RECURLENS: TRACING THE EVIDENCE FOR RECURSIVE SELF-IMPROVEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

When an agent improves after revising itself, the final task score does not reveal which part of the revision worked. The new method may never reach a successor, and a stronger descendant may still be worse at producing its next update. We introduce RecurLens, an executable audit for fixed-weight agents that records proposed and installed policies, follows their use across three update opportunities, and tests inherited methods in a common recipient. The audit exposes separations that an endpoint score hides. Sol's Recursive descendant scores 5.38 points above its Frozen counterpart but 17.09 below its own initial state, with no method change. A supplied method reaches 136 successors in a later handoff condition, while failed qualification checks limit the utility inference. In four independent packages, its full downstream contrast falls from 77.72 to 21.98 points when the task policy written by the first update is removed. Thus task gains, actual inheritance, and later improvement utility require separate evidence; the recorded runs establish some links while leaving others unresolved. We will release the benchmark implementation and reproducibility materials after review.

1 INTRODUCTION

An agent can revise the instructions that guide its work while the language model underneath stays fixed. If its next version scores higher, two explanations remain: it may have learned to solve the present task, or it may have learned to produce better revisions. Promptbreeder evolves both task and mutation prompts, STOP evaluates the programs produced by an improved improver, and harness adaptation work distinguishes writing an update from benefiting from it (Fernando et al., 2023; Zelikman et al., 2024; Lin et al., 2026). These studies make the task-versus-improver distinction concrete. In a running agent, however, a saved revision can fail to install, a successor can fail to receive it, and a higher score can arise from a task policy written before the supposed later improvement. We therefore need to observe where each revision goes and what it changes before treating a higher final score as evidence that an agent has become better at improving itself.

We build RecurLens to follow an improvement attempt from proposal to later use. Each run begins from one state and forks into Frozen and Recursive lineages. Both may revise their task policies, but only Recursive can pass a changed improvement method to the next producer. Over three scheduled opportunities, we record the candidate, the installed state, the method received by the successor, and the tests the descendant writes. We then put method snapshots into identical starting recipients and score the task plans produced after two more updates. This shared recipient removes the producer's accumulated task state, although the first diagnostic update can still write a new task policy. Executable transaction and cache faults supply a fixed test setting, and private outcomes remain unavailable to the updater. Figure 1 summarizes the resulting audit.

The audit shows why these links cannot be collapsed. Sol's Recursive branch beats Frozen by 5.38 points but ends 17.09 below its own starting score without changing its method. A supplied reference method reaches successors after an output-contract repair, yet that run does not qualify for a confirmatory utility estimate. Across four task packages, restoring the initial task policy cuts its downstream contrast from 77.72 to 21.98 points. These contrasts show that a higher score alone cannot identify which step in the fixed three-update chain produced an apparent gain.

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.

Benchmark / system	P1 inherit	P2 held-out	P3 T/I split	P4 use I	P5 swap I	P6 evidence	P7 two handoffs	Primary object
Recursive improvement benchmark protocols								
RSI-Exam (RSI-Exam Team, 2026)	✓	✓	✗	✗	✗	✗	✗	executable research methods
RSIBench (RSIBench Team, 2026)	✓	✓	✗	✗	✗	✗	✗	editable agent harness
Closest task-versus-improver and harness studies								
Promptbreeder (Fernando et al., 2023)	✓	✓	✓	✓	✗	✗	✗	task and mutation prompts
STOP (Zelikman et al., 2024)	✓	✓	✓	✓	✗	✗	✗	downstream improver utility
Harness Updating (Lin et al., 2026)	✓	✓	✗	✗	✗	✗	✗	update versus benefit
Self-modifying agent systems								
DGM (Zhang et al., 2026a)	✓	✓	✗	✗	✗	✗	✗	self-modifying coding agent
AIDE^2 (Srikanth et al., 2026)	✓	✓	✗	✗	✗	✗	✗	self-modifying research agent
Stagewise diagnosis of recursive method handoff								
RecurLens (ours)	✓	✗	✓	✓	✓	✓	✓	**method handoff and failure location**

P1: successive revisions inherit changed state; P2: final evaluation uses data or tasks unavailable during selection; P3: task policy T and improvement policy I are separate; P4: the exact installed I is consumed by a successor; P5: only I changes at a common recipient before new update and task calls; P6: missingness and provenance have distinct inference states; P7: G4 requires two qualified consecutive handoffs. A cross means the cited study does not specify that test. RecurLens uses a panel reused during development, hence P2 is absent.

Table 1: Whether related benchmarks and agent systems explicitly test seven properties, from inherited revisions and held-out evaluation to method swapping, evidence states, and repeated handoffs.

This view also changes how we read a failed run. A low descendant score may reflect a harmful update, but it can also follow a rejected output or a successor that never received the proposed method. Conversely, a positive endpoint difference can reflect a better task policy without any improvement in the method used for later revisions. We keep intermediate states and denominators, locating where evidence stops rather than labeling an entire lineage a success or failure.

Our contributions concern the audit protocol and the evidence it produces:

- We trace proposed updates through installation and successor use, so an edited artifact cannot be mistaken for an inherited method that a successor actually uses.
- We pair lineages with a common-recipient continuation to separate task performance from later method utility and expose the remaining task-policy pathway.
- We use natural runs and controls to show that a favorable branch contrast, a completed handoff, and a later score gain warrant different claims about improvement.

2 Evaluation Framework and Protocol

An endpoint score leaves the intervening revisions unobserved. Figure 1 follows five checks instead: local change (G0), method handoff (G1), descendant task performance (G2), later utility in a matched recipient (G3), and persistence after repeated handoffs (G4). T0–T3 are the scheduled states; F and R mark the Frozen and Recursive lineages. Appendix A specifies every transition.

2.1 Task: Writing Tests for Cache Errors

What the agent does. The agent writes a short test plan for a transaction and cache system. Its public input is a graph in which some nodes store integers and other nodes compute values from them. A plan is a sequence of `RESET`, `BEGIN`, `READ`, `WRITE`, `COMMIT`, and `ROLLBACK` operations, starting from a COLD state and using at most 1024 operations. The public model gives the expected result of each read; the private implementation under test may contain a fault. A legal plan detects a fault when its observed reads disagree with the public model. The aim is to expose faults early without reporting a fault on a correct implementation. Direct task tests and G3 continuations use these same rules, so their scores are comparable within this setting.

Which faults the tests can find. The panel contains two fault types. For M1, a cache may retain a value computed inside a transaction that was later rolled back. For example, if a derived value depends on a stored integer, the agent can change the integer, read the derived value, roll back, and read it again. The second read should reflect the committed state; a value left over from the aborted transaction exposes M1. For M2, committing a change may fail to invalidate a dependent cached

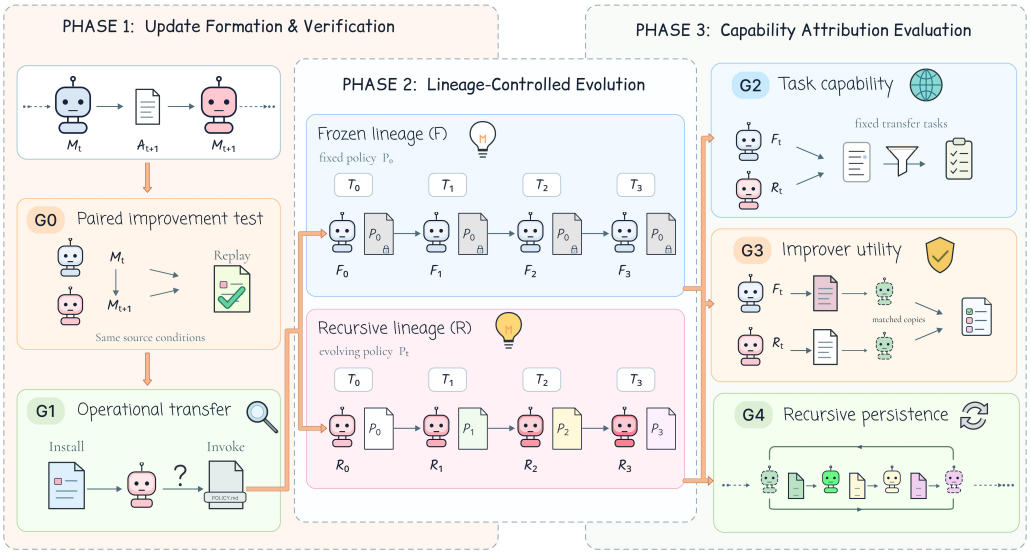

Figure 1: Three phases of the lineage audit across three update opportunities. An installed A_{t+1} produces successor state M_{t+1}; Frozen retains improvement policy P_0, while Recursive may pass a revised P_t to a successor. Both may update task policy. G0 and G1 check update and handoff, G2 scores task behavior, G3 tests later method utility in a matched recipient, and G4 checks persistence across handoffs.

value, which a read after the commit can expose. We also include correct implementations to measure false alarms. The fixed panel contains eight packs, 24 public graphs, and 426 implementations: 237 M1, 165 M2, and 24 correct. These are panel contents, not 426 independent trials of model improvement. SOURCE supplies the examples used for revision, TRANSFER supplies task tests, and G3 has its own SOURCE for the common-recipient comparison. The panel was reused during development, so these results do not establish performance on a fresh task distribution.

How task instances are checked. An independent oracle checks the expected reads, and witness plans verify that each intended fault can be exposed while correct behavior is not flagged. We also check the tested graph collections for structural duplicates and keep the exact graph, implementation, and assignment records for each run (Appendix B). These checks support the intended task execution and scoring. Whether an inherited method improves later updates, rather than merely copying useful task information, is a separate question probed by the G3 controls.

2.2 System State

Figure 1 uses M_t for the runnable agent state and P_t for its improvement policy, which we also call its method. More explicitly, $M_t = (\theta, \pi_t, P_t)$: model weights θ are fixed, the task policy π_t tells the agent how to write a test, and P_t tells it how to revise those instructions after seeing authorized examples. The document A_{t+1} is a proposed update; only a legal installation changes the state to M_{t+1}. At each scheduled opportunity, the model may leave the state unchanged, edit either policy, or edit both. The task rules, update schedule, scoring procedure, and resource budget are fixed for each run.

To propose a revision, UPDATE receives both policies and a fixed replay of authorized examples called SOURCE. To write a test, TASK receives the current task policy and the public description of its target. UPDATE cannot see the target's private fault label, the answers or scores from TRANSFER, the G2/G3 evaluations, or a complete history of its own update effects. Descendants do not gather new SOURCE examples during a lineage. These rules prevent private evaluation scores from silently guiding later revisions, while limiting the feedback available to the agent.

2.3 Question 1: Did the Update Take Effect and Reach a Successor?

A proposed method can fail before the next agent uses it, so G0 records local behavioral change and G1 records actual handoff. Three scheduled updates take the initial state T0 to T3. Frozen (F_t) and Recursive (R_t) begin together and both carry forward their task policy. Frozen keeps the initial improvement method P_0; Recursive passes a revised method only after legal installation.

Every update uses one opportunity. A `NO_CHANGE` or invalid proposal leaves the state as it was; other legal changes take effect under the branch's inheritance rule, even when they later lower the score. We report T3 rather than selecting the best earlier state. The paired improvement test (G0) runs the tests written before and after an update on the same SOURCE implementation. Operational transfer (G1) requires more: the installed method must differ in bytes, the next scheduled MAIN update must receive it, and that successor must return a legal candidate, including `NO_CHANGE`. A change made at the third update has no scheduled successor. This rule lets us identify a real handoff without treating a saved proposal or a byte change as evidence of benefit.

2.4 Question 2: Can the Descendant Still Solve the Task?

Handoff does not tell us whether the descendant still solves the task. G2 runs each scheduled state on the same fixed TRANSFER panel. A legal test receives more credit for finding a hidden fault earlier: if the first mismatch occurs at operation k under budget $B = 1024$, its value is $a = (B-k+1)/B$; a test with no detection receives zero. The task effectiveness score (TES) is 100 times the prespecified weighted mean over targets, fault types, and lineages, on a 0–100 scale. We also report detection, operation cost, scored coverage, and false alarms on correct implementations. The updater sees no private targets or scores.

