

Does Failure Feedback Direct Agent Self-Modification?

Rong He

Abstract

Self-improving agents use failure feedback to revise their code or instructions, but gains after selection do not establish target-specific repair or generalization. We measure directedness by comparing repair of a randomly selected target failure with repair of the parent’s other failures. Across 145 eligible modifications in public Darwin Gödel Machine logs, we find no detectable target-specific advantage. Controlled instruction-revision experiments with small open models show that relevant feedback can improve target repair. In a second study with a pre-specified analysis plan, target feedback raised repair by 20 percentage points for Qwen3-4B relative to matched same-category feedback; the 7-point increase for Phi-4-mini was not significant. Target repair did not reliably imply held-out improvement. Held-out gains were concentrated in settings with identifiable, repairable failure mechanisms, including planted errors and natural tool-interface failures. Benefits of edit localization and regression screening varied across settings, while eight-step revision loops and GEPA ended at similar mean held-out accuracy. Evaluations should report target repair, transfer to related failures, and held-out net change separately.

1 Introduction

In agent-improvement systems, an editor revises an agent after observing one of its failures, and an evaluator decides whether the revision is retained. The editor may be the agent itself or a separate agent. The Darwin Gödel Machine (DGM) [1] has a reasoning model read the log of a failed task and propose a change, which a coding agent then implements in agent code; the Huxley–Gödel Machine [2] and the Self-Improving Coding Agent [3] follow the same pattern, and reflective prompt optimizers [4–6] apply it to instructions. Run over many iterations with an archive, such a system is a population of agents in which some agents revise others and an evaluator selects among the results.

A higher score among retained revisions establishes an outcome of the whole process, but it does not identify how the observed failure influenced the edit. The same trajectory is compatible with revisions that target the failure they were shown, with revisions that are useful regardless of the failure, with selection retaining variants that happen to work, and with an editor that produces working variants more often. Melcer et al. [15] found that prompt optimization works about as well with deliberately wrong feedback, which already questions the first explanation for prompts.

We separate three questions about a single revision. Does it repair the failure that prompted it more often than the parent’s other failures (*directedness*)? Does the benefit extend to other failures in the same category (*category-level spread*)? And does it raise the score on a fixed evaluation set once failures repaired are weighed against correct answers broken (*net score change*)? Only the last is visible to selection, and only after selection. A further question is what makes directed repair raise accuracy on questions that no editor or selection step has seen.

The first question can be answered without a model of task difficulty or evaluation noise when the failure shown to the editor is drawn at random from the parent’s failures. Under the null of no directedness, the failure shown and the parent’s other failures are then exchangeable; the idea follows the fluctuation test of Luria and Delbrück [8], which asked whether mutations respond to the condition that selects them. DGM draws the failure this way, and its public logs record each agent’s parent, target, proposal and per-task outcomes. We analyze these logs and complement them with two controlled studies in which small open models revise their own instructions or the helper code of a tool-using agent, so that the evidence shown to the editor and the location of erroneous rules are under our control. The second study follows a pre-specified analysis plan and scores every revision on a held-out test split.

This paper makes four contributions.

1. A per-modification randomized measure of directedness, defined together with category-level spread and net score change, and the conditions under which it gives a valid test in adaptive populations (Section 2, Appendix A).
2. An analysis of the six public DGM runs: no detectable target-specific advantage at the level of the task, the programming language or the repository; small mean edit-level score changes; and a higher viability of revisions when the agent edits itself than when a fixed initial agent edits its descendants (Section 3).
3. A controlled experiment (study 1) in which directed repair lowers the score of clean instructions, mostly through revisions that rewrite the line controlling response length and format, and raises it when planted rules give many failures one editable cause (Section 4).
4. A second study (study 2) with a pre-specified analysis plan, a held-out test split, two model families and a tool-using agent. Holding the target fixed and varying the feedback, it finds directed repair in one model; edit scope, localization and regression screening have setting-dependent effects; and eight-step loops and GEPA end at similar mean held-out accuracy. An explanatory model relates directed repair, shared causes and net gain (Sections 5 and 6).

2 Problem Formulation and Measurement

2.1 Agents, Editors and the Revision Process

A parent agent is evaluated on a set of tasks and fails a subset $\mathcal{F}$ of them. An editor receives evidence about one failure $e \in \mathcal{F}$ and produces a child agent, which is evaluated in turn; an evaluator compares scores and decides whether the child joins the population from which later parents are drawn (Figure 1). In DGM the agents are coding agents, the editor is an agent that implements a proposal written by a separate reasoning model, and the evaluator is a benchmark. In our controlled studies the agent is a language model following an editable instruction text, its *playbook*, or a code-writing agent with an editable helper library, and the editor is the same model asked to revise them. We call the editor’s written account of a failure its *diagnosis*, and a change of the child’s outcome on a task from failure to success a *repair*. A diagnosis can name a cause without the revision repairing anything.

2.2 Directedness, Spread and Net Score Change

Let $Y(t) \in \{0, 1\}$ be the child’s outcome on task t , $\bar{Y}(S)$ its mean over a set of tasks S , $\mathcal{F}_e$ the parent’s failures in the category of e , and N the number of tasks in the evaluation set. For one

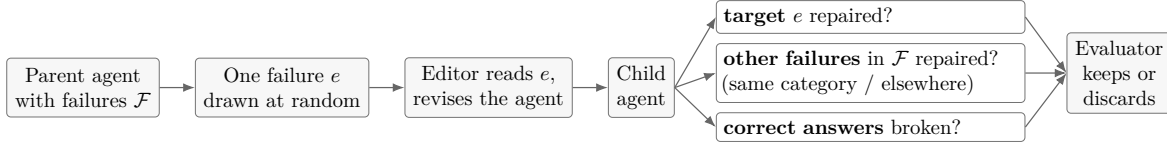


Figure 1. One self-modification and the quantities measured on it. A failure e of the parent is drawn at random and shown to the editor; the child is evaluated on e , on the parent’s other failures and on the parent’s correct answers. Directedness compares the target with the other failures of the same child; category-level spread compares other failures in the target’s category with failures elsewhere; net score change is failures repaired minus correct answers broken. The evaluator sees only the score.

Table 1. The three quantities measured on each revision.

Quantity	Measures	Does not establish
Directedness	repair of the failure shown, relative to the parent’s other failures	that the editor identified the cause of the failure
Category-level spread	repair of other failures in the category of the failure shown, relative to failures elsewhere	generalization across tasks or distributions
Net score change	failures repaired minus correct answers broken, on a fixed evaluation set	the long-run gain of a population after selection

revision prompted by failure e ,

$$d = Y(e) - \bar{Y}(\mathcal{F} \setminus \{e\}), \quad c = \bar{Y}(\mathcal{F}_c \setminus \{e\}) - \bar{Y}(\mathcal{F} \setminus \mathcal{F}_c), \quad \Delta = \frac{\sum_{t \in \mathcal{F}} Y(t) - \sum_{t \notin \mathcal{F}} \{1 - Y(t)\}}{N}. \quad (1)$$

Directedness is the mean of d over revisions, category-level spread the mean of c , and net score change the mean of Δ . Table 1 states what each measures and what it does not establish.

