changes the typed procedure stream. They are written in the same iteration receipt but gated independently, and neither can affect tasks still running in its source batch.

Table 11: Five-iteration MetaRule dynamics in the audited Diamond run.

Iter.	Candidate evidence	Paired-gate decision	Published artifact	Later executable effect
1	Missing-derivation repair; coverage $1/3$	Positive deferred for insufficient evidence	`negative-dff97f1aa34b092c`	Avoid blind reuse; verify givens and derive independently
2	Same trigger; coverage $1/6$	Positive rejected; no invalid regression	`negative-cd802e34261166c2`	Repeated exposure of guarded derivation procedure
3	Same trigger; coverage $0/6$	Positive rejected for absent execution	`negative-3b09ebc5e050da7f`	Countercheck remains available without claiming benefit
4	Countercheck candidate; mean delta -0.1111	Harm veto after repeated harm	`negative-b7a31ab2665c302d`	Prevent unsafe positive propagation
5	Same trigger; coverage $3/6$	Positive rejected for incomplete evidence	`negative-9a30b07278bdc961`	Final frozen rule set is auditable on held-out tasks

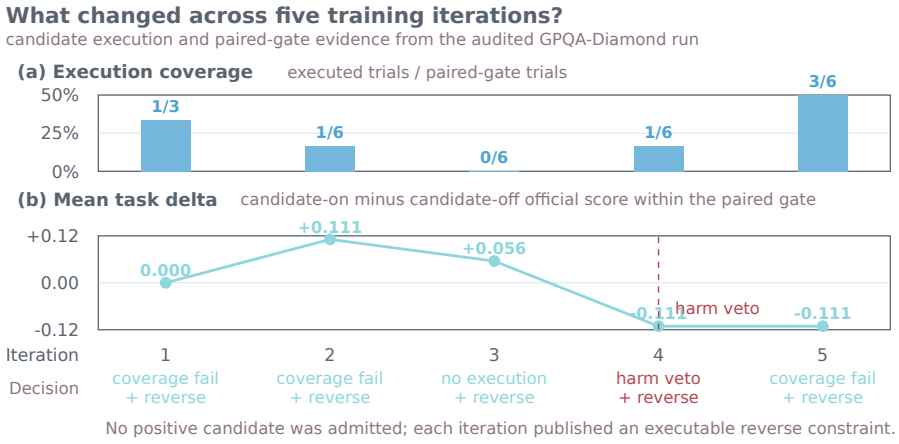


Figure 10: Receipt-backed five-iteration MetaRule dynamics for the audited GPQA-Diamond run. The upper bars show executed paired-gate trials over all paired-gate trials; the lower panel shows candidate-on minus candidate-off mean official score. Every iteration published an executable reverse artifact, and the dashed marker identifies the iteration-4 repeated-harm veto.

The run did not accumulate five positive repair policies; it accumulated five executable conservative constraints. Their frozen receipts make every artifact auditable, while the held-out action receipt records which trigger-matching rule was actually dispatched. The rule stream therefore affected harness evolution by changing which operations were admitted after a diagnosis, while the controller stream could change the broader D1–D6 configuration. A controller acceptance cannot be read as evidence that a rule passed, and a reverse transfer cannot be read as evidence that it improved accuracy.

D.10 Task-level outcome distributions

Aggregate metrics can hide whether changes are broad or concentrated. Figure 11 therefore groups tasks by how many of their five generations are correct before and after evolution. Each square matrix uses the B0 correct count as its row and the evolved-system correct count as its column. Cells to the right of the diagonal are task-level improvements; cells to the left are regressions. We show GPQA-Diamond and HMMT, where the task counts are large enough for this distributional view.