We compare T3 with T0 to see whether task performance was retained. We also compare the two endpoints:

$$\text{System Gain} = \text{TES}(R_3) - \text{TES}(F_3). \tag{1}$$

System Gain is in TES points. It describes the two realized paths, not the isolated effect of P_t: the branches may develop different task policies, and Frozen may simply deteriorate more than Recursive.

2.5 Question 3: Does the Method Lead to More Useful Later Updates?

G2 can rise even when the method for producing the next revision has not improved. G3 therefore places each method snapshot P_0, P_1, P_2, P_3 in a fresh copy of the same recipient r, with identical diagnostic SOURCE e_d, budget b, targets, and model profile. No copy inherits its producer's task policy. OUTER first uses the supplied method; after a legal installation, ENACTMENT uses whichever method is then in force. The recipient installs a legal second update and writes two downstream TASK plans for scoring. This isolates the starting recipient while allowing the first update to change its task policy.

Let D_θ denote this diagnostic continuation with model weights θ fixed. We score its final task plans with TES. The method-snapshot contrast, denoted RIG, compares the final method with the initial method in the matched recipient:

$$U_t = \text{TES}(D_\theta(P_t; r, e_d, b)), \qquad \text{RIG} = U_3 - U_0. \tag{2}$$

RIG is in TES points. Matching recipients removes the producer's accumulated task state, but OUTER can write a new one. Thus a positive RIG measures downstream total utility, not a pure gain in the ability to update. We probe the task-policy pathway by restoring the recipient's initial task policy after OUTER while keeping its installed method (Appendix F.5). A snapshot can also be tested without a qualified MAIN handoff, but that measures potential utility rather than benefit actually inherited by a successor.

We check this readout against a supplied reference method, independent runs of the same method, method-content changes, and different recipients. These controls test sensitivity, random variation, content, and compatibility. If identical inputs share one diagnostic run, its zero difference is one reused observation, not an independent estimate of variation.

2.6 Question 4: Can Later Utility Be Tested After Repeated Handoffs?

One handoff cannot show persistence. G4 first requires two consecutive qualified G1 handoffs in one lineage. We then examine the scheduled method snapshots after the second handoff, no later than T3. Without both handoffs, G4 is N/A; if no later utility can be scored, it is *not evaluable*. Among evaluable paths, we report three distinct patterns: later utility above the initial U_0, retention of the first observed gain, and improvement at every adjacent step. A plateau can satisfy the first two without continued growth; a decline can remain above U_0 while losing the first gain. These are descriptive finite-horizon patterns, not a single validated persistence verdict. Confirmatory G4 would additionally require a frozen reliability rule and independent later observations, which the archived natural runs do not supply.

An outcome is interpretable only if the required steps ran. We record the instructions shown to the model (V), any public output constraint (C), and the final validator (J) separately. Since output assistance changes how responses are generated, API/native, local/strict, and local/assisted runs remain separate profiles. Invalid outputs, technical failures, UNKNOWN outcomes, and descendants skipped after a SOURCE failure are not scored as task zeros. We report the weighted score denominator and bounds for missing scores; these bounds describe what the missing values could change, not sampling uncertainty. Model calls and execution costs are tracked separately from the operations inside a test plan (Appendix D).

3 Experiments

We run the same lineage audit in each registered model condition. Starting from one state, we follow three scheduled updates in Frozen and Recursive branches, score the resulting task plans, and test available method snapshots in a matched recipient. API and local profiles use different model conditions or panels, so their scores are compared within a condition rather than pooled. Table 2 summarizes the observed links; Appendix H gives the intermediate states and incomplete endpoints.

Model	**Initial** T0 TES	**G0** +/-/=	**G1** edit/handoff	**G2** R3-F3 / R3-T0	**G3** RIG / status	**G4** eligible/evaluable
Sol	94.93	4/7/5	0/0	+5.38 / -17.09	0/0	0/0
Astra	93.53	8/0/7	0/0	+1.00 / -2.03	0/0	0/0
Gemini Flash	60.21	1/4/59	0/0	-0.22 / -6.02	0/0	0/0
Luna	60.30	25/20/3	9/8	-1.69 / -12.45	-0.247	0/0
Terra	75.70	17/15/32	4/3	-1.30 / -18.45	+1.619	0/0
Llama-4-Scout	0	0/0/52	7/4	0/0	0/0	1/0
Qwen3.8-27B	0	0/0/64	0/0	0/0	0/0	0/0
Qwen3-8B	0	0/0/40	0/0	0/0	0/0	0/0
Qwen3-1.7B	0	0/0/0	3/1	0/0	0/0	0/0
Llama-3.1-8B	0	0/0/14	0/0	0/0	0/0	0/0

Table 2: Observed initial task scores and G0–G4 readouts by model. G2 contrasts use endpoint task scores; Table 4 reports their scored-cell coverage.

A changed method may reach a successor, but many do not. Sol, Astra, and Gemini have local comparisons without a qualified changed-method handoff. Luna records nine method edits and eight handoffs; Terra records four and three; Scout records seven and four. The 136-attempt reference check shows why we track delivery separately: a different method was installed in six of eight blocks, but dependent failures and missing TASK calls blocked the planned utility estimate. Table 3 gives each request role's outcome. In another local condition, all 16 terminal lineages completed and six of eight Recursive terminal methods differed from P_0. These observations establish production and some actual handoffs under their conditions, not benefit or a strict repeat of the earlier Terra result.

Request role	Attempts	Observed readout	Remaining issue
SOURCE	8	8 returned	One returned plan was invalid.
OUTER	16	11 accepted	One rejected; four context failures.
ENACTMENT	32	32 accepted	None recorded.
TASK	80	70 strictly valid	Eight invalid; two externally unknown.

Table 3: Completion and validity at SOURCE, OUTER, ENACTMENT, and TASK in the 136-attempt reference-method handoff check. Each role has its own denominator.

Model	T0	F3	R3	T0 cells	R3 cells	R3–F3	R3–T0
gpt-5.6-sol	94.93	72.46	77.84	108/108	108/108	$+5.38$	-17.09
gpt-6-astra	93.53	90.50	91.50	108/108	108/108	$+1.00$	-2.03
gemini-3.8-flash	60.21	54.41	54.19	548/548	548/548	-0.22	-6.02
gpt-5.6-luna	60.30	43.97	42.28	548/548	436/548	-1.69	-12.45
gpt-5.6-terra	75.70	58.55	57.25	548/548	548/548	-1.30	-18.45
Llama-4-Scout	0.00	0.00	0.00	502/548	438/548	0.00	0.00
Qwen3.8-27B*	0.00	0.00	0.00	548/548	548/548	0.00	0.00
Qwen3-8B	0.00	0.00	0.00	424/548	334/548	0.00	0.00
Qwen3-1.7B	0.00	0.00	0.00	23/548	12/548	0.00	0.00
Llama-3.1-8B	0.00	0.00	0.00	77/548	28/548	0.00	0.00

* Qwen3.8-27B used an older decoder flagged for descendant-slot impact review.

Table 4: Observed task TES at T0, F3, and R3, with scored-cell coverage and endpoint contrasts. Incomplete rows are not full-panel estimates.

A favorable branch difference can conceal task decline. Sol's Recursive T3 scores 5.38 TES points above Frozen T3 but 17.09 below its own T0, without changing its method. Gemini and Terra also end below T0 by 6.02 and 18.45 points on complete panels, while Luna's missing-score bound stays negative even at its favorable extreme. Table 4 shows observed endpoints and their scored-cell coverage. Zero task scores need their denominators: Qwen3.8-27B scores zero on all 548 T0 cells, Scout on 502, and Qwen3-1.7B on only 23. The other cells may be missing rather than zero. On eight separately frozen structural targets, a public fixed rule scores 9.941/83.542 TES for M1/M2, and a finite search scores 96.064/74.445; neither flags a correct world among the eight tested. These controls show that the fault families can be detected within the task budget, although they neither bound the model scores nor validate a new task distribution. Intermediate scores, detection, operation cost, and missingness appear in Appendix H.3 and Appendix H. The main panel was reused during development.

The later-utility readout depends on both delivery and variation. In a guided fixed-panel calibration, a supplied reference method scores 86.01 M1 TES, compared with 20.14 for the base method and zero for an ablated method. All 16 reference TASK plans use the intended rollback-and-read pattern and detect M1; the base arm detects eight faults, while the ablated arm detects none. This is a behavioral check under instructed application, without the full OUTER step (Appendix F.4). An early full-chain comparison cannot test the same claim: all 576 reference OUTER responses failed strict JSON parsing, so P_0 remained installed. Its -2.297 TES contrast is a pipeline result. Independent same-method chains differ by -3.047 TES, with a 95% interval of $[-16.361, 10.266]$ that does not establish agreement within five points. Table 5 also retains content contrasts. Appendix Table 20 shows that their acceptance rates vary: FULL ENACTMENT passes 0/348 requests, and principle-only passes 73/348 versus 207/348 for neither. Those differences prevent a pure content attribution.

After the output contract was repaired, the reference method was installed in 144/144 OUTER chains and read by 136/144 successors. Its all-block fixed-panel identification bound is $[10.142, 19.864]$ TES. Yet the reference-minus-initial mean among complete blocks is $+10.824$ on the 40GB deployment and $+26.220$ on the 80GB deployment. Independent same-method three-opportunity chains also differ by $+17.140$ TES across eight blocks, seven on 40GB and one on 80GB. That null comparison and the reference bound have different estimands and cannot simply be subtracted; together they show why a positive delivered contrast is not yet a stable method effect. Failed qualification gates, 69 missing raw outcomes, and the deployment change prevent a confirmatory estimate (Appendix I.4).

Comparison	Units	Effect (TES)
Reference vs. initial	288/288	-2.297
Same method	32/32	-3.047
Full vs. broken method	348/348	-2.939
Principle vs. neither	348/348	$+5.150$
Structural targets	32/32	$+6.235$

Table 5: Early full-chain and method-content TES contrasts, with complete units for each comparison.

M1 package	Full	Task policy restored
1	90.33	0.00
2	87.92	0.00
3	71.02	39.26
4	61.61	48.66
Mean	77.72	21.98

Table 6: Reference-minus-base TES in four independent packages, with the OUTER task policy retained or restored.

Comparison	ΔTES
Reference vs. initial	$+8.3$
Harmful vs. initial	-5.7
Independent initial A vs. B	$+0.7$
Skip ENACTMENT	$+2.9$
Restore task policy	$+5.4$
Restore task policy; fix method	$+1.0$
Restored vs. fixed method	$+4.4$

Table 7: Method controls and pathway contrasts in the later batch. Each comparison uses 48 blocks.

Comparison	ΔTES
New target: reference vs. initial	$+4.8$
Fixed replay: T3 vs. T0	-1.8
Feedback: T3 vs. T0	$+2.5$
Feedback vs. fixed replay	$+4.3$
Principle only vs. initial	$+4.1$
Full vs. source facts only	$+3.6$
Task plus G3 vs. task only	$+3.1$

Table 8: Selected TES contrasts for target transfer, feedback, method text, and G3-based candidate choice.

A positive downstream contrast need not come from the later method. In four independent task packages, the reference-minus-base contrast averages 77.72 TES when the task policy written by OUTER is retained. Restoring the common initial task policy reduces it to 21.98 TES. Package-bootstrap 95% intervals are $[66.32, 89.13]$ and $[0, 43.96]$ respectively, while exact two-sided tests give $p = 0.125$ and $p = 0.5$. The small number of independent packages matters even for the large full contrast. Table 6 shows each package beside the early controls. The full contrast includes an earlier task-policy pathway; the restored contrast is positive in only two packages and does not identify a stable gain in improvement-method ability.

A later author-checked batch probes that pathway more directly. Its raw responses and scoring files have not yet been attached to the replayable archive, so Table 7 reports point contrasts without intervals. Across 48-block comparisons, the reference exceeds the initial method by 8.3 points, a harmful method scores 5.7 points lower, and two independent initial-method chains differ by 0.7 points. Skipping ENACTMENT leaves a 2.9-point reference contrast; restoring the initial task policy leaves 5.4 points; fixing the later method as well leaves 1.0 point. The reported paired contrast between the last two conditions is 4.4 points. These readings suggest method-content sensitivity, but the intervention records are needed to establish that the final contrast changes only the later method. One same-method difference cannot define a general noise bound.