2.3 Randomized Target Selection and Inference

Index the modifications by m in the order they were made, ordering arbitrarily the modifications that run in parallel. Let $\mathcal{A}_{m-1}$ be everything recorded for modifications $1, \dots, m-1$, together with the parent of modification m and its evaluation, and let $\mathcal{P}_m$ be the *pool*, the parent’s failures from which the target is drawn. The pool depends on the history. Given the history, the target e_m is drawn uniformly from the pool with fresh randomness:

$$P(e_m = t \mid \mathcal{A}_{m-1}) = 1/|\mathcal{P}_m|, \quad t \in \mathcal{P}_m. \quad (2)$$

The child then works or not (V_m), is evaluated on a set of tasks $\mathcal{E}_m$, and has outcome $Y_m(t)$ on each evaluated task. A modification enters the analysis if its child works and its target was evaluated, $I_m = \mathbf{1}[V_m = 1, e_m \in \mathcal{E}_m]$, and its *analysis set* is $\mathcal{F}_m = \mathcal{P}_m \cap \mathcal{E}_m$. Write $y_m = Y_m(e_m)$ and

$$q_m = \frac{1}{|\mathcal{F}_m|} \sum_{t \in \mathcal{F}_m} Y_m(t). \quad (3)$$

The null hypothesis is that, given $\mathcal{A}_{m-1}$, the drawn failure affects neither whether the child works, nor the tasks on which it is evaluated, nor its outcomes: $(V_m, \mathcal{E}_m, Y_m)$ is conditionally independent of e_m . Under this null and Equation (2), the target is still uniform on the pool after

conditioning on $(V_m, \mathcal{E}_m, Y_m)$, and therefore uniform on $\mathcal{F}_m$ after also conditioning on $I_m = 1$. The probability that an included target is repaired is then q_m , and $Z_m = I_m(y_m - q_m)$ has conditional mean zero given everything that precedes it, however later parents, pools, viability and evaluation stages depend on earlier draws (Appendix A).

Primary test. Since $|Z_m| \leq 1$, for every $\lambda \in [-1, 1]$ the product $\prod_m (1 + \lambda Z_m)$ is a non-negative martingale with initial value one under the null. Averaging over a fixed grid Λ of 20 equally spaced values in $(0, 1]$ gives

$$E = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \prod_m (1 + \lambda Z_m), \quad (4)$$

an e-value for a directed effect. By Ville’s inequality [25], $P(E \geq 1/\alpha) \leq \alpha$ under the null for every α , at any sample size and at any stopping time [26, 27]; evidence at the 5% level requires $E \geq 20$. Under the stated null and the randomized design of Equation (2), this guarantee needs no asymptotic regularity conditions.

Auxiliary statistics. The martingale statistic $z = \sum_m Z_m / (\sum_m I_m q_m (1 - q_m))^{1/2}$ is approximately standard normal when the martingale central limit theorem applies [28], and we report it as an asymptotic approximation. When modifications do not influence one another, as in the controlled experiment, $T = \sum_m y_m$ is exactly Poisson–binomial with parameters (q_m) ; for the DGM logs this distribution is only a reference. We summarize the effect by the mean of d over the analysis sets, with intervals from resampling modifications or parents, and by the Mantel–Haenszel odds ratio with one stratum per modification.

2.4 What the Measurements Identify

Directedness is a contrast within one child, so it does not depend on how good the child is, but it has limits that matter for interpretation. A revision that repairs every failure equally, for example by removing a cause that all failures share, has no directedness and may still be effective (Section 6). A test that does not detect directedness therefore does not show that the feedback was useless, that the diagnosis was wrong, or that the edit had no effect. Conversely, directedness after the editor has seen the correct answer shows that the revision responded to the evidence, not that the agent gained a general reasoning ability. Spread is measured within the benchmark’s own categories, and net score change on a fixed evaluation set before selection.

These limits also apply to edits that are general by design. DGM’s diagnosis prompt asks for an improvement that is not specific to the issue, but it shows the failed issue, the agent’s patch and the hidden-test results, so the failure is the evidence on which a general edit is chosen. An edit that removes the cause of the observed failure is expected to repair that failure more often than a randomly chosen other failure, unless all failures share the cause. The test asks whether the evidence shaped the edit in a way that matters for that evidence; whether the edit was useful is the question of net score change.

3 Evidence from the Darwin Gödel Machine

3.1 Logs, Evaluation Stages and Inclusion

DGM keeps an archive of coding agents. In each iteration it samples a parent with probability proportional to $\sigma(10(s - 0.5))/(1 + c)$, where s is the parent’s score and c its number of children, and chooses an *entry*. On SWE-bench, with probability 0.25 the entry is the goal “solve empty patches” if at least 10% of the parent’s evaluated tasks ended with an empty patch; otherwise, with probability 0.25, “solve stochasticity”; otherwise, with probability 0.25, “solve context length” if the parent hit the context limit on a failed task; otherwise a task drawn uniformly from the parent’s unresolved tasks. On Polyglot the entry is always a task, drawn uniformly from the parent’s unresolved and empty-patch tasks. For a task entry, a reasoning model (o1) reads the parent’s log on that task, including the hidden-test results, and writes an improvement proposal that is instructed not to be specific to the issue; a coding agent implements the proposal in agent code. In the main run the editing agent is the parent itself; in the *fixed-editor* ablation (“w/o self-improve” in [1]) the initial agent edits every descendant. A child is evaluated on 10 tasks, then on 60 (SWE-bench) or 50 more (Polyglot) if it clears a threshold, and the strongest agents on 200 SWE-bench tasks [1].

The released logs cover six runs: DGM, the fixed editor, and DGM that always edits the latest working agent (the *no-archive* run, “w/o open-ended exploration” in [1]), each on SWE-bench Verified and Polyglot. The Polyglot no-archive release is partial (20 attempts). For every agent we recover its parent, entry, proposal text and per-task outcome; for Polyglot children we read the per-task prediction files, which also cover the second evaluation stage. A modification enters the directedness analysis if its entry is a task, its child works and the child was evaluated on the target; incomplete Polyglot evaluations are dropped (Appendix B counts them as failures instead).

3.2 Task-Level and Group-Level Directedness

Of 145 eligible modifications, 12 repaired their target and 16.4 were expected under the null (Figure 2a; per-run counts in Table 7, Appendix B). The target’s repair rate was 8.3%, against 11.6% for the parent’s other failures (difference -3.3 percentage points, 95% CI -8.2 to $+2.0$), and the pooled odds ratio was 0.69 (0.31–1.18). The e-value for a directed effect is 0.15, far below the value of 20 required for evidence at the 5% level, and the martingale statistic is $z = -1.23$ (asymptotic one-sided $p = 0.89$). No run showed a significant directed effect, and five of six point the other way.