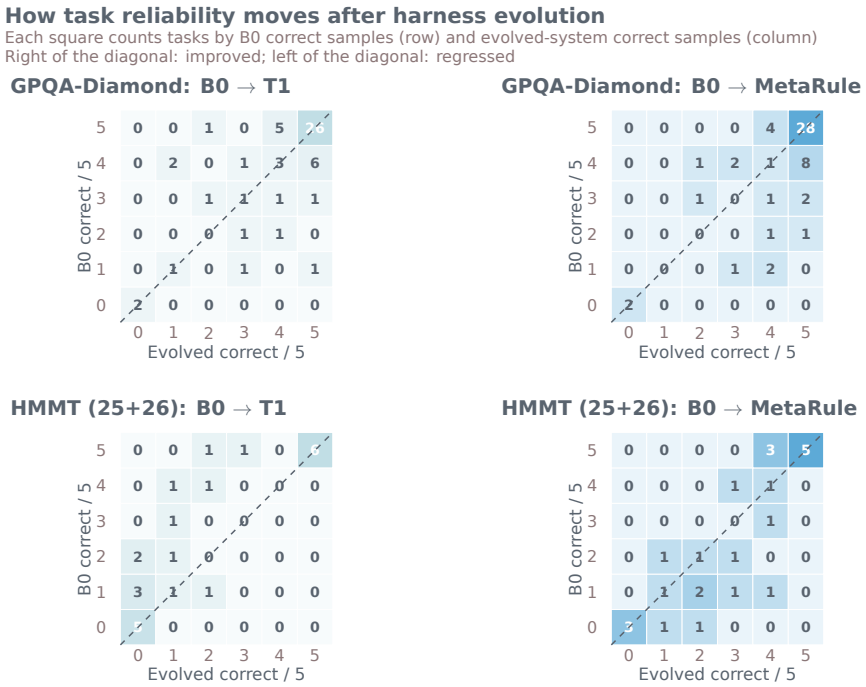


Figure 11: Square task-transition heatmaps for five-sample reliability. Rows are the number of correct B0 generations (0–5); columns are the number after native evolution (T1, orange) or the MetaRule full system (T-full, green). Each cell contains a task count. The dashed diagonal denotes no change, so cells to its right improved and cells to its left regressed.

D.11 Evidence matrix and claim boundaries

The case studies support different claims at different evidential levels. The matrix below prevents a process receipt from being silently promoted to a causal performance claim.

Table 12: Claim-to-receipt matrix for the MetaRule case studies.

Claim	Required evidence	Current case evidence	Status
Diagnosis is answer-oblivious	Closed projection and request-schema audit	Isolated diagnosis receipt plus replayed projection; answer/reasoning/gold absent	Supported
Revision is same-thread and bounded	Native resume identity, cumulative usage, interruption receipt	Primary cross-domain trace and task execution receipt	Supported
Closure admits revision contract	Required checks, contradiction flag, selected-answer decision	Diamond correction traces and HMMT rule executions record completed checks; official verifier separately scores each answer	Supported structurally
Rules cross the batch barrier	Candidate, gate, frozen-state digest, later attempted-rule ID	Five-iteration transfer table and held-out attempted/executed rule IDs	Supported
Rule execution changes later procedure	Initial and revised public answers, executed operations, closure	`reccVFW11Elovnf8T`: B→A; `rec4L69T0Y1AS4AFS`: A→C	Supported observationally
The rule transfers across held-out scientific domains	Frozen rule provenance, source-domain supports, target split, executed operations, official verifier	Chemistry/physics supports only; two Biology corrections and one Chemistry comparison under `negative-9a30b07278bdc961`	Supported observationally
The paired gate prevents repeated harm	Repeated rule-off/on outcomes, execution IDs, task deltas, transferred successor	HMMT iteration 4: off $3/3$ correct, on $1/3$ correct for one task; positive candidate rejected for repeated harm	Supported for the gate decision
Rule execution aligns with fewer repeated errors	Complete per-task five-sample outcomes and execution IDs	Diamond `reckl6PkaEMWM50yZ`: B0 $1/5$, T1 $3/5$, MetaRule $4/5$; three executed samples are correct	Descriptive
Distinct rule families execute in another benchmark	Typed rule IDs, triggers, operations, closures, and official outcomes	HMMT `hmmt_feb_2025-27`: format and contradiction rules execute; MetaRule reaches $4/5$ and correct Cons@5	Supported observationally
The complete method expands task coverage	Per-task five-sample official outcomes and a correct rule-executed trace	HMMT `hmmt_feb_2026-8`: B0/T1 Pass@5 $= 0$, MetaRule Pass@5 $= 1$; one correct sample executes the format rule	System-level evidence
MetaRule alone raises accuracy	Matched MetaRule-off/on verifier outcomes	MetaRule vs T1 changes diagnosis, revision, closure, controller scheduling, and procedure transfer together	Not identified
MetaRule improves cross-domain generalization	Complete frozen held-out outcomes with strata	Diamond cross-domain Avg@5 82.11% vs T1 72.63%; HMMT (25+26) cross-type 55.00% vs B0 48.33%	System-level evidence