The same batch probes whether the finding survives changes in targets and protocol. A new topology and fault family yield a 4.8-point reference contrast on 40 blocks. Across 32 lineages, development feedback and selection score 4.3 points above fixed replay in a paired comparison. On 40 blocks not used to write the methods, full text exceeds source facts alone by 3.6 points. Task-only and task-plus-G3 rules choose different candidates in eight of 24 pools and differ by 3.1 points on independent

Model	Actual calls	Scored	Valid	Invalid	Skip	Shared	Unknown
gemini-3.8-flash	487	345	131	11	9	384	0
gpt-5.6-luna	481	321	150	10	159	240	0
gpt-5.6-terra	528	368	160	0	0	352	0
Llama-4-Scout	517	292	140	85	35	328	0
Qwen3.8-27B	496	352	144	0	0	384	0
Qwen3-8B	328	204	90	34	312	240	0
Qwen3-1.7B	136	20	21	95	696	48	0
Llama-3.1-8B	136	37	0	99	648	96	0

Table 9: Physical calls and scoring, validity, skip, sharing, and unknown statuses for the 880-slot model conditions shown.

later evaluation. Table 8 selects these and the relevant baselines; Appendix Table 24 expands the target, feedback, method-text, and selection comparisons with their units. Without archived raw records or calculated intervals, these point readings do not establish broad transfer, a pure principle effect, or a stable selection advantage.

Natural later gains and persistence remain unresolved. Scout's method contrast is zero on 330/548 paired cells, Luna's bound crosses zero, and Terra's historical $+1.619$ TES lacks independent response provenance. Other scored API profiles and two local Qwen models have unchanged methods; a further $+33.528$ TES contrast occurs without a method change. The archived fixed-pool selection check also chooses the same candidate under all three rules (Appendix Table 21). Terra's original-condition repeat is blocked by duplicate response IDs, and Appendix H.5 gives the P_0–P_3 diagnostic readings.

Only one archived Scout chain reaches the two-handoff G4 entry rule, and it has no evaluable later utility. In the later author-checked batch, seven of 24 natural lineages have two substantive handoffs; five have evaluable utility, of which three stay above the starting score, two retain the first gain, and one improves at every adjacent step. The smaller counts express stronger patterns, not successive estimates of one success rate. Their underlying records are needed for replay and no natural G4 verdict is confirmed. The constructed fixture in Appendix H.6 separates a plateau, growth, and decline above the starting score. Table 9 records scored, invalid, skipped, shared, and unknown slots for the listed 880-slot conditions. Independent offline replay matches 6,539 task scores and 146,629 world results from 11,203 stored raw requests, verifying recorded arithmetic rather than delivery or response independence.

Across the recorded conditions, an update can change behavior, reach a successor, or raise a later task score without establishing the next link in the chain. RecurLens locates those distinctions. The additional point contrasts are suggestive, while the archived natural runs still do not establish a reliable gain in producing later improvements or an evaluable persistence result.

4 Discussion and Limitations

The cases differ in where the evidence stops. Sol has a favorable branch contrast without an improved endpoint or a changed method. Luna and Scout pass edited methods forward, but their later benefit remains unverified. An early reference comparison fails at strict parsing before the proposed method is installed; a later interface delivers the reference but misses its qualification conditions. Treating all four outcomes as one success or failure rate would discard the distinction the audit is designed to preserve.

These records also show what the additional checks contribute to interpretation. An endpoint-only reading would call Sol's favorable Recursive–Frozen difference an improvement, whereas its T0 comparison and unchanged method rule out the stronger inference. The large common-recipient reference contrast without a task-policy clamp could similarly be read as a better improver; restoring the initial task policy removes much of it. The audit therefore changes the conclusion drawn from observed trajectories. Its separate value for selecting future methods remains unconfirmed by

the archived candidate pool, and the later reported selection contrast still needs its raw evaluation records.

These findings describe a fixed-weight, fixed-replay protocol on a reused panel. Task policies can change along each lineage and during the common-recipient continuation, so a positive G3 contrast is downstream total utility unless further controls isolate the method pathway. Same-method variation, missing outcomes, restricted feedback, and uncertain response origins limit effect claims. The archived candidate pool also cannot show added selection value from G3 because every rule chose the same method; a later author-checked pool contains different choices but lacks replayable raw Y records. Offline replay verifies stored arithmetic, while method delivery and independent generation require their own records. The audit localizes the unsupported links in these runs without settling how another verified interface or feedback regime would behave.

5 Related Work

Artifacts and task contracts. Prior benchmarks assess modified training source, engineering artifacts, discovered agent code, model-produced submissions, or data strategies under fixed scoring procedures (Chi et al., 2026b;a; Hu et al., 2025; Chan et al., 2025; Meng et al., 2026). Procedural support, recombined rules, and controlled task access further shape measured adaptation (Gao et al., 2026; Zhou et al., 2026; Bragg et al., 2026). These designs motivate the fixed task and information contract in RecurLens; our evaluation also follows the recipient that uses an inherited method to produce another update.

Retention and realized benefit. Harness adaptation already separates a system's ability to write useful updates from its ability to activate and benefit from them during task solving (Lin et al., 2026). Saved experience can be retrieved in later sessions, while longitudinal and within-episode evaluations measure retention or refinement (Xue et al., 2026; Deng et al., 2026; Zhang et al., 2026b). Outer self-edit learning and inner acquisition can likewise diverge (Zweiger et al., 2025). RecurLens follows the installed method into the next update producer and records when that handoff or its later outcome cannot be evaluated.

Recursive modification. Promptbreeder evolves both task prompts and the mutation prompts that revise them (Fernando et al., 2023). STOP evaluates an improved improver by the quality of programs it produces on downstream tasks (Zelikman et al., 2024); recursive policy changes and executable agent descendants provide further examples (Yin et al., 2025; Zhang et al., 2026a). The task-versus-improver distinction and downstream meta-utility therefore precede this work. RecurLens adds an audit of one realized lineage, linking exact installation and successor consumption to task outcomes while keeping missing or dependent diagnostic results visible. Its common-recipient contrast measures two-step total utility; attribution to the inherited method requires the additional controls discussed in Section 3.

6 Conclusion

A descendant that solves a task better need not be better at producing the next improvement. RecurLens tests that distinction by following each revision from proposal through installation and successor use, then measuring task performance and downstream utility under matched starting conditions. The recorded runs show task decline despite a favorable branch contrast, real handoffs without verified natural benefit, and a supplied reference whose interpretation changes with delivery and task-policy controls. Within this fixed panel and finite horizon, the result is a more precise account of what an apparent improvement chain does and does not establish.

AI use statement

Generative AI assisted methodological review, interpretation of archived observations, writing and revision, reference formatting, and preparation of figures, tables, and build scripts. An independent offline scoring replay checked stored closure artifacts without making new model requests. This assistance did not conduct the reported model experiments. The authors are responsible for checking the manuscript, its scientific claims, and the final disclosure before submission.

Ethics statement

The benchmark uses bounded executable software tests with separate private scoring and explicit validity checks. An apparent score gain is not treated as sufficient evidence of a useful improvement method. The planned release will exclude credentials and will be checked against task-data permissions and provider restrictions on raw responses.

Reproducibility statement

Sections 2 and 3 define the reported comparisons; Appendix D describes the required records. After review, we plan to release the benchmark runner, task construction and scoring code, exact aggregation weights, prompts and output schemas, condition-level configurations, and replay instructions. Permitted records will include a complete traced lineage and the data needed to recompute its scores. The current manuscript source package contains LaTeX sources and the compiled paper, not that experimental release; the later manually checked contrasts await attachment of their raw records.

References

Jonathan Bragg, Mike D'Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi, Sergey Feldman, Dany Haddad, Jena D. Hwang, Peter Jansen, Varsha Kishore, Bodhisattwa Prasad Majumder, Aakanksha Naik, Sigal Rahamimov, Kyle Richardson, Amanpreet Singh, Harshit Surana, Aryeh Tiktinsky, Rosni Vasu, Guy Wiener, Chloe Anastasiades, Stefan Candra, Jason Dunkelberger, Dan Emery, Rob Evans, Malachi Hamada, Regan Huff, Rodney Kinney, Matt Latzke, Jaron Lochner, Ruben Lozano-Aguilera, Cecile Nguyen, Smita Rao, Amber Tanaka, Brooke Vlahos, Peter Clark, Doug Downey, Yoav Goldberg, Ashish Sabharwal, and Daniel S. Weld. AstaBench: Rigorous benchmarking of AI agents with a scientific research suite. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.21652v2.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095v6.

Yizhe Chi, Deyao Hong, Dapeng Jiang, Tianwei Luo, Kaisen Yang, Boshi Zhang, Zhe Cao, Xiaoyan Fan, Bingxiang He, Han Hao, Weiyang Jin, Dianqiao Lei, Qingle Liu, Houde Qian, Bowen Wang, Situ Wang, Youjie Zheng, Yifan Zhou, Calvin Xiao, Eren Cai, and Qinhuai Na. FrontierEng: Benchmarking Self-Evolving Agents on Real-World Engineering Tasks with Generative Optimization, 2026a. URL https://arxiv.org/abs/2604.12290v2.

Yizhe Chi, Wenyi Li, Deyao Hong, Xiaoqiu Wang, Mingju Gao, Kaisen Yang, Bingxiang He, Youjie Zheng, Calvin Xiao, and Qinhuai Na. AI4AI-Bench: Benchmarking LLM Agents in Algorithmic Design for Recursive Self-Improvement, 2026b. URL https://arxiv.org/abs/2608.20318v1.

Bo Deng, Kang Zhou, Lifan Guo, Chongyang Tao, Xuanren Chen, Chenggang Xie, Renzhao Liang, Feng Chen, and Chi Zhang. FinEvo-Bench: A Longitudinal Benchmark for Self-Evolving Agents in Professional Financial Workflows, 2026. URL https://arxiv.org/abs/2608.06144v1.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-Referential Self-Improvement Via Prompt Evolution, 2023. URL https://arxiv.org/abs/2309.16797v1.

Xingze Gao, Chuanrui Hu, Hongda Chen, Pengfei Yao, Zhao Wang, Yi Bai, Zhengwei Wu, Yunyun Han, Xiaofeng Cong, Jie Gui, Yafeng Deng, and Teng Li. EvoAgentBench: Benchmarking Agent Self-Evolution via Ability Transfer, 2026. URL https://arxiv.org/abs/2607.05202v1.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.08435v2.

Minhua Lin, Juncheng Wu, Zijun Wang, Zhan Shi, Yisi Sang, Bing He, Zewen Liu, Tianxin Wei, Zongyu Wu, Zhiwei Zhang, Dakuo Wang, Xiang Zhang, Benoit Dumoulin, Cihang Xie, Yuyin Zhou, Suhang Wang, and Hanqing Lu. Harness Updating Is Not Harness Benefit: Disentangling Evolution Capabilities in Self-Evolving LLM Agents, 2026. URL https://arxiv.org/abs/2605.30621v1.

Fanqing Meng, Lingxiao Du, Qiguang Chen, Ziqi Zhao, Haocheng Lu, Mengkang Hu, and Michael Qizhe Shieh. RSIBench-Data: Benchmarking Data-Centric Research for Recursive Self-Improvement, 2026. URL https://arxiv.org/abs/2607.25886v1.

RSI-Exam Team. RSI-Exam: Benchmarking Recursive Self-Improvement through Executable Research. Project protocol, 2026. URL https://rsi-exam.ai/blog.html.

RSIBench Team. RSIBench. Project protocol, 2026. URL https://recursive-self-improvement-bench.com/docs/.

Dhruv Srikanth, Bingchen Zhao, Dixing Xu, Yuxiang Wu, and Zhengyao Jiang. Recursive Self-Improvement of AI Research Agents, 2026. URL https://arxiv.org/abs/2609.26457v1.