Modifications of the same parent share history, so we also resampled parents rather than modifications: with 101 parents the interval for the difference is -8.2 to $+1.8$ points. Leaving out any one run, the difference stays between -1.5 and -4.2 points. The test has power against effects of practical size: adding a directed boost δ to each q_m , the reference test rejects at the 5% level with probability 0.72 for $\delta = 6$ points, 0.89 for $\delta = 8$ and 0.96 for $\delta = 10$, which roughly doubles the repair rate. Further checks in Appendix B hold the task fixed, count incomplete evaluations as failures, and condition on how many failures a child repaired.

The proposals are consistent with this. Early in the SWE-bench DGM run, three different failures, in two repositories and diagnosed for two different parents, produced the same proposal: let the editor change a range of lines instead of rewriting whole files. Comparing embeddings, a proposal is more similar to the issue text of its own target than to that of a randomly selected other failure of the parent with probability 0.58 in the DGM run, 0.52 with the fixed editor and

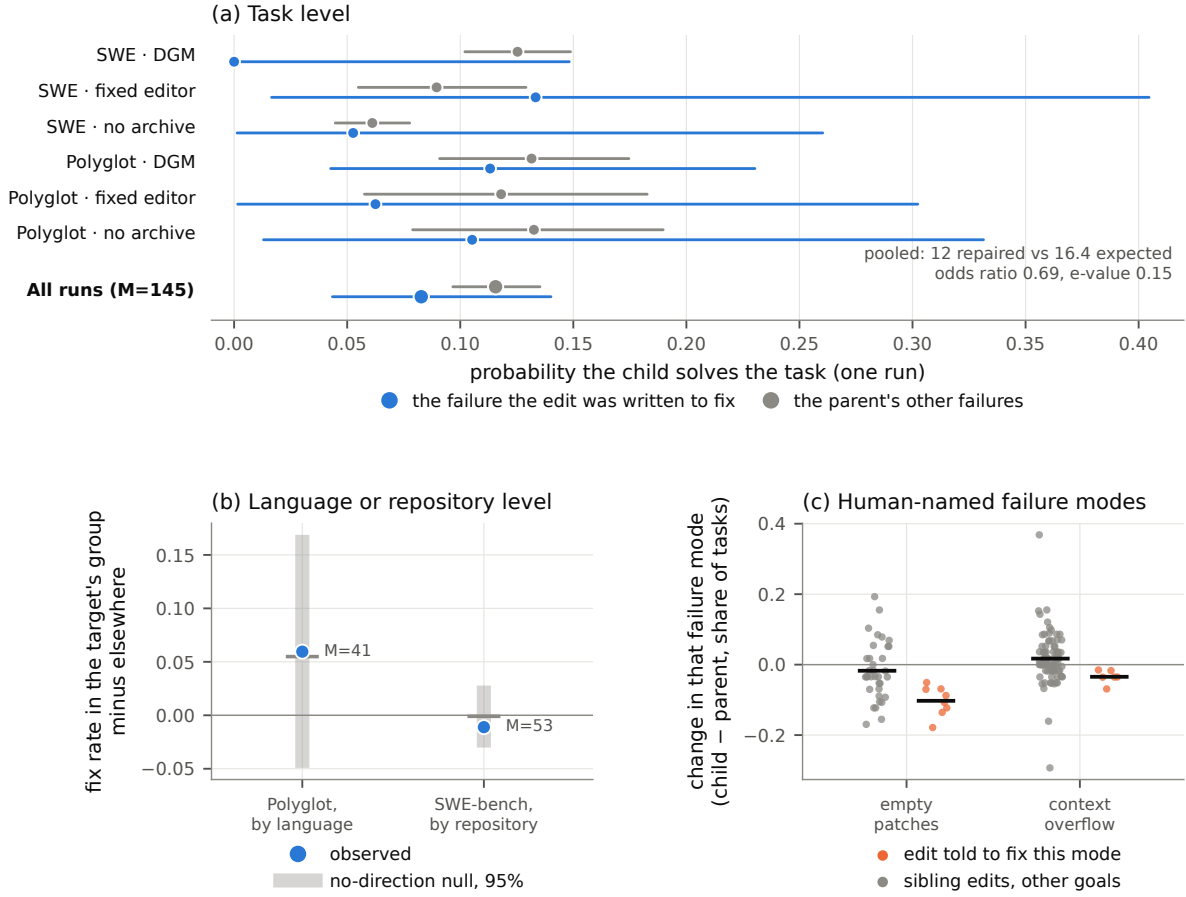


Figure 2. Directedness of DGM self-modifications. (a) Repair rate of the target (blue; Clopper–Pearson binomial 95% interval, a reference that does not account for adaptive sampling) and mean repair rate of the parent’s other failures (gray; bootstrap over modifications), per run and pooled; 145 modifications. (b) Repair rate of the parent’s other failures in the target’s language (Polyglot) or repository (SWE-bench) minus the rate elsewhere (blue), against its randomization null (gray band, 95%). (c) Change in a human-named failure mode for working children evaluated on at least 50 tasks, by assigned goal (orange) or other entries (gray); bars are means. Panel (c) conditions on survival, which the goal can affect (Section 3.3).

0.59 without the archive, and the coding agent that implements the proposal never sees the failure.

Build tools and test runners differ across languages, so a diagnosis could be directed at the target’s language without being directed at the task. Comparing the repair rate of the parent’s other failures in the target’s group with the rate outside it, against the randomization null obtained by redrawing the target within its pool, the excess on Polyglot, grouping by language, is +6.0 points against +5.4 expected ($M = 41$, $p = 0.45$); on SWE-bench, grouping by repository, it is −1.1 against −0.1 ($M = 53$, $p = 0.75$; Figure 2b). Neither shows a detectable group-level effect.

Table 2. Human-named failure modes in the SWE-bench runs. Working children: children that worked and were evaluated on at least 50 tasks; counts are children that reduced the parent’s rate of the mode, with the goal against other entries (Fisher p ; descriptive, conditions on survival). All attempts: attempts that produced a working child that reduced the mode, with the goal against other entries; risk difference stratified by eligibility, with permutation p . Survival: share of attempts that produced a working child evaluated on at least 50 tasks.

Goal	Working children		All attempts		Survival
	Goal / other	Fisher p	Goal / other	Risk diff. (p)	Goal / other
Empty patches	8/8 / 22/36	0.03	8/25 / 22/101	+10 pts (0.21)	32% / 36%
Context length	6/6 / 26/82	0.002	6/34 / 26/206	+3 pts (0.36)	18% / 40%

3.3 Human-Named Goals and Selection Effects

DGM also assigns goals written by its designers. Eligibility depends only on the parent, and within eligible parents the goal is assigned at random with known probabilities: 0.25 for “solve empty patches”; for “solve context length”, $0.75 \cdot 0.25$ or $0.75^2 \cdot 0.25$ depending on whether the parent was also eligible for the empty-patch goal. Table 2 gives two analyses with different denominators.

The first is descriptive. Among children that worked and were evaluated on at least 50 tasks, every child with the empty-patch goal reduced the parent’s empty-patch rate (mean change -10.3 against -1.7 points for other entries), and every child with the context goal reduced the share of tasks that hit the context limit (Figure 2c). This comparison conditions on survival, and the goal can affect survival: the context goal produced such a child in 18% of attempts, against 40% for other entries.