The appropriate interpretation is layered. Trace and dynamics receipts establish that MetaRule is an executable component of harness evolution, not a post-hoc label. Held-out aggregates establish that the complete MetaRule treatment performs better on the reported cross-domain or cross-type strata. The remaining causal gap is mediation: a matched task-level comparison must show that rule-on changes a specified operation and then changes the official outcome while diagnosis, revision, closure, controller state, and execution protocol are fixed. Until that comparison is available, these are mechanism-validating case studies and the generalization result is a complete-system claim.

E Data Leakage and Split Integrity

The data-flow boundary is enforced at three interfaces. First, training-state updates read only the ordered training tasks and their terminal official-verifier outcomes; the development set can select a state that was already frozen but cannot modify it. Second, held-out evaluation binds a sealed state digest and has no write capability to the ExperienceBank, controller, harness, prompts, retrieval policy, candidate set, or MetaRule set. Third, diagnosis receives a closed telemetry projection only: it contains checkpoint counters, phase/status classes, and completion metadata, while omitting the current answer, reasoning, gold answer, verifier result, raw response, and task identity fields used for answer reconstruction. The official verifier is the only component that reads the private reference answer.

The registered split manifests are immutable for each benchmark. Task order, model, verifier, retry policy, and held-out identifiers are shared across B0, T1, and T-full within a benchmark. External pure-LLM rows are retained only as contextual one-outcome references; they are not used to tune the method or to choose a held-out split. Before and after digests are compared for every read-only evaluation, and any mismatch invalidates the corresponding audit rather than being converted into a score.

For the claim-bearing GPQA and HMMT runs, the resolved launcher imports `phase7.diamond_agent_v3:GPQADiamondV3Agent` and its telemetry-only isolated-diagnosis implementation, and validates the request against its field whitelist before dispatch. Older qualification launchers whose prompts mention the current answer or reasoning are legacy paths and are excluded from all reported

treatment aggregates. The run manifest, implementation import path, request-schema digest, and exclusion list are retained with the immutable run receipts so that the leakage audit can be replayed independently.

Table 13: Field-access summary for the leakage audit. A check means the field is available to that component; a dash means it is absent from the request.

Component	Public task	Telemetry	Verifier outcome	Gold answer
Task agent	yes	yes	no	no
Isolated diagnosis	no	yes	no	no
Controller / distiller	no	sanitized	post-batch class	no
Official verifier	selected answer	no	yes	yes
Held-out updater	frozen snapshot	no	no	no

The complete four-metric table is reported once in the main text as Table 3. The appendix retains the underlying receipts, invalid counts, task-level audits, and benchmark-specific breakdowns rather than repeating the same aggregate rows.

F Lifecycle Supplement

Figure 12 gives the complete lifecycle that is abbreviated in the main mechanism figure: closed evidence is distilled, the efficacy gate or precautionary gate selects a typed artifact, the artifact enters the next frozen state, and later diagnosis/closure controls any task-local use.