Shuhan Xue, Zixin Ding, Yichen Shen, Yinjie Wang, Zhenfei Yin, Yingcheng Wu, Yuxin Chen, Mengdi Wang, and Ling Yang. PAST-Bench: Benchmarking the Foundations of Recursive Self-Improvement in Personal Agents, 2026. URL https://arxiv.org/abs/2608.04003v1.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel Agent: A Self-Referential Agent Framework for Recursively Self-Improvement, 2025. URL https://arxiv.org/abs/2410.04444v4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2310.02304v3.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. In *International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2505.22954v3.

Wentao Zhang, Jianfeng Wang, Liheng Liang, Yilei Zhao, HaiBin Wen, and Zhe Zhao. EvoCodeBench: A Human-Performance Benchmark for Self-Evolving LLM-Driven Coding Systems, 2026b. URL https://arxiv.org/abs/2602.10171v1.

Leijun Zhou, Zhihao Liu, Xiang Qu, Chenxu Liu, Yifei Liu, Yanke Yu, Jingzhe Xu, Xuejun Wu, Buyue Qian, Xi Chen, Yaowei Zheng, and Junhao Hu. GDPevo: Evaluating Agent Self-Evolution on Real Business Tasks, 2026. URL https://arxiv.org/abs/2608.03764v1.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-Adapting Language Models, 2025. URL https://arxiv.org/abs/2506.10943v2.

A Benchmark Protocol and Formal Definitions

A.1 Problem Formulation

The inferential object is the improvement procedure carried by a descendant, rather than the occurrence of a successful task output in isolation. An observed success is an outcome under a particular task, policy state, and generation condition. It does not identify a change in the probability of success without an appropriate comparison, and it does not identify which component of the system produced that change. Even a well-supported task gain leaves open whether the recipient can generate a more useful subsequent update. The distinctions below specify the evidence needed to move between these questions without treating them as interchangeable outcomes.

Let ℓ index a registered lineage and let $b \in \{F, R\}$ identify its branch. Using Figure 1's notation, the runnable state is $M^b_{\ell,t} = (\theta, \pi^b_{\ell,t}, P^b_{\ell,t})$, with fixed model weights θ, an editable task policy π, and an editable improvement policy P. The archived diagnostic files call the method snapshots I0 through I3; these correspond to P_0 through P_3, not to another policy. A task artifact is the executable output generated using the task policy; it is not identical to that policy. The proposed update A_{t+1} can change either policy, but only the legally installed state can govern subsequent work. The record must therefore distinguish the candidate, the installation decision, and the effective state actually supplied to the next producer.

G0 through G4 are evidence dimensions attached to this lineage. Their numbering does not define a sequence of five generation stages or imply that task effectiveness entails improver utility. The permitted inference depends on the observation and its comparator. A failed measurement also differs from an evaluated negative outcome: the former leaves a question unresolved, whereas the latter can constrain the claim under the conditions in which it was measured.

A.2 Evaluation Dimensions and Operational Criteria

G0: Local update effect. A local comparison evaluates parent and child task artifacts on the same currently authorized SOURCE implementation. The evidence consists of both artifact identities, their common implementation and execution conditions, and their evaluable outcomes. An observed difference is interpretable only after these bindings agree. Subtracting results obtained on different adjacent implementations would confound the update with a change in the test. A policy-level `NO_CHANGE` does not guarantee identical TASK outputs when those artifacts are generated separately. The local record retains both facts rather than reclassifying sampling variation as a policy revision. No additional aggregate threshold for G0 is defined here. An absent or negative local effect cannot establish that every possible descendant method is ineffective, and a positive local effect cannot establish realized method inheritance.

G1: Realized improver inheritance. This dimension identifies whether a changed method actually participates in a subsequent MAIN update. For an opportunity with an observed successor, write

$$H^b_{\ell,t} = \mathbf{1}\{I^b_{\ell,t} \wedge C^b_{\ell,t} \wedge L^b_{\ell,t+1}\}, \tag{3}$$

where $I^b_{\ell,t}$ denotes a legally installed method whose bytes differ from the parent, $C^b_{\ell,t}$ denotes exact consumption of that method in the next MAIN input, and $L^b_{\ell,t+1}$ denotes a legal canonical successor candidate. This notation restates the protocol's qualified-handoff rule. It is applied only when the necessary records exist; an unresolved input or decision is not silently converted into $H^b_{\ell,t} = 0$. A legal successor `NO_CHANGE` satisfies the last predicate. Consumption inside a G3 diagnostic does not satisfy the MAIN-consumption predicate.

The required evidence binds the installed bytes to the next request and its candidate decision. Installation without that request cannot establish realized use. A method produced at the final opportunity has no scheduled MAIN successor, which is an observation limit rather than a failed handoff. Consecutive depth is measured within one lineage. Byte changes and content classes are reported together, because whitespace or a paraphrase can satisfy an exact-change predicate without establishing a substantive innovation. G1 establishes neither a task gain nor greater future-update utility.

G2: Direct task effectiveness. Each scheduled task policy is evaluated through its generated plans on the predetermined TRANSFER panel. Task effectiveness, detection, operation use, and coverage are bound to that state and its targets; CORRECT false positives remain separate. The operational measurement requires legal executable plans and the corresponding external evaluations, not merely a policy text or an explanation that a fault should be detectable. A legal non-detection contributes the score defined in Appendix E. An illegal or unavailable execution contributes a missingness state instead. A G2 result characterizes direct task behavior under the registered conditions, without establishing that the same policy state is a better producer of improvements.

G3: Common-recipient future-update utility. The diagnostic binds a method snapshot to the same recipient initialization, diagnostic SOURCE experience, resource contract, model profile, and downstream targets used for its comparator. Its record must preserve two installations. OUTER first generates a candidate under the tested snapshot; after that candidate is installed in the isolated recipient, ENACTMENT uses the actual effective method of the resulting state to generate a further candidate. The second installation determines the state that produces the two DOWNSTREAM task artifacts. Reusing the originally tested snapshot unconditionally during ENACTMENT would evaluate a different procedure whenever OUTER changed the effective method.

The resulting utility is U_t in Equation 2. A complete diagnostic record establishes the snapshot identity, OUTER candidate and decision, post-OUTER state, actual ENACTMENT input and candidate, final installed state, and downstream executions. An evaluable proposal-only diagnostic can establish potential utility even when no qualified MAIN handoff occurred; it cannot establish that this utility was realized in the original lineage. Conversely, realized MAIN consumption without an evaluable diagnostic leaves future-update utility unresolved. Common recipients constrain the comparison but do not identify a complete causal decomposition of the main system trajectory.

G4: Finite-horizon persistence. Eligibility requires at least two consecutive qualified MAIN handoffs within the same lineage. In the three-opportunity protocol, only scheduled method snapshots after the second handoff and no later than T3 can supply a later observation. An ineligible lineage is N/A; an eligible lineage without a scored later snapshot is not evaluable. For scored paths, we describe three different patterns: utility above U_0, retention of the first observed gain, and improvement between adjacent snapshots. None follows from the handoff count alone. The archived fixture's positive-pattern label checks the first pattern under a separate reliability rule, but the complete rule and its inputs are not supplied in this manuscript. We therefore report these patterns descriptively and assign no confirmed natural G4 success. The later manually checked lineages remain separate until their records can be replayed.

A.3 Temporal Evaluation and Information Visibility

T0 is the initial policy state. T1, T2, and T3 are the states after the first, second, and third scheduled MAIN opportunities, respectively. They are not labels for transfer and recursive evaluation phases. G2 can be measured at the scheduled states, and G3 concerns the corresponding method snapshots under its own common-recipient contract. This distinction prevents a diagnostic execution from being mistaken for an extra generation of the main lineage.

The first opportunity uses the protocol's shared initial-input contract. Shared input does not establish that separately sampled outputs are identical; any actual response sharing must be explicitly recorded. Subsequent UPDATE inputs contain the current policies and authorized SOURCE experience. TASK inputs contain the current task policy and public target information. Private fault labels, downstream G2 and G3 outcomes, and a complete self-effect history are excluded from UPDATE. SOURCE observations follow the predetermined replay schedule rather than being recollected online under each descendant policy.

Private evaluation is observational with respect to MAIN state selection. Its scores neither authorize promotion nor trigger rollback. A legal candidate is installed even if its measured utility is poor, and legal `NO_CHANGE` consumes the opportunity while retaining the state. An invalid candidate also consumes the opportunity and retains its parent, without human repair or resampling. SOURCE failure blocks only nodes that require its experience; T0 evaluation is independent of SOURCE success, and a G0 or G2 measurement failure does not by itself invalidate the main state. The registered endpoint is T3, with no additional opportunity selected to rescue an unfavorable trajectory.

A.4 Frozen and Recursive Lineage Construction

The two paths begin at the same declared initialization:

$$M^F_{\ell,0} \longrightarrow M^F_{\ell,1} \longrightarrow M^F_{\ell,2} \longrightarrow M^F_{\ell,3},$$
$$M^R_{\ell,0} \longrightarrow M^R_{\ell,1} \longrightarrow M^R_{\ell,2} \longrightarrow M^R_{\ell,3}. \tag{4}$$

On F, the task policy is inherited but the effective improvement policy remains P_0. Method proposals from that branch are retained as proposals without becoming its successor's effective method. R inherits its task policy and whichever method was legally installed. Both branches follow the same candidate legality rules and registered comparison conditions. This construction isolates the designated inheritance rule; it does not keep every realized state identical, nor does it exclude strategic information from the task-policy channel.

Pairing makes the endpoint contrast interpretable relative to a particular initial state and condition. The records must retain that pairing, rather than assembling a favorable R endpoint and an unrelated F endpoint after the fact. A positive System Gain can coexist with deterioration from T0 or an unchanged improvement method. Its attribution therefore depends on the state and handoff records and remains distinct from the common-recipient method comparison.

Identical G3 executions can be reused only under the specified equivalence rule. Once snapshots exist, and before the first relevant request, the method and all recipient, source, budget, profile, output-contract, and target bindings are frozen into an equivalence map. A shared diagnostic represents one observation reused under that map; its zero contrast is not a pair of independent samples proving equivalence. Figure 1 is a conceptual schematic; the state definitions and installation sequence here determine the executable protocol.

B Benchmark Construction and Information Isolation

B.1 Task and World Construction

The executable task separates a public representation from the private implementation whose behavior is tested. Public integer dependency graphs and transaction semantics define what an action sequence should do, whereas registered implementation variants determine whether execution conforms to those semantics. The public graph contains no private fault label, although repeated exposure could still reveal regularities of the task family. The COLD starting condition and PUBLIC_MODEL belong to the task contract rather than to a model-specific adaptation.

The current author-specified panel contains eight packs and 24 public graphs. Its 426 registered implementations comprise 237 M1 variants, 165 M2 variants, and 24 CORRECT instances. M1 concerns restoration of speculative cache state after rollback; M2 concerns dependency invalidation following commit. Correct implementations make false alarms observable. These counts describe variants of a structured dataset, not independent draws of improvement ability. The number of private implementations does not determine the weight of a graph in an aggregate score.

The author-provided construction description specifies a program oracle and positive and negative witnesses for validating task instances. The manuscript source package does not contain the generation code, graph sampling distribution, fixture identities, or complete representation schema. Mechanism counts cannot recover these choices. Exact reproduction requires the construction manifest, including graph-to-pack membership and implementation bindings; a new generator chosen to match the reported totals would define a different benchmark.

B.2 Dataset Split and Freeze Protocol

Design choices, validation exposure, and formal evaluation have different roles in the evidential history of a benchmark. During design, tasks and representations can be adjusted to instantiate the intended behavior. Validation can reveal malformed instances or an insensitive measurement. A target freeze then identifies the exact task and evaluator artifacts used for a specified comparison; subsequent evaluation must be interpreted relative to that identity. This ordering describes the distinction required for a confirmatory claim, not an assertion that an unexposed final split has already been supplied.