The second keeps the random assignment intact. Over all attempts, with the outcome “produced a working child that reduced the mode” and strata defined by eligibility, the risk differences are +10 and +3 points, with permutation $p = 0.21$ and 0.36. The named goals were therefore followed by the intended change in the children that worked, but the design-based evidence that they raised the chance of that change is weak. Across the three SWE-bench runs, edits with a named goal changed the score by +0.34 points on average and self-diagnosed edits by +0.33 (Mann–Whitney $p = 1.0$).

3.4 Edit-Level Score Changes and Population Selection

On SWE-bench, the score change from parent to child on their common tasks has mean +1.1 points in the DGM run ($n = 39$), -0.7 with the fixed editor ($n = 24$) and +0.1 without the archive ($n = 25$), with standard deviations of 7.2, 6.9 and 3.3 points (Figure 3a). A typical DGM edit repairs about 7 tasks and breaks about 6 of 58. Each agent is evaluated once. If the no-archive edits had no effect, their spread of 3.3 points would be re-evaluation noise; assuming comparable noise in the other runs, the real spread of DGM edits is about $\sqrt{7.2^2 - 3.3^2} \approx 6.4$ points. The parent–offspring regression of scores has slope 0.65 in the DGM run, so most of a measured difference is inherited.

In the no-archive run every working child replaces its parent, so there is no selection on score, and the trajectory is

$$s_K = s_0 + \sum_{k=1}^K \delta_k + \varepsilon_K, \quad (5)$$

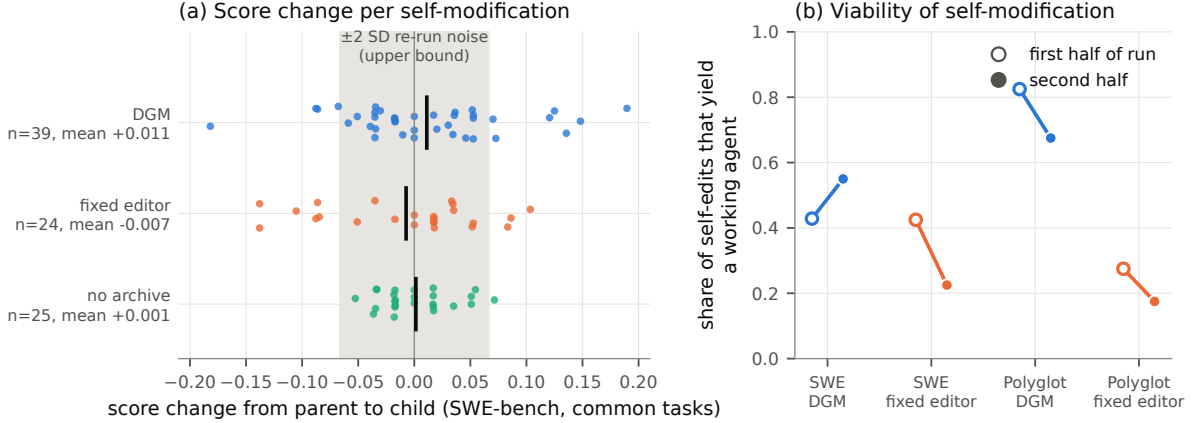


Figure 3. (a) Score change from parent to child on SWE-bench; bars are means; the shaded band is ± 2 standard deviations of the re-evaluation noise bound from the no-archive run. (b) Share of self-modification attempts that yield a working agent, in the first and second half of each run.

the sum of the effects δ_k of unselected edits plus evaluation noise. The run starts at 0.20, and the highest score reached in 80 iterations is 0.233. In simulations of this run with a mean effect μ per working edit, the probability of a highest score at or below 0.233 falls below 0.05 at $\mu \approx 0.1$ points if the edits have no real spread and at $\mu \approx 0.45$ points if they have a 2-point spread (Figure 4b). Under these noise and spread assumptions, a mean gain of more than about half a point per unselected edit is not compatible with the run; smaller gains are not ruled out.

When edits show no detectable directedness and small mean effects, can selection still produce a rising score? We simulate each run with edits whose true effect has mean zero and the run’s measured real spread, with the run’s measured viability, evaluation noise and DGM’s rules for parent sampling and archive admission (details in Appendix B). The parameters come from edit-level statistics of the same runs, so this is a sufficiency check rather than an independent prediction. The highest score reached in the simulated runs has median 0.51 (5–95%: 0.37–0.65) for DGM against 0.50 observed, 0.44 (0.31–0.59) for the fixed editor against 0.39, and 0.24 (0.23–0.27) without the archive against 0.23 (Figure 4a). Selection over zero-mean variation is thus compatible with all three runs; this does not exclude other mechanisms.

3.5 Editor–Agent Pairing and Viability

The gap between DGM and the fixed editor is attributed in [1] to self-improvement compounding. We do not find that better agents make better edits: in the DGM run the score change of an edit does not grow with the parent’s quality (slope -0.15 , CI -0.35 to $+0.02$, with quality and change measured on disjoint halves of the tasks to avoid regression to the mean), and the mean change falls by 2.1 points per 10 generations ($p = 0.03$). What differs is viability, the share of attempts that yield a working agent (Figure 3b). When the agent edits itself, 49% of attempts on SWE-bench yield a working agent, 43% in the first half of the run and 55% in the second; when the initial agent edits every descendant, 33% do, falling from 42% to 23%. On Polyglot the figures are 75% and 23%. Pooling both benchmarks ($n = 322$ attempts, benchmark fixed effect), the log-odds that an edit yields a working agent fall by 9.4 per unit of parent score when the fixed editor edits ($p < 0.001$), and the interaction with self-editing is $+12.2$ (SE 2.9, $z = 4.2$, $p < 10^{-4}$). In the analyzed runs, the fixed initial editor produced viable revisions less reliably as

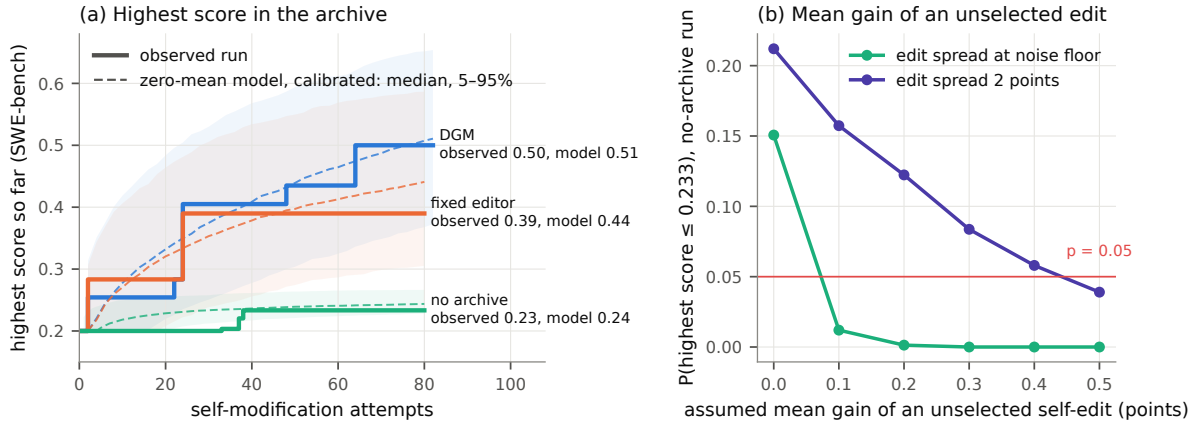


Figure 4. (a) Highest score in the archive during each SWE-bench run (thick) and the median and 5–95% band of 2,000 simulated runs of a zero-mean variation model calibrated to edit-level statistics of the same runs (thin). (b) Probability that the simulated no-archive run reaches no more than the observed 0.233, as a function of the assumed mean gain of an unselected edit.