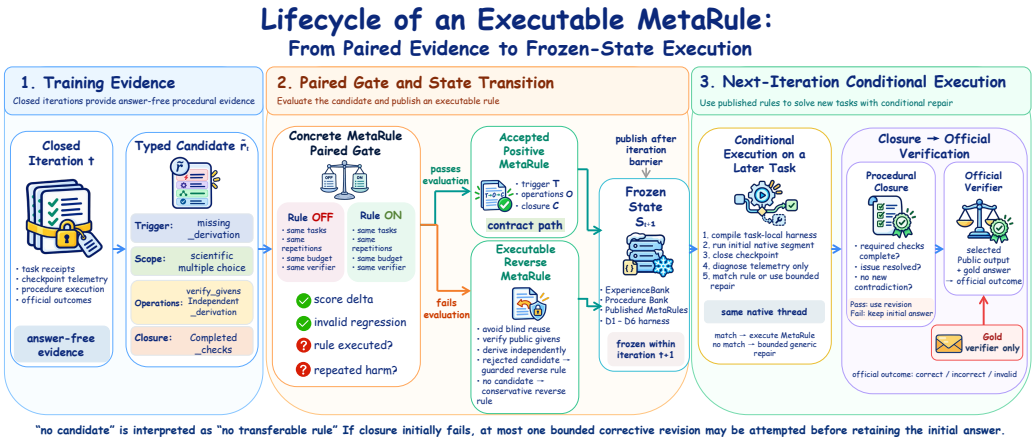


Figure 12: Complete MetaRule lifecycle. Closed answer-free evidence is distilled into a candidate, evaluated by a paired gate, admitted as a positive program or guarded reverse countercheck, and published only at the next batch barrier. Later trigger matching, same-thread revision, typed closure, suspension, and official verification complete the loop.

G Resource and Audit Supplement

The main text reports scientific outcomes and keeps resource accounting separate from accuracy claims. The following receipt-backed supplement records the accepted-outcome measurements used for that separation. A selected outcome is a terminal model call bound to the final aggregate; failed attempts, retry waiting, setup, verifier execution, and campaign gaps are excluded. Successful-call latency may still include provider queue or network time, so it is a model-turn measurement rather than hardware-only reasoning time. Missing fields remain missing and are not treated as zero.

For the matched GPQA-Diamond evaluation, the recorded total tokens per selected outcome are 7,027 for B0, 12,189 for T1, and 14,593 for T-full; the corresponding successful-call latencies are 18.3, 17.8, and 25.2 seconds. The associated five-sample Avg@5 values are 82.55%, 83.27%, and 88.73%. These measurements describe the complete treatment cost and do not constitute a budget-matched causal comparison. The full cross-benchmark tables and plots are retained here for reproducibility.

The complete split manifests, task identifiers, gate receipts, invalid-recovery sensitivity, state digests, and evidence matrix are retained in the run assets listed in the reproducibility files accompanying this paper. These records are audit evidence and do not alter the estimands or treatment comparisons reported in the main text.

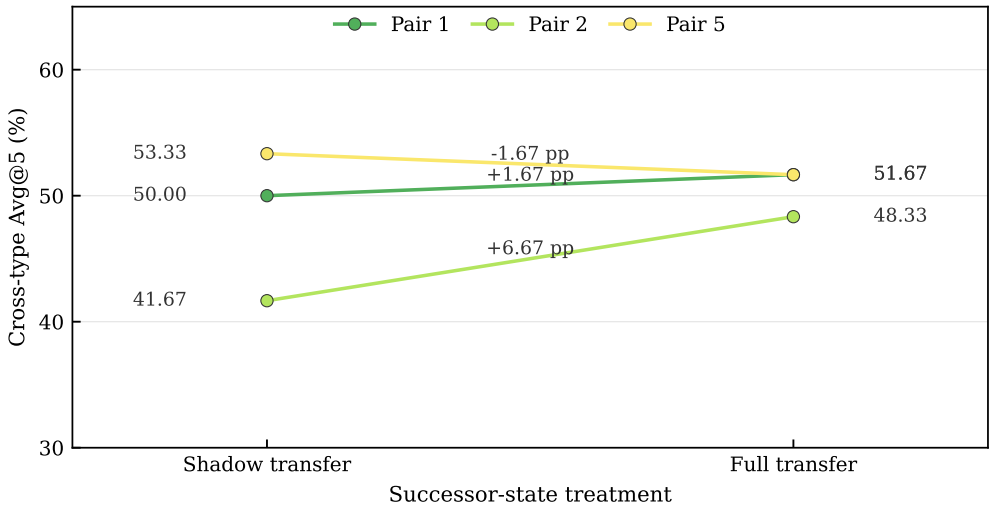


Figure 13: Matched HMMT trajectory-shadow comparisons on the cross-type held-out tasks. Each colored line connects the shadow and full treatment from one completed paired training order or registered comparison. Endpoints are official-verifier cross-type Avg@5 values over five generations per task. The lines are descriptive pair comparisons, not independent training seeds or a pooled significance estimate.