The available panel has been reused during development and review. Freezing it before a corrected execution can stabilize that comparison, but cannot undo prior exposure or convert it into a fresh holdout. The frozen identities must cover the public representations, private variants, SOURCE schedule, TRANSFER bindings, diagnostic source, recipient initialization, and scoring rule. A changed binding may change the scientific condition even when the public graph files themselves are unchanged.

SOURCE provides authorized observations through predetermined replay. TASK receives its current task policy and public target information; UPDATE receives the current policies and authorized experience. Private fault labels and downstream evaluation results are excluded from update inputs. This restriction prevents the evaluation from becoming an undeclared optimization signal. It does not certify the absence of all contamination, which also depends on the construction and exposure history. Exact payloads and target bindings are needed to establish that the intended isolation was realized.

B.3 Structural Uniqueness and Leakage Audit

A structural audit asks whether nominally different targets repeat the same problem under superficial changes. Non-isomorphism checks can rule out graph equivalence under the relation that was tested, while duplicate detection can identify exact or representation-level reuse. Neither establishes novelty relative to every prior task, nor does either show that the measured method generalizes beyond the checked family. The author-supplied description reports non-isomorphism within the panel and relative to an available historical collection; the collection identity and check outputs are needed to delimit that claim.

Artifact hashes address a complementary question: whether a target or evaluator changed after it was fixed for a comparison. Equality of hashes establishes byte identity, not the correctness of the generator or the absence of semantic overlap. A scientifically interpretable leakage audit therefore connects identity checks with the relevant comparison set and the information actually visible to the model. The local paper package lacks the underlying fixtures and audit outputs, so it supports a description of these construction checks without an independent certification that they all passed.

C Runtime System and Action Interface

C.1 Action Representation

An executable action represents an operation together with the fields that determine its arguments. A syntactically well-formed object is therefore insufficient: the requested operation must exist in the public vocabulary, its required fields must be present, and its arguments must satisfy the public semantics. For example, a `READ` action without its required node does not specify an executable read. Accepting that object under a permissive structural schema would not make the missing argument scientifically immaterial.

The model-visible contract V, generation-side structural constraint C, and semantic validator J play different roles. The visible contract states what the model is asked to produce. A structural constraint can restrict the outputs available during generation, whereas the validator determines whether a completed candidate has the required meaning. Preserving J while changing V does not isolate the effect of C. Likewise, increasing the proportion of schema-valid responses does not establish an improvement in task utility, because structural constraints may alter which substantive candidates are sampled. The field-level schema and full messages remain required artifacts; the source package does not contain a verified substitute for them.

C.2 Execution and Evaluation

The evidence path begins with the retained model response. Parsing determines whether a candidate representation can be recovered; structural validation checks its shape and declared types; semantic validation determines whether the proposed operation or plan is admissible. Execution then exposes the candidate to the registered environment, and the evaluator assigns an outcome from the resulting

behavior. These distinctions describe what must be identifiable in the record, without asserting that every implementation uses a separate software component for each stage.

Each transition must preserve the relation between the raw response and the object that was executed. A repaired action is a different action. Consequently, an unrecorded intervention between parsing and execution would make the outcome attributable to the combined intervention rather than to the model response alone. Diagnostic parsing may reveal where a response failed, but cannot retroactively turn an invalid scientific attempt into a valid one. A valid candidate can still produce a poor task result; an unavailable execution, in contrast, has produced no such measurement.

The attribution also depends on the contract actually supplied. Omission of a required field after a complete instruction and omission caused by an incomplete instruction are different explanations. Even in the former case, a model-level conclusion requires a functioning parser and validator. Raw output, the exact visible contract, the validator decision, and any resource interruption jointly determine which interpretation is supported.

C.3 Failure Taxonomy

Table 10 separates the location of a failure from its cause. A parser or schema rejection is an observed interface outcome; whether it reflects the generated response, a defective implementation, or a contract mismatch must be established from the corresponding evidence. Resource failure is similarly a description of an interrupted opportunity, not an automatic judgment about the model's reasoning.

Category	Attribution evidence	Scientific treatment
Parser failure	Raw response and parser behavior under the supplied contract	No task score from the rejected output; retain independent valid measurements
Schema failure	Required fields and types versus actual object; schema version	Invalid representation; preserve the actual attempt and rejection reason
Resource failure	Budget, truncation or termination evidence, and service or device record	Unavailable outcome; cause may be budget exhaustion or infrastructure
Semantic invalid plan	Complete public semantics and a supported validator rejection	No legal task execution; do not assign a non-detection score
No-change decision	Canonical legal candidate retaining the parent state	A valid decision that consumes its scheduled opportunity
Valid update	Legal candidate and recorded installation	Admissible successor state; utility remains to be evaluated

Table 10: Output and update statuses, the record used to assign each status, and whether the attempt can supply a task score or successor state.

A legal program that executes but detects no fault is a valid negative task outcome under the scoring rule. It belongs outside the invalid-plan category. Dependent SKIPs identify evaluations that were not attempted because a required predecessor was unavailable; counting them as repeated model failures would inflate both the apparent sample size and the estimated failure frequency. UNKNOWN preserves unresolved attribution rather than concealing it inside either the valid or invalid count.

D Experimental Setup and Reproducibility

D.1 Model Configuration

A model condition is identified by the requested model or checkpoint, the provider or local serving configuration, the version, and the identity evidence returned during the run. These fields have different evidential roles. A gateway response can establish the identifier returned by that gateway without independently authenticating an upstream checkpoint. Conversely, locally available weights do not identify the active serving configuration unless the loaded checkpoint and inference setup are bound to the run. API access and weight availability are therefore reported as separate properties.

The archive includes requested identifiers, gateway-returned identities, selected request-level records, and local model configuration hashes for the reported conditions. These records support

the named route or local checkpoint condition, but a gateway-returned identifier does not authenticate the provider's upstream weight version. For each included condition, the record distinguishes the requested identifier, returned identifier, version evidence, and scope of the identity guarantee; unspecified values remain unavailable rather than being completed from the model name.

D.2 Runtime Configuration

Reproduction requires the generation parameters as actually sent, including any temperature or sampling controls, output cap, reasoning budget, context limit, and stopping settings. An omitted parameter must be distinguished from an explicit value and from a provider default, whose behavior may depend on the serving version. When reasoning usage is not exposed, its absence does not imply that no reasoning computation occurred.

The update protocol prohibits repairing or resampling an invalid scientific candidate. This restriction does not, by itself, document how a serving client handles transport errors. Request identifiers and retry records must establish whether a failed dispatch was repeated, whether a response was generated more than once, and which response entered the lineage. Neither a generic client setting nor a claim of single dispatch can replace that record. The per-run request and response ledgers permit condition-specific inspection of transport and generation settings; this manuscript does not infer a universal policy across providers from any one run.

Execution budgets are recorded separately from generation budgets. The task's primitive allowance governs program execution, while an output cap constrains generated tokens; matching one does not match the other. Costs likewise distinguish task execution from source acquisition and additional future-update diagnostics. Available input, output, cache, and reasoning usage should be retained without substituting zero for unreported fields.

D.3 Hardware and Software Environment

The experimental environment is identified by the actual accelerator configuration, runtime and serving versions, benchmark version, and evaluator version. A source commit is useful only when it identifies the implementation used for the reported run; uncommitted changes require an accompanying snapshot or patch. GPU model and count, relevant precision settings, and the allocation on which the condition ran cannot be inferred from a planned deployment or a filename.

The archived run packages contain condition-specific model, decoder, and runtime records, including local configuration hashes. A single complete environment manifest covering every historical condition is not established by these selected records. The typesetting environment establishes how this manuscript was built, but provides no evidence about the machines that generated the research outcomes. Successful manuscript compilation does not resolve any missing experimental environment field.

D.4 Artifact Identity and Replay

A reproducible comparison binds protocol, dataset, evaluator, initial policies, and source schedule to immutable artifact identities. Hashes establish whether the same bytes were used; they do not establish that those bytes implement the intended scientific condition. Protocol review and execution validation remain necessary even when all hashes agree. In particular, a dataset hash without the partition and source bindings cannot establish which information was available to an update.

Replay should reconstruct the evaluated states from the retained candidates and installation decisions, associate each evaluation with its actual input state, and apply the specified evaluator and aggregation rule to the corresponding outcomes. This is distinct from making new model requests. Shared diagnostic results must retain their equivalence-map membership, so a single recorded execution does not become several independent observations on replay. Any claim of independent checking must identify the records covered and the calculations verified, including unresolved exclusions.

The separate research archive used for this analysis contains run reports, selected raw request/response records, scorebooks, and provenance ledgers; the manuscript source package itself is not the experimental record bundle. Protocol, dataset, and evaluator hashes cannot be filled with

hashes of the LaTeX files. The planned experimental release will bind each reported comparison to its protocol version, task and SOURCE manifests, initial policies, actual model requests and generation settings, validator and scorer versions, exact TES aggregation weights, response-provenance ledger, and replay command. A source-complete traced lineage will show both installations and the scored downstream TASK outputs. Restricted provider responses may require a permitted substitute, but their omission must be explicit. Until this package is available, an external reader cannot independently rerun the full benchmark. Credentials and identifying server details do not belong in the release.

E Metrics and Statistical Analysis

E.1 Metric Definitions and Comparison Units

The task score is defined on legal executions. For a faulty implementation, let k be the index of the first primitive at which the fault is detected under the fixed allowance $B = 1024$. The per-implementation value is

$$a = \begin{cases} (B - k + 1)/B, & \text{if a legal execution detects the fault,} \\ 0, & \text{if a legal execution does not detect the fault.} \end{cases} \tag{5}$$

The score rewards earlier detection within the same execution allowance. It does not reward an invalid plan that stops early. CORRECT instances supply a separate false-positive measurement and must not be interpreted as missing fault detections.

TES is 100 times the archived scorebook's hierarchical mean of these values. Target, mechanism, and lineage weights cannot be inferred from implementation counts, and their exact values are not enumerated in this manuscript source package. The reported readouts therefore depend on the archived scorebooks until the scorer and aggregation configuration are released. System Gain compares paired R and F task endpoints, while RIG compares common-recipient utilities U_3 and U_0. Both are in TES points; neither is a probability of recursive success.

The current protocol does not define a single improvement rate, transfer rate, persistence rate, or recursive success rate that replaces these measurements. Any such rate would require a frozen success predicate, sampling unit, eligibility rule, and denominator. A byte-change frequency, a handoff frequency, a task-detection frequency, and the fraction of lineages satisfying a persistence criterion are different quantities. Without those specifications, attaching one of the broader rate labels would introduce a new benchmark definition. Accordingly, no additional rate or threshold is manufactured for this appendix.

Local effects retain their same-source parent and child comparison. Method records separately report installed changes, exact successor consumption, qualified handoffs, and maximum consecutive depth within a lineage. Content review can describe those changes but cannot delete a legal event retrospectively. Counts of independent conditions or lineages remain distinct from counts of graphs, private implementations, sampled task artifacts, and logical evaluation nodes.

E.2 Denominators, Coverage, and Missing Outcomes

Coverage is specific to an estimand. A valid MAIN state can have unavailable G2 or G3 measurements, and an evaluable G3 proposal need not belong to a lineage with a qualified MAIN handoff. Partial coverage means that only a subset of the registered observations for a stated measurement is available. Not estimable means that the required comparison or aggregation cannot be formed from those records. A valid negative result requires an evaluable observation under the specified comparator; it is not inferred from missing output.

Planned denominators, actual attempts, and evaluable observations must therefore be retained separately. A SOURCE-dependent SKIP is not another model generation. A shared diagnostic is not another independent sample. An ineligible G4 lineage is excluded by a structural condition, whereas a missing eligible-window outcome limits evaluation within that condition. Reporting both categories is necessary to distinguish a population that did not exhibit the required handoffs from one for which subsequent utility could not be measured.