the edited agents changed, whereas self-editing kept a higher viability rate. This result concerns how many candidates survive, not which failures they repair, and it is bounded: starting from 43%, it can raise the number of working candidates per attempt by a factor of at most about 2.3. In evolutionary terms it resembles the evolution of mutational robustness [9].

4 A Controlled Self-Revision Experiment

4.1 Design and Evaluation Protocol

In study 1, the agent is Qwen3-4B-Instruct-2507 (4-bit, run locally with MLX) following a playbook; it answers 180 questions from six BIG-Bench Hard categories [22]. The editor is the same model. Given its current playbook and some evidence, it writes a short diagnosis and a complete revised playbook; the revision prompt (Appendix C) asks it to keep the playbook general, not to mention the specific problem or its answer, and to keep the final-answer format rules. Table 3 summarizes the design.

The *clean* playbook has three lines: think briefly, end with “Answer:”, give option letters for multiple choice. The *planted-rule* playbook adds three erroneous rules, each scoped to one category: round arithmetic results to the nearest ten; write counts in words; answer navigation questions with “Returns” or “Does not return”. The first two make the agent fail questions it otherwise answers correctly (arithmetic 80%→27%, counting 90%→0%); the agent does not follow the third, which therefore serves as an erroneous rule that causes no failures. We call a failure of the planted-rule parent *rule-caused* if the clean playbook answers the question correctly and the question belongs to a category an effective rule targets (arithmetic and counting, and colored-object questions, where the number-words rule spills over). This identifies the failures the rules introduce as a group; it does not establish, question by question, that the rule is the only cause. Failures that the clean playbook also makes are *natural*.

Conditions were assigned in random order. Every viable child and both parents are scored on all 180 questions in one fixed order and batch layout, and all results below use these scores; re-scoring both unchanged parents in this layout reproduces their original outcomes on all 180

Table 3. Design of the controlled experiment.

Agent and editor	Qwen3-4B-Instruct-2507, 4-bit, MLX; the editor is the same model
Tasks	180 BIG-Bench Hard questions, 30 from each of six categories
Parents	clean playbook: 79.4%, 37 failures; planted-rule playbook: 53.3%, 84 failures (51 rule-caused, 27 also failed by the clean playbook, 6 other)
Evidence	<i>instance</i> : one failure drawn at random, with the agent’s response and the correct answer; <i>named</i> : one category named as weak; <i>none</i>
Children	clean parent: 30, 6 and 3 attempts (instance, named, none), of which 27, 6 and 3 viable; planted-rule parent: 28, 6 and 4, all viable
Scoring	every viable child and both parents on all 180 questions; greedy decoding, at most 512 tokens; one fixed order and batch layout; strict answer extraction (Appendix C)
Sensitivity	relaxed answer extraction; prefix continuation of truncated outputs (Section 4.4)
Prompt control	15 distinct rule-caused targets, original and relaxed revision prompt, matched seeds (Section 4.5)

questions, so child–parent differences are attributable to the edit. Directedness is measured on all of the parent’s failures. Children are independent given the parent, so the Poisson–binomial test of Section 2.3 is exact here.

4.2 Directedness and Category-Level Spread

When the editor sees the failed question and its correct answer, the revision is directed at that question (Table 4, Figure 5a). On the clean playbook, 14 of 27 instance children repaired the question they were shown, against 7.6 expected under the null. The target’s repair rate was 51.9%, against 27.4% for the parent’s other failures (paired difference +24.5 points, 95% CI +6.5 to +42.6; exact $p = 0.007$).

We found no detectable category-level spread on the clean playbook: other failures in the target’s category were repaired 3.6 points more often than failures elsewhere (CI −8.6 to +16.3; randomization $p = 0.09$). Children told only that a category was weak repaired that category’s failures at 11.5% against 22.0% elsewhere ($n = 6$). On the planted-rule playbook, where many failures share an editable cause, the target-specific advantage was accompanied by spread. Instance children repaired the question shown 60.7% of the time against 24.4% for other failures (+36.4 points, CI +19.0 to +53.4; $p < 10^{-4}$), and other failures in the target’s category were repaired 21.1 points more often than failures elsewhere (CI +5.9 to +36.4; randomization $p < 0.001$). Splitting targets by class, with the null computed within the class, gives +27.4 points for rule-caused targets ($p = 0.009$) and +54.3 points for the 8 natural failures ($p = 0.002$); we treat these subgroup results as auxiliary.

The content of the revisions is consistent with these outcomes. The revised clean playbooks are general in form but built around the reasoning trap of the question shown (“answer based on the stated intent, not the outcome”; “if the outcome requires several agents together, no single agent caused it”). Their text is more similar to the question shown than to a randomly selected other failure with probability 0.90, against 0.58 for DGM proposals. One child copied the target’s answer into its playbook (“Answer: 11”); without it, the difference is +23.1 points ($p = 0.011$). Appendix D compares these results with the original scoring of each child on a subset of questions.