H Out-of-Domain Biology Extraction Stress Test

This section reports an exploratory mechanism probe on a public biology batch: the first 60 `biology` questions of the MMLU-Pro test split (Wang et al., 2024a), ten-option multiple choice, with the dataset's answer key as gold. The solver is Claude Haiku 4.5 and the diagnoser is GPT-5.5. All 60 tasks are discovery tasks, `min_support=3`. The task IR uses `domain=BIOLOGY`, `acceptance=[single_final_choice]`, base capabilities {`biological_reasoning, closed_book, exact_answer`}, and keyword-derived subfield tags (genetics 39 tasks, ecology/evolution 15, physiology 14, cell biology 12, plant biology 10, microbiology 5, general 7).

This is not a third benchmark treatment. In the paper's terminology, B0 is the native MemoHarness baseline, T1 is the native-evolution treatment (without typed diagnosis, closure, or MetaRule publication), and the MetaRule treatment is the complete lifecycle with admission and delayed publication. This probe exercises diagnosis, same-thread revision, closure, experience projection, and candidate distillation only; it has no B0/T1 comparison, paired admission, publication, successor-state execution, or held-out evaluation. Its result is therefore candidate-extraction feasibility, not an accuracy or transfer gain attributable to MetaRule.

Method change. Two deviations from the registered telemetry-only diagnosis interface, both confined to inputs. (i) The diagnoser receives the content-visible projection $\pi^{+}(Q) = \pi(Q) \cup \{$`public_derivation_answer_masked`$\}$ and selects directly among the five HMMT codes or `none`; gold, verifier output, and the answer line remain hidden. (ii) The solve prompt adds *written as plain text with no Markdown emphasis* to the answer-line instruction, because on the previous batch the solver's habit of emitting `**ANSWER: X**` was unreadable to the frozen extractor and pre-empted every other diagnosis with `format_error`. Continuation, closure, experience projection, distiller, validation, and host-instruction rendering are unchanged.

Step 1: one task through the loop (g20). Table 14 follows g20: *What cell components typically found in a eukaryotic cell are missing from a bacterial cell?*, gold `A` (nuclear membrane, histones, mitotic spindle, Golgi, mitochondria, ER); option `H` lists a second, narrower set of membrane-bound compartments.

Stage	Record
Initial segment	Checks each option, concludes *both A and H are technically correct*, then picks A as *more comprehensive*. Two selection criteria (absent from bacteria; typical of eukaryotes) are applied jointly without separating them. Ends `ANSWER: A`. Counters: 748 output tokens, 1 call, 5.5 s, completed early.
Isolated diagnosis	`failure_class=conflated_criteria` ⇒ `code=conflated_criteria`, `needs_revision=true`. No admitted rule matches; generic instruction, `required_checks=[]`.
Same-thread revision	Separates the criteria: H's compartments are not universal in eukaryotes (some lack vacuoles or peroxisomes), so A alone satisfies *typically found*. Ends `ANSWER: A`.
Typed closure	`resolved=true`, `new_contradiction=false`, `format_valid=true`, four checks. Revised answer replaces the initial one.
Verifier	`correct`, `reward=1`, `invalid=0`.
Experience row	`diagnosis_code=conflated_criteria`, `revision_executed=true`, `closure_verified=true`, `closure_failure=none`, `reward=1`, `capabilities=[biological_reasoning, cell_biology, closed_book, exact_answer, microbiology]`.

Table 14: Task-level trace for g20 under π^+.

`rule_id` (suffix)	trigger	support / positive / risk	operations
`separate_criteria-guarded-s1`	`conflated_criteria`	g17 g18 g20 g52 / g20 g52 / g17 g18	`[verify_givens]`
`resolve_conflict-guarded-s2`	`reasoning_contradiction`	g11 g29 g38 g43 g46 g51 g55 / g11 g38 g43 / g29 g46 g51 g55	`[verify_givens]`
`map_requested_expression-guarded-s0`	`choice_mapping`	g06 g10 g44 / g10 / g06 g44	`[verify_givens]`

Table 15: All candidates emitted for the batch; `rule_id`s carry the prefix `hmmt-v32-i1-`. Scope is `[biological_reasoning, closed_book, exact_answer]` for each; all are `candidate`, `executable=false`, `evidence_level=hypothesis`.