For a registered finite population of legal fault-test scores whose values are partly unavailable, missingness can be bounded without treating missing values as observed zeros. If the frozen hierarchy induces nonnegative implementation weights w_j summing to one, let $\mathcal{O}$ denote the evaluable subset. Because each legal fault-test value lies in $[0, 1]$, the full task score is bounded by

$$100 \sum_{j \in \mathcal{O}} w_j a_j \;\le\; \mathrm{TES}_{\mathrm{full}} \;\le\; 100 \left(\sum_{j \in \mathcal{O}} w_j a_j + \sum_{j \notin \mathcal{O}} w_j \right). \tag{6}$$

These are algebraic sensitivity bounds conditional on the original population, weights, and score range, not a replacement estimator or confidence interval. Structurally undefined outcomes cannot be inserted into this population without an explicit estimand definition. The archived scorebooks report these bounds where the original weights and slot statuses are available. Renormalizing the observed weights instead would describe the evaluable subset and could change the target quantity when difficult cases are disproportionately missing. The same distinction must be retained when interpreting paired contrasts.

E.3 Statistical Procedures and Uncertainty

A paired analysis preserves the registered relation between F and R, including their common initialization and condition. Repeated observations from related graphs, successive states, or shared diagnostics cannot be counted as independent replicates simply because they occupy separate table cells. A confidence interval requires a stated estimand and sampling model; it does not correct an unidentified model version, a mismatched control, or an unavailable scoring rule.

A bootstrap specification must identify the resampling unit, treatment of paired and clustered observations, number of resamples, interval construction, and handling of missing or reused responses. Those choices affect the interpretation of its uncertainty estimate. The independent same-method A2 analysis corrected an earlier pair-row resampling unit to the task-package unit; its resulting interval does not establish equivalence within five TES points. Cluster and leave-one-out analyses likewise require their actual units and outputs to be stated with each reported result. None of these procedures repairs response reuse or missing dependent TASK outcomes.

A small observed difference is not an equivalence result. Such a result would additionally require a prespecified equivalence margin and the corresponding completed analysis. When a diagnostic is shared because its bound inputs are identical, the resulting zero difference follows from reuse and supplies no independent estimate of sampling uncertainty. Missingness bounds describe a different source of uncertainty and are reported separately from intervals derived from a sampling procedure.

Resource accounting accompanies these measurements rather than modifying their success criteria. TASK-only execution, SOURCE-inclusive execution, and additional G3 diagnostics retain separate costs, alongside available input, output, cache, and reasoning-token usage. Unknown service charges or device usage remain unknown. Apparent savings due to invalid termination or short non-detecting programs do not establish a performance-efficiency gain.

F Control Results and Qualification Limits

F.1 Output-Cap Qualification Status

Archived summaries record the output limit sent in each condition. They do not supply a matched comparison across 1,024, 4,096, 8,192, and 16,384 generated-token caps, so this paper makes no claim that a particular cap was selected by a completed validity study or that changing it was purely administrative. A larger cap can change which substantive candidate is completed as well as whether its JSON parses. The separate allowance of 1,024 task primitives in Equation 5 is an execution budget, not an output-token cap. Results obtained under different caps remain distinct conditions.

F.2 Execution Risks and Archived Checks

An implementation audit asks whether the stated intervention was the one actually applied. Table 11 maps risks found in archived reports to the record needed to resolve each one; its third column is

an audit requirement, not a declaration that every check passed. A repair may validate one interface condition without retrospectively changing an earlier result.

Issue	Scientific risk	Required resolution evidence
Visible contract changes alongside decoder constraints	An apparent assistance effect conflates information access with generation constraints	Exact visible payloads and schema versions establishing the intended matched condition
False diagnostic rejection	Missing future-utility observations are attributed to the tested method	Request-level rejection review and a validated condition-specific decision rule
Required READ field absent	An invalid representation is scored as a legal non-detection or silently repaired	Raw response, complete contract, and typed and semantic validation kept separate
Legacy source aggregation assumption	An exposed subset changes the intended source distribution or denominator	Versioned source bindings and full-panel aggregation validation
Installation differs from successor input	Utility is credited to a procedure that was not used	Exact installed state and successor payload correspondence

Table 11: Interface risks in archived runs, the scientific error each could cause, and the record needed to resolve it.

State and data contracts must be checked together: a structurally legal state can still receive the wrong source record. Similarly, a functioning runner can execute a protocol-deviating comparison without raising an error. Information-isolation checks must therefore establish which fields reached the model, while semantic-invalid separation must preserve the difference between an inadmissible plan and an executed negative outcome. A successful parser test addresses neither question on its own.

F.3 Recorded Controls and Unestimated Sensitivities

No completed cumulative-readout ablation holds one lineage record fixed while selectively revealing endpoints, handoffs, common-recipient utility, and coverage. The available archived candidate-selection experiment also lacks enough independent validation groups to estimate whether G3 predicts later utility beyond prior TASK scores. These are unestimated comparisons, not negative findings about G3.

The complementary reference-method control asks whether the G3 diagnostic is sensitive to a useful difference in procedure. A 12-call given-method calibration recorded four of four OUTER method artifacts, four of four downstream task-policy installations, and three of four valid fault-detecting TASKs versus zero of four historical no-method TASKs. The controls were neither contemporaneous nor randomized. In a separate fixed-artifact check, 32 new TASK calls were scored on ten M1 worlds across two previously used development targets. REF_A exceeded BASE by $+17.0276$ M1 TES with native updates (two of four experience units positive) and $+38.0774$ with instructed application (four of four positive); 27 of 32 TASK outputs were strictly valid. The check reused historical update artifacts and development targets. Both studies establish local score sensitivity, not autonomous method discovery or a new independent G3 confirmation. Synthetic passes remain implementation checks only.

F.4 Five-Arm Fixed-Method Calibration

The subsequent normalized-method study generated 116 new requests on local Qwen3.8-27B: four SOURCE, 32 ENACTMENT, and 80 TASK calls. Its four SOURCE units supplied authorized experience, and the 12 target identities were new instances of the existing CHAIN_BRANCH_JOIN topology and M1/M2 fault families. No OUTER or autonomous method generation was run. The instructed recipient received a fixed reference method, a baseline method, an ablated method, or no update. REF_A and REF_B were independently generated applications of the same reference text. A frozen adapter removed only a complete outer Markdown fence before passing ENACTMENT content to the original strict update parser. Original strict acceptance and post-adapter acceptance are separate observations.

Arm	Native update	Adapted update	Valid TASK	M1 TES or bound
BASE	8/8	8/8	14/16	20.14
REF_A	0/8	8/8	14/16	[75.83, 82.08]
ABLATED	0/8	8/8	16/16	0.00
REF_B	0/8	8/8	16/16	86.01
INITIAL	–	–	14/16	[26.50, 32.75]

Table 12: Native and fence-normalized update acceptance, valid TASK counts, and M1 TES for five fixed-method arms. This guided calibration omits OUTER.

REF_B exceeded BASE by 65.8710 TES and ABLATED by 86.0126 on the complete fixed panel. All 16 REF_B TASKs showed the specified rollback-and-read operation pattern and detected M1, compared with nine patterned and eight detecting TASKs for BASE, and one patterned but no detecting TASK for ABLATED. REF_A has one unavailable target; its full-panel TES bound is shown in Table 12. The three preregistered composite tests did not all meet their completeness and multiplicity requirements. INITIAL omits the ENACTMENT call, so it is also not a matched-compute method arm. These results show functional separation under an instructed, normalized interface, while leaving full OUTER handoff, native strict acceptance, and independent autonomous G3 untested.

F.5 Real-OUTER Bridge and Task-Policy Clamp

Another local Qwen3.8-27B study generated 112 bridge requests across four independent M1 task packages, with two producer methods and two repeats nested within each package. The NATIVE branch carries OUTER's installed task and improvement policies into ENACTMENT. METHOD_CLAMP restores the common initial task policy after the same OUTER response but retains its installed improvement policy; ENACTMENT and both TASKs are then generated independently. TASK requests omit the structured-output parameter in this new evaluation condition, while the original strict receiver, scorer, and 1,024-operation task cap remain in force. The branch comparison therefore cannot be pooled with an earlier structured-output condition.

The package-level contrasts are shown in main-text Table 6.

For NATIVE, the package-bootstrap 95% interval is [66.32, 89.13] TES and the exact two-sided sign-flip value is $p = 0.125$. For METHOD_CLAMP, they are [0, 43.96] and $p = 0.5$. The four-package precision is insufficient to identify a stable pure method-content effect. In particular, the NATIVE contrast can carry OUTER's task-policy change, whereas the clamped contrast is positive in only two packages. In the preceding structured-output condition, 25/32 TASK responses repeated an invalid empty-transaction pattern; removing the TASK structured-output parameter changed the generation condition and made a usable bridge possible. No new natural recursive-success claim follows from this controlled bridge.

No numerical denominator, validity-filter, horizon, or aggregation sensitivity estimate is added here. The reported missingness bounds retain the planned denominator. Restricting an analysis to valid outputs would change its population; shortening the horizon could change eligibility but could not supply an unexecuted continuation. Any future alternative weighting would be secondary to the frozen scorebook aggregation. These distinctions limit the present claims and should not be read as completed robustness results.

G Trace Availability and Failure Attribution

G.1 What a Complete Trace Must Show

The materials available for this manuscript do not contain a source-complete successful natural lineage. We therefore do not combine favorable fragments from different conditions into a worked success case. The planned experimental release should include a complete attempted lineage, whether successful or not, linking its T0 policies and SOURCE input to each raw candidate, legality and installation decision, successor payload, TRANSFER result, and common-recipient continuation. A positive G3 or G4 claim would additionally need the initial-method comparator and the later

scored window. Until that trace is accessible, the numerical summaries do not provide an external end-to-end reconstruction of one lineage.

The exact before-and-after method texts matter as much as their hashes. A byte change can be whitespace, a near-paraphrase, task-specific facts, or a procedural revision. A content label requires those texts and a documented review procedure; it cannot change the recorded legal handoff event after the fact.

G.2 Observed Breaks in the Evidence Chain

Local change without changed-method inheritance. Sol has four positive matched fault-world G0 cells, but no changed-method handoff. Those cells share the run structure and are not four independent demonstrations of improvement. Its Recursive endpoint also falls below T0 despite beating Frozen. Local task variation, branch advantage, and inherited method benefit therefore give different readings of the same run.

Handoff without established later benefit. Scout records seven method edits and four qualified handoffs. Its measured method contrast is zero on 330 of 548 planned paired diagnostic cells, while the others do not supply a complete paired readout. One Scout lineage meets the two-handoff G4 entry condition but lacks evaluable later utility. These observations establish some realized use, not a general zero method effect or negative persistence result.

Numerical utility without independent provenance. Terra's historical $+1.619$ TES method contrast cannot serve as an independent replication after the response-ID audit. Two later qualification requests returned the same native response object and ID, and the strict original-condition repeat did not proceed to new scientific requests. The reported number remains a historical readout rather than a verified natural improver gain.

Interface failure before method evaluation. In the early P1A comparison, all 576 reference-arm OUTER outputs were rejected by the strict parser, leaving the initial method installed. A later structured-output condition installed a changed reference method in 144/144 OUTER chains and passed it to 136/144 successors. These runs demonstrate different delivery conditions; neither retroactively repairs the other, and the later utility comparison still lacks confirmatory qualification.

Historical Scout content notes mention trailing whitespace and near-paraphrase edits. Complete texts and request bindings are needed to verify those classes. We retain their legal handoff counts without relabeling them as substantive method innovations.

H Full Experimental Results

H.1 Models and Units

The archived evidence contains several run conditions. The early API continuation (Stage113), cross-model API matrix (Stage116), and remediated local panel (Stage124) are reported separately; a changed output contract or decoder does not inherit an earlier result. The full early API continuation and local panel each schedule 880 logical slots, including branch-equivalent shared uses, while the Sol and Astra conditions use smaller panels. Logical slots, physical model attempts, scored world cells, and independent upstream generations are different denominators. Main-text Table 4 gives endpoint task comparisons; Table 17 gives method transition and diagnostic status.

The earlier Stage113 trajectories for DeepSeek V4.1 Flash, Qwen3.8 Flash Next, and MiniMax M2.7 are withheld from the current comparison pending their replacement runs. A separate Stage116 DeepSeek V4.1 Flash run used a different panel and reused provider response IDs, so its zero task score and unchanged method are not pooled with the retained models. Stage116 DeepSeek V4 Flash stopped at a SOURCE length limit, Kimi K3 at smoke timeouts, and Kimi K3 Pro at the returned-identity gate. GLM 5.3 Flash reached API qualification only. Older DeepSeek V4 Pro and reference-method runs remain historical controls.