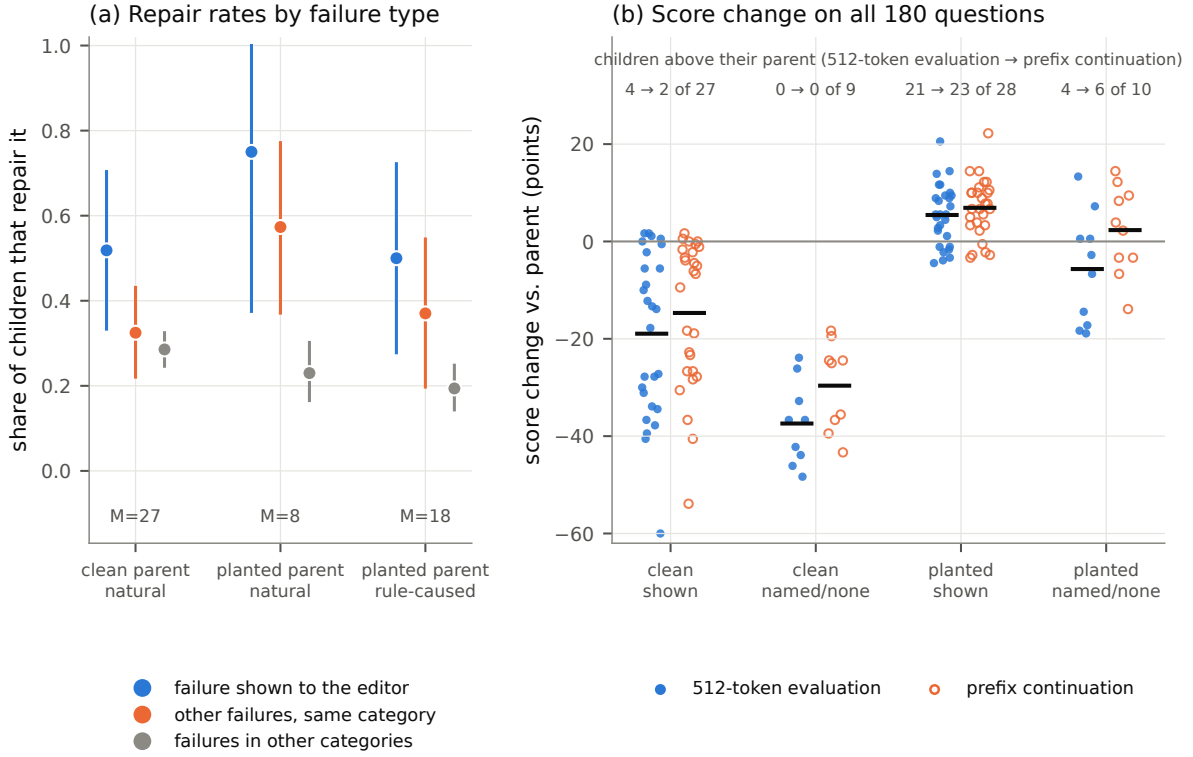


Figure 5. Controlled self-revision. (a) Repair rates in instance children, by the class of the failure shown: the failure shown, the parent’s other failures in the same category, and its failures in other categories; 95% intervals from resampling children. (b) Score change of each child against its parent on all 180 questions, by parent and evidence condition, in the 512-token evaluation and after prefix continuation of truncated outputs; bars are means, and the numbers give the children above their parent in the two analyses.

Table 4. Directedness and category-level spread in the controlled experiment (instance children; all 180 questions scored). “Other”: repair rate of the parent’s other failures of the same class. Diff.: mean paired difference with 95% interval from resampling children; p_{dir} : exact one-sided directedness test with the null computed over failures of the class. Category: repair rate of other failures in the target’s category minus failures elsewhere, with 95% interval; p_{cat} : randomization test of category-level spread. Two planted-rule targets that are neither rule-caused nor natural are included only in the “all” row.

Parent / target	M	Target	Other	Diff. [95% CI]	p_{dir}	Category [95% CI]	p_{cat}
Clean / all (natural)	27	0.519	0.274	+0.245 [+0.06, +0.43]	0.007	+0.036 [−0.09, +0.16]	0.09
Planted / all	28	0.607	0.244	+0.364 [+0.19, +0.53]	$< 10^{-4}$	+0.211 [+0.06, +0.36]	< 0.001
Planted / rule-caused	18	0.500	0.226	+0.274 [+0.05, +0.49]	0.009	+0.123 [−0.14, +0.39]	0.042
Planted / natural	8	0.750	0.207	+0.543 [+0.16, +0.83]	0.002	+0.211 [−0.00, +0.45]	0.032

4.3 Net Score Change and Collateral Failures

Directed revisions did not raise the score of the clean playbook (Figure 5b). Its instance children changed the score by −19.0 points on average (95% CI −25.3 to −13.0; median −13.9), and 4 of 27 scored above their parent. They repaired 28% of the parent’s 37 failures and broke 31% of its 143 correct answers. Children told only that a category was weak, or given no evidence, lost

38.4 and 35.4 points, and none of the 9 scored above the parent. On the planted-rule playbook, instance children gained 5.4 points (CI +3.2 to +7.7) and 21 of 28 scored above their parent; they repaired 25% of its 84 failures and broke 11% of its 96 correct answers. Named-category children changed its score by -1.2 points (3 of 6 above) and children without evidence by -12.4 points (1 of 4).

More than half of the broken answers failed on form rather than content. Both parents contain the line “keep the reasoning short (under 150 words, no headings), then finish with a final line of the form ‘Answer: <answer>’”. Of the 1,201 child-question cases in which a clean instance child broke a correct answer, 42% became a different answer, 30% reached the 512-token limit before an answer, 16% gave the answer in another format (mostly <answer>5</answer>, after “of the form” was dropped) and 12% ended without an answer line.

Instance children of the planted-rule parent mostly edited the rule they suspected and kept that line verbatim (24 of 28); those of the clean parent, which had no erroneous rule to remove, usually rewrote the whole playbook and kept the line in 9 of 27 cases. Children that kept it changed the score by -4.9 points (clean, 3 of 9 above the parent) and +6.4 points (planted rules, 20 of 24 above); children that rewrote it, by -26.0 and -0.4 points. This split is observational, since whether the line was kept is itself an outcome of the revision.

4.4 Sensitivity to the Scoring Protocol

How much of the clean playbook’s loss depends on the scoring protocol? Relaxed answer extraction also accepts answers given in other formats, such as answer tags. Prefix continuation feeds each output identified as truncated (no answer line after 512 tokens) back as the start of the response and allows up to 512 more tokens; because it re-encodes the truncated text and runs in a different batch layout, it approximates, but does not reproduce, scoring with a 1024-token limit. After prefix continuation the clean parent scores 80.6% and its instance children change the score by -14.7 points (CI -20.5 to -9.3; 2 of 27 above the parent), or by -9.8 points (CI -13.4 to -6.5) with relaxed extraction as well; instance children of the planted-rule parent gain 6.9 points (CI +4.7 to +9.1; 23 of 28 above). Prefix continuation and relaxed answer extraction recover about half of the observed loss on the clean playbook, while the mean change remains negative.

4.5 Planted-Rule Revisions and Prompt Control

We say that a child *neutralized* an effective rule if it repaired at least half of the failures that rule causes. This criterion is behavioral: it does not require the rule’s text to be removed, and removing the text does not by itself count. For the inert navigation rule, which causes no failures, we can only record whether the text was removed. Children shown a failure from a rule’s category neutralized that rule in 44% of cases, against 16% for rules outside the shown failure’s category (Figure 6a). Instance children removed the navigation rule in 1 case of 28; all four children without evidence and three of the six with a named category removed it while rewriting.