Step 2: batch distillation. Of 60 rows, 16 carry an actionable code: `reasoning_contradiction` 7, `conflated_criteria` 4, `choice_mapping` 3, `missing_derivation` 2; final accuracy is 55/60. Three triggers reach three distinct supporting tasks. For `conflated_criteria` the supporting tasks are g17 (XYY and behaviour), g18 (non-locomotor functions of bone), g20 above, and g52 (*Drosophila* yellow-mutant mating data). Two of them are risk rows: on g17 the revision changed `I` → `H` (the gold) but closure reported a new contradiction, so the frozen procedure kept the initial answer and the task scored incorrect; on g18 the revision restated the same single answer and closure failed, initial answer kept, correct. Because risk rows exist, the distiller skips the `targeted` variant and, with failure mode `new_contradiction`, selects `guarded`, whose program is `[verify_givens]`. The common capability set of the four tasks is the base set, so that is the scope. The same logic yields a `guarded` candidate for the other two triggers. `distill()` returns `status=candidates_generated` with three candidates (Table 15).

Step 3: the `conflated_criteria` METARULE. Verbatim from `distill()`, passing `validate_rule`:


```
{"schema": "meta-cot-memoharness/hmmt-positive-rule/v1",
"rule_id": "hmmt-v32-i1-separate_criteria-guarded-s1",
"polarity": "positive", "trigger": "conflated_criteria",
"dimension": "constraint_handling", "mechanism": "criteria_conflation",
"action": "separate_criteria",
"variant": "guarded", "operations": ["verify_givens"], "required_checks":
["verify_givens"],
"postcondition_version": "public-closure-v2",
"required_capabilities": ["biological_reasoning", "closed_book",
"exact_answer"], "excluded_capabilities": [],
"support_ids": ["g17", "g18", "g20", "g52"], "positive_ids": ["g20", "g52"],
"risk_ids": ["g17", "g18"],
"failure_modes": ["new_contradiction"], "parent_rule_ids": [],
"history_through_iteration": 1,
"status": "candidate", "executable": false, "evidence_level": "hypothesis",
"support_experience_refs": [4 refs], "risk_experience_refs": [2 refs]}
```

When a later task in scope is diagnosed `conflated_criteria`, the compiler injects the rule's `host_instruction` into the continuation prompt and closure must list `verify_givens` among its completed checks:

"First verify the proposed repair premise against the original public givens."

Because all three candidates share the program `[verify_givens]`, they render to the same sentence; they differ in trigger, `required_checks` enforcement at closure, and provenance. The `action` field (`separate_criteria`) is carried in the rule's identity and lifecycle records but is not part of the rendered instruction for the `guarded` variant.

Status. Single-iteration candidate hypotheses; the paired gate, admission, delayed publication, and held-out evaluation were not run. The batch also exhibits the closure boundary at work: on g17 a revision that reached the gold answer was rejected for introducing a new contradiction, and the initial wrong answer was retained, as the procedure specifies.

I AI USE DISCLOSURE

We used generative AI tools in several research and writing subtasks. They assisted with implementation of the MetaRule harness, experiment-launch and receipt-audit scripts, debugging, draft experiment descriptions, literature-search queries, reference formatting, and editing the manuscript for clarity and concision. They also assisted with proposing and refining the transactional procedure-learning framework and with planning and interpreting diagnostic and ablation analyses.

Generative AI tools were not used as the official verifier, as an oracle for hidden answers, or as a substitute for the registered held-out evaluator. They did not receive gold answers in the training-agent, isolated-diagnosis, controller, or MetaRule prompt paths. No synthetic benchmark data were used as reported evaluation outcomes. All experimental results in the paper come from recorded runs and official verifiers; model-generated code and configuration changes were inspected, executed, and checked against receipts, replay traces, and compilation tests. Citation metadata were checked against the cited papers and their original records, and claims were reviewed by the authors against the underlying artifacts.

The authors reviewed and take responsibility for all AI-assisted text, code, figures, tables, interpretations, claims, and supplementary artifacts in this submission. Any remaining error, omission, or misrepresentation is the authors' responsibility.