Panel	Model	T0	F1	F2	F3	R1	R2	R3
TES	gpt-5.6-sol	94.93	72.46	83.87	72.46	72.46	89.31	77.84
TES	gemini-3.8-flash	60.21	55.00	56.28	54.41	55.00	59.84	54.19
TES	gpt-5.6-luna	60.30	47.13	41.17	43.97	47.13	41.17	42.28
TES	gpt-5.6-terra	75.70	74.92	65.89	58.55	74.92	65.47	57.25
Cells / 548	Llama-4-Scout	502	422	400	422	422	422	438
Cells / 548	Qwen3.8-27B*	548	548	548	548	548	548	548
Cells / 548	Qwen3-8B	424	317	317	317	317	317	334
Cells / 548	Qwen3-1.7B	23	52	52	52	52	24	12
Cells / 548	Llama-3.1-8B	77	28	14	28	28	28	28

The upper and lower panels come from Stage113 DEV API and Stage124 local runs, respectively. TES is computed on each model's observed score cells; Luna's T3 coverage is 436/548. * Qwen3.8 used an older decoder flagged for descendant-slot impact review.

Table 13: T0–T3 task TES for the listed API models and scored-cell coverage for the listed local models, by Frozen and Recursive lineage.

H.2 Intermediate T0 through T3 States

Table 13 gives intermediate Frozen and Recursive readouts for the models listed there. The Stage113 TES trajectories are descriptive within each condition. They are not necessarily monotone: for example, Sol's T2 score rises relative to T1 and then falls by T3. For the Stage124 local models, the table reports scored-cell coverage at each state rather than repeating seven observed zero scores without their denominators. A state with incomplete coverage cannot supply a full-panel point score by carrying forward or replacing missing cells. Historical method-channel controls use different panels and execution conditions; Table 14 keeps their units and interpretation separate from the trajectories above.

Study	Observed result	Supported interpretation
Known-method channel	Given reference method: 4/4 created and accepted; 3/4 downstream TASK effects.	Local functional sensitivity; historical controls and guided method.
Unseeded discovery	Separate 136-call smoke: zero methods actually enacted downstream.	No verified autonomous method uptake in this smoke.
Real method handoff	136 new attempts; reference method content changed in 6/8 blocks; six calls missing.	Handoff and rollback primary effects not established.
Complete reference chain	66/72 calls; I+ versus I0 effect unavailable; fixed-panel bound [0.24, 43.99].	Missing dependent TASKs prevent a complete estimate.
Independent same-method repeat	126/128 calls; two independently generated streams.	Equivalence within 5 TES points not established.
Principle-content intervention	82/84 calls; principle-only minus neither +17.31 TES, cluster interval [-5.18, 44.82].	Pure principle attribution not established.
Natural-method replication	Terra fixed method: 128/128 calls but 82 slots share provider IDs; from-scratch run stopped at 389/880.	Historical positive readout lacks independent replication.
Recipient by method	48/48 local calls; fixed-denominator interaction bound [-2.49, 10.01].	Recipient independence remains inconclusive.
Public rule baseline	Older fixed panel: fixed rule / finite search TES, M1 9.94/96.06, M2 90.25/80.36; both 0/8 correct-world false positives.	Historical fixed-panel reference; the new structural panel is in Table 21.
Independent future utility	132/324 calls; natural pool has two independent validation groups; controlled pool has one editorial source.	Incremental predictive value of G3 not estimable.

Table 14: Archived method-channel controls and natural-method follow-ups, with completed calls or blocks and the inference supported by each record.

H.3 Fault Detection, False Positives, and Operation Use

The TES trajectories above do not show whether a task plan actually finds a fault, falsely reports one on a CORRECT world, or uses fewer operations because it stops early. Table 15 gives the scorer's fault-detection rates together with each state's scored-cell coverage and correct-world false-positive count. Table 16 reports task-only and SOURCE-inclusive primitive use under their distinct accounting scopes. Detection and cost use evaluable outputs only; a missing plan is not a zero-detection execution, and sparse correct-world denominators cannot establish a distribution-level safety rate. The API and local panels remain separate conditions.

	Fault detection % (scored cells)			Correct-world false positives		
Model	**T0**	**F3**	**R3**	**T0**	**F3**	**R3**
Sol	100.0 (108)	75.0 (108)	81.3 (108)	0/8	0/8	0/8
Astra	100.0 (108)	100.0 (108)	100.0 (108)	0/8	0/8	0/8
Gemini Flash	62.3 (548)	56.3 (548)	56.1 (548)	0/32	0/32	0/32
Luna	62.8 (548)	45.3 (436)	43.9 (436)	0/32	0/26	0/26
Terra	78.9 (548)	60.2 (548)	59.1 (548)	0/32	0/32	0/32
Llama-4-Scout	0.0 (502)	0.0 (422)	0.0 (438)	0/30	0/25	0/26
Qwen3.8-27B*	0.0 (548)	0.0 (548)	0.0 (548)	0/32	0/32	0/32
Qwen3-8B	0.0 (424)	0.0 (317)	0.0 (334)	0/24	0/17	0/18
Qwen3-1.7B	0.0 (23)	0.0 (52)	0.0 (12)	0/1	0/3	0/1
Llama-3.1-8B	0.0 (77)	0.0 (28)	0.0 (28)	0/5	0/2	0/2

* Qwen3.8-27B used an older decoder flagged for descendant-slot impact review.

Table 15: Fault detection among scored faulty cells and false alarms on correct implementations at T0, F3, and R3 for each model.

	Direct task primitives			SOURCE-inclusive primitives	
Model	**T0**	**F3**	**R3**	**F3**	**R3**
Sol	52.91	36.04	41.88	143.29	149.13
Astra	67.25	98.26	88.05	247.89	237.67
Gemini Flash	27.74	27.53	26.69	51.91	51.06
Luna	35.82	24.91	28.25	68.18	71.52
Terra	42.76	29.27	30.31	79.34	80.38
Llama-4-Scout	1.00	1.00	1.00	2.50	2.50
Qwen3.8-27B*	3.00	3.00	3.00	7.50	7.50
Qwen3-8B	12.81	13.45	13.10	35.35	35.00
Qwen3-1.7B	22.00	10.25	9.00	25.25	24.00
Llama-3.1-8B	11.00	10.00	10.00	25.00	25.00

* Qwen3.8-27B used an older decoder flagged for descendant-slot impact review.

Table 16: Mean primitive operations for initial and terminal task plans by model, counted directly for TASK and again with SOURCE operations included.

H.4 G0, G1, and Diagnostic Eligibility

The G0 sign counts in Table 17 were reconstructed from matched before/after rows on fault worlds in each run's parent-child ledger. They are descriptive world-level comparisons with repeated branches, lineages, and mechanisms. They neither substitute for the benchmark's hierarchical G0 aggregate nor supply independent trial counts. G1 content changes, exact successor consumption, and qualified handoffs are tracked separately from protocol-valid update messages. A change at the final scheduled opportunity is not presumed ineffective merely because no later successor was scheduled.

G3 diagnostic score cells retain planned and observed denominators and the recorded origin of each response. A zero from identical-method sharing is no changed-method contrast. Luna's incomplete diagnostic yields a full-denominator RIG identification bound that crosses zero; Terra's numerical positive readout is not counted as independent replication after the response-provenance audit. G4 structural eligibility requires sequential qualified handoffs, while a later scored window is a separate requirement. The remediated local Scout run has one structurally eligible lineage and no evaluable natural G4 utility result.

Model	G0 paired cells $+/-/=$	G1 method edit/handoff	G3 paired cells	G3 inference status	G4 eligible/evaluable
gpt-5.6-sol	4/7/5	0/0	108/108	no method contrast	0/0
gpt-6-astra	8/0/7	0/0	100/108	no method contrast	0/0
gemini-3.8-flash	1/4/59	0/0	497/548	no method contrast	0/0
gpt-5.6-luna	25/20/3	9/8	401/548	RIG -0.247; bound crosses zero	0/0
gpt-5.6-terra	17/15/32	4/3	548/548	historical $+1.619$; provenance unverified	0/0
Llama-4-Scout	0/0/52	7/4	330/548	observed RIG 0; incomplete	1/0
Qwen3.8-27B*	0/0/64	0/0	548/548	no method contrast	0/0
Qwen3-8B	0/0/40	0/0	317/548	no method contrast	0/0
Qwen3-1.7B	0/0/0	3/1	0/548	not estimable	0/0
Llama-3.1-8B	0/0/14	0/0	0/548	not estimable	0/0

Rows above and below the rule are API (Stage113 DEV, except Astra at Stage116) and local (Stage124) runs. G0 world comparisons are linked diagnostics, not independent trials. G1 records method-byte changes and qualified handoffs. Scout reaches handoff without observed G3 gain; Qwen3-1.7B has one handoff but no matched G3 scores. * The Qwen3.8 decoder qualification in Table 4 applies here.

Table 17: G0 cell changes, G1 method edits and handoffs, G3 paired coverage and inference status, and G4 eligible versus evaluable lineages for each model.

H.5 Method-Snapshot Diagnostic Trajectories

Table 18 gives available common-recipient TES readings across P_0 through P_3 and the paired cells used for the diagnostic contrast. The archives call these method snapshots I0 through I3; the table follows Figure 1's notation. Snapshot TES and RIG answer different arithmetic questions when coverage differs, while an unchanged method supplies no intervention even if all four scores can be read. The table therefore retains missing initial-method values and incomplete pairs rather than filling them with zero.

Model	P_0	P_1	P_2	P_3	Paired	RIG	Method evidence
Sol	94.05	94.05	94.05	94.05	108/108	—	unchanged
Astra	96.81	96.81	96.81	96.81	100/108	—	unchanged
Gemini Flash	59.56	59.56	59.56	59.56	497/548	—	unchanged
Luna	61.43	61.65	61.18	61.18	401/548	-0.247	changed; partial
Terra	73.37	73.17	74.98	74.98	548/548	+1.619	changed; reused IDs
Llama-4-Scout	0.00	0.00	0.00	0.00	330/548	0.000	changed; partial
Qwen3.8-27B*	0.00	0.00	0.00	0.00	548/548	—	unchanged
Qwen3-8B	0.00	0.00	0.00	0.00	317/548	—	unchanged
Qwen3-1.7B	—	0.00	0.00	0.00	0/548	—	no pair
Llama-3.1-8B	—	—	—	—	0/548	—	unavailable

* Qwen3.8-27B used an older decoder flagged for descendant-slot impact review.

Table 18: Common-recipient TES for method snapshots P_0 through P_3, with paired coverage, final versus initial RIG, and the status of each method contrast.

H.6 What the G4 Judge Fixture Distinguishes

Table 19 records three numerical paths in the archived, human-authored judge fixture. Its original positive-pattern label checks later scores against the initial utility U_0 under a separate reliability condition. It does not test whether the first gain was retained or whether every adjacent step improved:

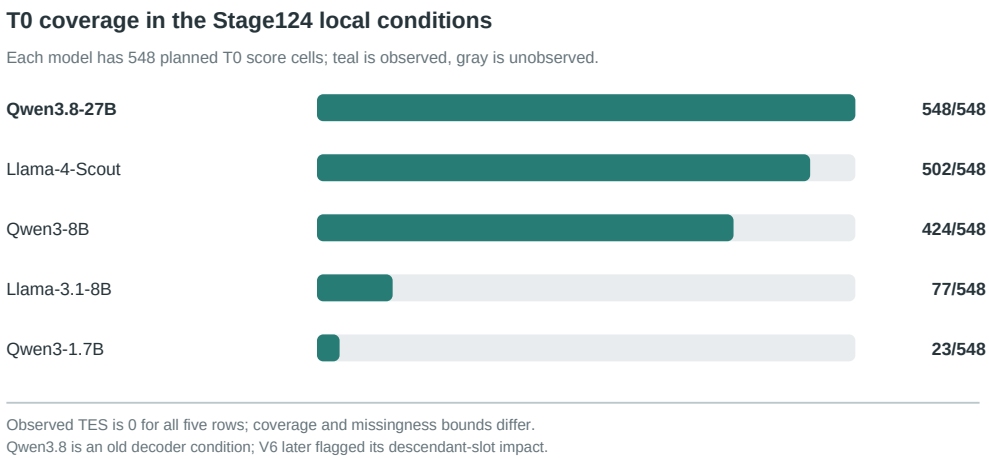

Figure 2: Observed T0 scoring coverage out of 548 planned cells for each local model. Teal marks scored cells; gray marks unavailable cells.

the label stays positive for a path that declines twice while remaining above U_0. We report those two stronger patterns separately in the main text. The fixture does not contain a model-generated lineage or substitute for an evaluable natural G4 continuation; a reproducible G4 claim also needs the exact reliability rule and its input records.