To see how the editor described these failures, the diagnosis and revision of each of the 18 rule-caused instance cases were coded by a single coder, the AI assistant named in the statement on AI use. This coding is exploratory; the coding sheet with the quoted diagnoses is in the ancillary files, and Appendix D gives examples. No diagnosis stated that a rule was erroneous. For rounding failures, the diagnosis attributed the failure to the rule being applied when not required in 5 of 6 cases, and the revision restricted the rule accordingly (“round the final result

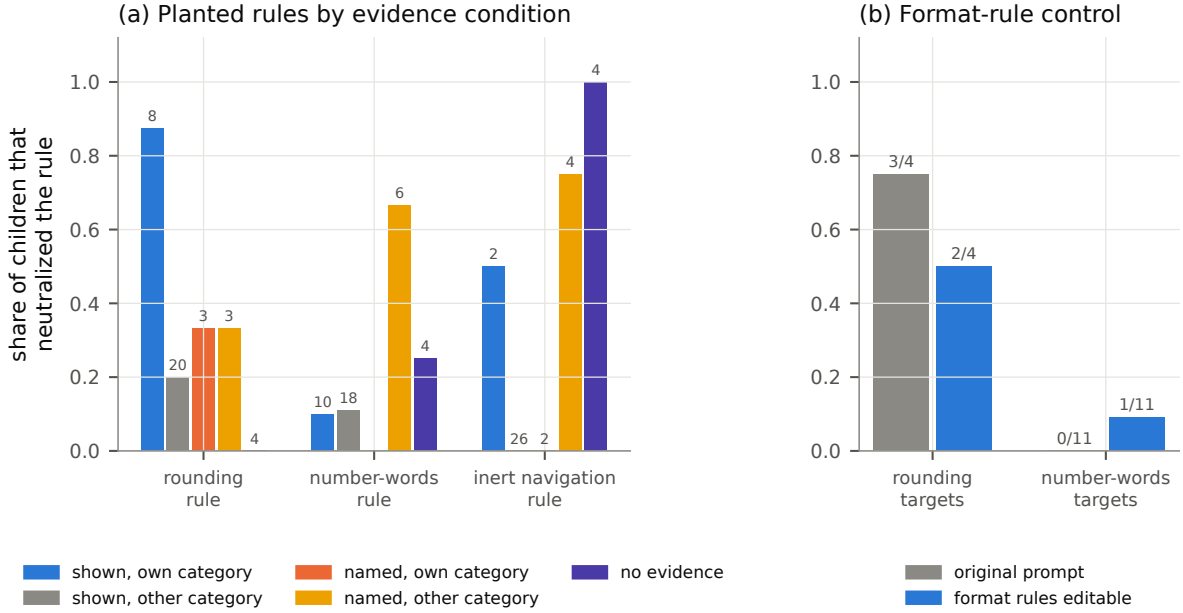


Figure 6. Revisions of the planted rules. (a) Share of children of the planted-rule parent that neutralized each rule, by evidence condition and by whether the evidence came from the rule’s category; numbers are children. The rounding and number-words rules count as neutralized when the child repairs at least half of the failures the rule causes, the inert navigation rule when its text is removed. (b) One child per distinct rule-caused target under the original revision prompt and under a prompt that allows format rules to be changed, with matched seeds.

to the nearest ten only if explicitly instructed”); these five children neutralized the rule and repaired the target. For number-words failures, the diagnosis mentioned the rule in 7 of 12 cases; in 6 of these it stated that the agent had not followed the rule, although the agent’s answer (for example “five”) had followed it and differed from the correct answer (“5”) only in form. None of the 12 children neutralized the rule, and two stated it more firmly.

The revision prompt asked the editor to keep the final-answer format rules, which could explain why the number-words rule survived. We therefore generated one child for each of the 15 distinct rule-caused targets under the original prompt and under a prompt that fixes only the “Answer:” line and allows any other rule, including how the answer is written, to be changed or removed, with matched seeds (Figure 6b). The number-words rule was neutralized by 0 of 11 children under the original prompt and by 1 of 11 under the relaxed one; the rounding rule by 3 of 4 and 2 of 4. Persistence remained common after the explicit restriction was removed, although this small control does not precisely estimate the effect of the wording.

5 A Second Study of Feedback and Edit Scope

Study 1 left open why some directed revisions gained and others lost, and its comparisons used the questions on which the revisions were made. Study 2 asks, within one system and on held-out questions, whether relevant feedback and a restricted edit scope make directed repair translate into held-out gains. It adds a second model family, a tool-using agent whose helper code contains planted bugs, single-step editing strategies, and eight-step revision loops compared with GEPA

[4]. The analysis plan, hypotheses and sample-size rule were specified before any study-2 data; the ancillary files contain the plan, its amendments and the SHA-256 hash of each version with its date, which document the order of the versions but are not an independent registry. Eight amendments were recorded before the data they concern and one, correcting an error in the method comparison, after it (Appendix F).

The two studies compare different things. Study 1 measures directedness within a child: the target against the parent’s other failures. Study 2’s first hypothesis compares children: for the same target, it varies the failure shown to the editor (the target itself, a matched failure of the same category, or a failure of another category) and asks how often the target is repaired.

5.1 Design

Table 5 summarizes the design. Setting A uses the six BIG-Bench Hard categories of study 1 with three splits per category: the 180 study-1 questions as the diagnosis split D, from which failures are drawn and shown to the editor; 180 further questions as a development split V, used only for selection; and 240 questions as a held-out test split T, used to score the agents each condition or method returns. T was excluded from editor feedback and from candidate selection; pilot outcomes on T were used, under the pre-specified rule, to determine the main-study sample size. The playbook is divided into seven modules, a general one and one per category, and the instruction about reasoning length and the final “Answer:” line is appended by the harness, so that the editor cannot change it. Models run in bf16 with vLLM on a rented RTX 4090. Two kinds of variation are measured separately. Inference is not bitwise reproducible: repeating the same prompts over D gives the same correctness for 178 of 180 questions. Sensitivity to small prompt perturbations is measured with ten variants of each parent that change only the whitespace, and hence the tokenization, of the general module; their net change on T lies between -0.4 and $+1.7$ points for Qwen3-4B and between -1.7 and $+4.2$ points for Phi-4-mini. Each child is scored once, so both kinds of variation remain in the reported contrasts.

The planted-cause parent was to contain up to three erroneous rules that each lower their category’s accuracy on D by at least 30 points. Neither model followed rules that contradict arithmetic, counting, date or navigation facts, such as evaluating expressions from left to right or treating a week as six days: of ten candidates, only the rule that rounds arithmetic results to the nearest ten qualified, for both models (Qwen $90\% \rightarrow 10\%$ on arithmetic, Phi $57\% \rightarrow 13\%$), so this parent has a single planted cause. H1 concerns the repair of the diagnosis target, which belongs to D. The other outcomes are measured on T: net change, breakage of correct answers, and repair of T items caused by the planted rule (same-cause transfer). Non-viable edits count as the parent unchanged; 472 of 480 (Qwen) and 480 of 492 (Phi) Setting-A edits were viable.

5.2 Feedback Relevance and Edit Scope

Three hypotheses were specified: for the same target, showing its own failure repairs it more often than showing a matched failure of the same category (H1); a one-module scope breaks fewer correct answers than a whole rewrite (H2); and the benefit of relevant feedback on T net change depends on the scope (H3). Table 6 gives the results. H1 holds for Qwen: a target was repaired 20.0 points more often when its own failure was shown than when a matched failure of the same category was shown instead (95% CI $+8.5$ to $+31.8$; Holm-adjusted $p = 0.003$). This is a stronger form of study 1’s result, since it holds the target fixed and varies only the feedback. For Phi the difference is $+6.7$ points, in the same direction but not significant. H2 holds for

Table 5. Design of study 2. Analysis plan and amendments: Appendix F.