Constructed TES path	Later minus U_0	Adjacent gains	First gain retained	Original label
10, 20, 20, 20	10, 10, 10	10, 0, 0	yes	positive pattern
10, 20, 30, 40	10, 20, 30	10, 10, 10	yes	positive pattern
10, 30, 20, 15	20, 10, 5	20, -10, -5	no	positive pattern

Table 19: Three constructed G4 utility paths that share the original positive-pattern label but differ in adjacent gains and retention of the first gain.

H.7 Execution Coverage and Missingness

Main-text Table 9 partitions each 880-slot graph into scored, valid, invalid, skipped, shared, and unknown states. These statuses explain why two observed zero scores can carry different information. A semantic invalid output is not an executed TASK zero; a dependent skip is not another model call; and a shared node is not an independent replication. The model-specific scorebooks additionally retain failure reasons, token and elapsed-time usage, and the missingness bounds for each task state.

Table 4 reports observed endpoints and scored-cell coverage; the model-specific scorebooks retain full-panel identification bounds for unavailable cells under the planned weights and score range. Those bounds are not confidence intervals. They do not repair non-independent generations or authorize a post hoc change of the estimand. The controlled chain results in Table 14 retain separate package, block, call, and validation-group units for the same reason.

I Controlled Runs, Format Validity, and Replay

The separate G3 closure retained 11,203 raw formal requests, comprising 10,349 local and 854 remote requests. Empty-directory extraction and independent offline scoring replay agreed on 6,539 task scores and 146,629 worlds. This validates stored calculations, not intervention delivery or generation independence. The archive SHA-256 is `58a2896078da742ff332e606f2d03a81a492f297cdaf802621dacdfdba63e670`. Remote TP4 blocks and local TP1 concurrency changes leave the original all-TP1 P1A and P1C confirmation unavailable.

I.1 Delivery Before Utility

P1A records 288 complete four-chain units, but every reference-arm OUTER response failed strict raw-JSON parsing and retained the initial method. The wire request named a decoder schema without sending a structured-output parameter. Table 20 retains acceptance counts. In P1C, text and format validity vary together; source-only text may also carry procedural information, preventing a pure principle interpretation.

Study	Arm and stage	Accepted	Planned	Consequence
P1A	reference OUTER	0	576	initial method stays installed
P1A	initial OUTER	218	576	acceptance varies within control
P1C	FULL ENACTMENT	0	348	full text is not installed
P1C	BROKEN ENACTMENT	30	348	format differs by arm
P1C	PRINCIPLE ONLY ENACTMENT	73	348	format differs by arm
P1C	SOURCE ONLY ENACTMENT	177	348	may carry procedural content
P1C	NEITHER ENACTMENT	207	348	comparator acceptance differs

Table 20: Strict JSON acceptance at OUTER and ENACTMENT in the early P1A and P1C controls, with planned denominators and consequences for method installation.

P1A's descriptive reference contrast is -2.297 TES. Its two independent same-method arms have 288 complete pairs each, with mixed-deployment means -1.289 and $+3.502$ TES. The separate 32-pair three-opportunity null is -3.047 TES, 95% CI $[-16.361, 10.266]$, and two-sided $p = 0.640$; the five-point equivalence margin remains unresolved. Each P1C principal contrast has 348 complete pairs: FULL minus BROKEN is -2.939 TES, and PRINCIPLE ONLY minus NEITHER is $+5.150$ TES. Original all-TP1 intervals and Holm decisions are unavailable. The 32-pair structural transfer contrast is $+6.235$ TES, 95% CI $[-8.257, 20.727]$, and remains descriptive.

I.2 Natural Production, Alternative Recipients, and Prediction

All 16 precommitted new local terminal lineages completed. Six of eight Recursive terminal methods differ from I0, and the complete set has seven terminal content groups including I0. Fixed-method retests assess stability of a selected method rather than the frequency with which a new method is discovered. Four complete recipient-by-method cases yield descriptive interactions. An unchanged-method case gives an apparent $+33.528$ TES interaction, demonstrating that a nonzero case contrast can arise without method-content intervention. None of these local results reproduces the original Terra condition: two Terra qualification DEV requests returned an identical native response object and ID, and no new scientific request was made.

The public fixed rule and finite search were evaluated on eight newly frozen structural targets with the same task budget. Their respective M1/M2 TES pairs are $9.941/83.542$ and $96.064/74.445$. Each had zero CORRECT-world false positives on those eight targets. This is a fixed-panel observation, neither a distribution-level safety result nor an upper bound supplied by finite search.

One X selection campaign had seven candidates with complete measurements. Task-only, task plus G3, and five-task rules all chose `U3.P0.RECURSIVE`; 64 Y chains for that method were reused, giving zero TES differences by identical selection. Cross-source ranking value remains unestimable because source-group and within-group content were insufficient. The older D condition is also unestimable, although ten of 27 methods have all five task measurements.

I.3 Corrections and Subsequent Qualification

The corrected historical A2 I0 package-bootstrap interval is $[-18.594, 16.334]$ TES. For the older four-package A3 exact two-sided test, the smallest attainable p is 0.125; that condition cannot be retrospectively described as a .05 confirmation. The first blind content-agent records were retained, but a final independent two-agent review was not completed and is not described as human review.

Question	Observed units	Result and limit
Recipient and method	4 four-cell cases	An unchanged-method case has a +33.528 TES interaction. This cannot be a method-content effect.
New local methods	16 lineages	Six of eight Recursive terminal methods changed. The run does not replicate Terra.
Simple baselines	8 targets	Fixed rule M1/M2: 9.941/83.542 TES; finite search: 96.064/74.445. Both made 0/8 false positives on correct worlds.
Later utility	7 candidates; 64 later chains	Three frozen rules selected the same method, so their later TES difference is zero by identical selection. Cross-source value remains unknown.

Table 21: Recipient and method cases, new local methods, simple baselines, and later candidate selection, with the observed unit and limitation for each check.

An author-supplied format-control DEV status reports installation and consumption in 16/16 chains, but only 14/16 PRINCIPLE ONLY chains with two strictly parseable TASK responses. The latter misses its 15/16 gate after some outputs reached a 16,384-token cap. No preregistered confirmatory request followed that failed gate. The subsequent user-directed allocation is reported separately below.

I.4 Later Format-Controlled Handoff Across Two Deployments

After the failed DEV qualification, a later allocation assigned 176 complete blocks of the same four G3 studies to a 40GB TP4 service. Once that machine finished 112 blocks, the remaining 64 whole blocks moved to an 80GB TP1 receiver. This is an exploratory mixed-deployment condition, not the originally planned all-TP1 confirmation. Table 22 keeps the deployments separate. The archive contains 2,563 complete raw responses across 2,608 physical claims, with 45 transport-unknown claims and 24 dependent TASK slots never sent. A cross-machine audit found no overlap among the observed claimed slots, returned response IDs, or response objects; it cannot authenticate upstream generation. On the receiver, a separate offline program recalculated 512 TASK scores and 11,523 world results without disagreement. These checks support integrity of observed records, not completeness of every assigned chain.

Deployment	Blocks	Planned slots	Claimed	Raw	Unresolved / unsent
Original 40GB TP4	112/112	1,657	1,657	1,648	9 / 0
Receiver 80GB TP1	64/64	975	951	915	36 / 24
Combined allocation	176/176	2,632	2,608	2,563	45 / 24

Table 22: Planned, claimed, completed raw, unresolved, and unsent request slots in 176 G3 blocks split between the original 40GB TP4 and receiver 80GB TP1 deployments.

The structured-output run changes the delivery finding. All 144 P1A reference OUTER proposals installed a method different from their parent I0, and 136 of those methods were consumed by a later ENACTMENT; the remaining eight ENACTMENT requests timed out. All 440 P1C fixed-text ENACTMENT updates were accepted, although later TASK outcomes were sometimes absent. This demonstrates functional method delivery under the tested interface and removes the earlier all-fenced-response failure as an explanation of every reference-method result. It does not repair the failed pre-run DEV gates or the change in deployment between machines.

Table 23 reports each frozen contrast without imputing missing outcomes. The P1A reference-minus-initial comparison has 60 complete four-chain blocks among 72 assigned blocks; its all-block TES identification bound is $[10.142, 19.864]$. Both independent same-method contrasts and the P1B three-opportunity null are retained, since a nonzero difference can arise without a method-content intervention. The 40GB and 80GB complete-pair means also differ materially. For P1C, the FULL-minus-BROKEN bound crosses zero, while the PRINCIPLE ONLY-minus-NEITHER

bound is positive on this fixed panel but cannot identify a pure principle effect. No confidence interval, significance decision, or claim of independent Terra replication follows from this user-directed mixed-deployment run.

TES contrast	Scored blocks 40GB / 80GB	Observed mean 40GB / 80GB	All-block bound
P1A reference − initial	38/40; 22/32	+10.824; +26.220	[10.142, 19.864]
P1A initial independent null	38/40; 29/32	+0.452; −5.974	[−5.641, 1.303]
P1A reference independent null	39/40; 25/32	−4.829; +9.965	[−4.522, 7.978]
P1B three-step branch null	7/7; 1/1	+18.172; +9.912	[17.140, 17.140]
P1C full − broken principle	59/59; 23/29	+3.208; −4.554	[−3.426, 4.529]
P1C principle only − neither	53/59; 25/29	+13.125; +2.102	[2.854, 14.218]
P1C full − principle only	56/59; 23/29	−2.213; −6.415	[−8.597, 1.630]
P1C source only − neither	56/59; 27/29	+8.935; +0.241	[1.928, 7.609]
P1M structural reference − initial	6/6; 1/2	+35.374; −28.057	[23.023, 35.523]

Table 23: P1A, P1B, P1C, and P1M TES contrasts split by 40GB TP4 and 80GB TP1 deployment. All-block bounds include missing assigned outcomes and are not confidence intervals.

The combined OWNER archive has SHA-256 `668b7d02441e3df28ebcb374384d2be241f08bec407ec3181bf35a7951f1df90`. Its three nested delivery files passed their recorded SHA-256 checks. These hashes identify files but do not make them available to readers or establish the scientific validity of their interventions. The allocation is distinct from the earlier G3 closure, the R2B all-80GB null condition, and any proposed rerun of closed API models.

I.5 Additional Manually Checked Contrasts

The later point estimates summarized in main-text Table 8 are listed in full in Table 24. The author reports having checked these values, but their raw responses, scoring files, and intervention records are absent from the replayable package. They remain exploratory point readings, are not pooled with archived controls, and do not support independent reconstruction or interval estimates in this manuscript.

Comparison	Units	ΔTES
Reference vs. initial on new topology and fault family	40 blocks	+4.8
Fixed replay, terminal vs. initial	32 lineages	−1.8
Feedback and selection, terminal vs. initial	32 lineages	+2.5
Feedback and selection vs. fixed replay	32 pairs	+4.3
Full method vs. initial	40 blocks	+4.8
Principle only vs. initial	40 blocks	+4.1
Source facts only vs. initial	40 blocks	+1.2
Broken principle vs. initial	40 blocks	−0.4
Full method vs. source facts only	40 blocks	+3.6
Task plus G3 vs. task only on independent Y	24 pools	+3.1

Table 24: Additional target, feedback, method-text, and selection point contrasts from the later batch. The current archive has no raw records or confidence intervals for these values.