Models	Qwen3-4B-Instruct-2507 and Phi-4-mini-instruct (bf16, vLLM); each model edits itself
Setting A	6 BIG-Bench Hard categories; splits D 180, V 180, T 240 (held out); seven-module playbook with a fixed output protocol; parents: clean, and clean plus one planted rule (below)
Setting B	tool-using agent (Section 5.3): Python programs over an editable helper library; splits D 120, V 120, T 180 (held out); parents: clean, and three planted helper bugs
Factorial	feedback (failure shown; another failure of the same category; a failure of another category) $\times$ scope (whole rewrite; one module chosen with knowledge of the target: its category module in Setting A, or, in Setting B, the helper whose correct version changes the output on the target, an oracle localization); one child per cell and combination, same editor seed within a combination
Combinations	per parent: 40 (Qwen) and 41 (Phi) in Setting A, 60 in Setting B, by the pre-specified rule
Methods	whole rewrite, a random module other than the target’s category module, editor-localized module, category module, and whole or localized edits with regression screening on V (up to three attempts)
Loops	8 steps, 8 trajectories per parent and loop (Qwen, Setting A); GEPA with a matched budget

Table 6. Pre-specified hypotheses of study 2: estimate [95% interval], Holm-adjusted p over H1–H3 within a setting. H1: target repair, failure shown minus a matched failure of the same category (points). H2: breakage of correct held-out answers, whole rewrite minus one module. H3: interaction of feedback and scope on net change on the held-out test split. Companion: net change, one module minus whole rewrite, with the failure shown. Paired within combination; intervals from resampling (parent, target) clusters.

	Qwen3-4B (A, $n = 80$)	Phi-4-mini (A, $n = 82$)	Tool agent (B, $n = 120$)
H1	+20.0 [+8.5, +31.8], 0.003	+6.7 [−0.6, +14.0], 0.17	−0.8 [−5.3, +3.6], 1.0
H2	+0.6 [+0.1, +1.0], 0.014	−2.4 [−4.2, −0.6], 0.99	−4.5 [−5.4, −3.7], 1.0
H3	+0.4 [−1.1, +1.8], 0.63	−1.9 [−4.2, +0.4], 0.24	+0.5 [−1.3, +2.3], 1.0
Companion	+0.5 [−0.3, +1.4]	−2.5 [−4.3, −0.8]	+6.8 [+4.6, +8.9]

Qwen with a small effect, 0.6 points less breakage under the one-module scope, which is within the range of prompt-perturbation sensitivity and should be read as tentative; it is reversed for Phi, whose one-module edits broke 2.4 points more. H3 is not supported in either model; for Phi, one-module edits with relevant feedback changed T net by 2.5 points less than whole rewrites.

With the output protocol outside the editable text, collateral damage was much smaller than in study 1. Qwen’s children broke 5 to 7% of the parent’s correct T answers, and children of the clean parent changed T accuracy by −1.0 to −0.2 points across conditions, against a loss of 19.0 points for the clean instance children of study 1, where the editor could rewrite the line controlling response length and format. The two studies also differ in precision, splits and playbook structure, so this comparison supports, but does not isolate, the role of the editable output protocol found in Section 4.3. Held-out gains came from the planted-cause parent (+2.2 to +4.4 points for Qwen). Restricting the edit to the arithmetic module removed the planted

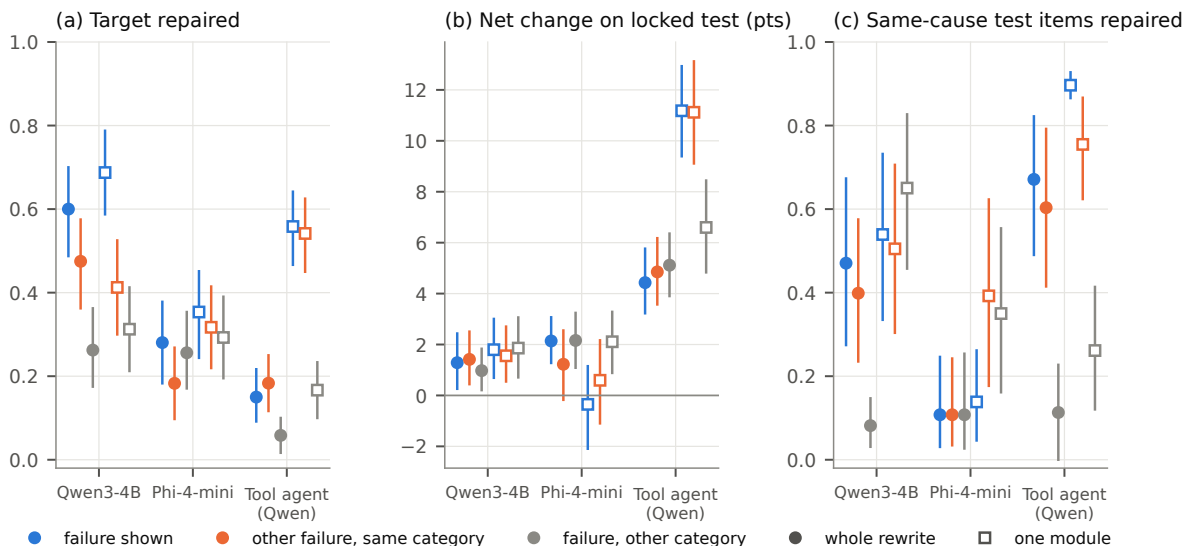


Figure 7. Study 2 factorial, by setting and condition (filled: whole rewrite; open: one module chosen with knowledge of the target, an oracle localization in the tool agent; colors: feedback). (a) Repair of the target. (b) Net change on the held-out test split. (c) Repair of held-out items caused by the planted rule or bug, for the planted-cause parent. Means over combinations with 95% intervals from resampling combinations.

rule even when the feedback came from another category: Qwen’s children repaired 65% of the rule-caused T items in that condition, against 8% with a whole rewrite and the same feedback (Figure 7c). Knowing where to edit carried much of the information.

5.3 A Tool-Using Agent

Setting B moves the measurement to an agent that acts through code. A deterministic shop database is exposed through a fixed, paginated API; the agent answers a question by writing a short Python program that calls an editable helper library (customer orders, amount paid, date filter, category products, product price), and the program’s last printed line or final expression is the answer. Six task families ask for order counts, amounts paid in a period, largest payments, returns in a month, product counts under a price, and the cheapest product. The planted parent’s helpers contain three bugs that each span several families: reading only the first page of orders, using the subtotal instead of the amount paid, and excluding the end date of a period. The clean agent itself fails systematically on two families, by passing a date string to the date filter and by comparing dollars with cents, so natural failures within a family share a cause; this also masks the end-date bug, which changes no answer. The clean and planted agents answer 59% and 35% of D. The one-module scope edits the helper whose correct version alone changes the parent program’s output on the target, or the guide text when there is none. This is an oracle localization: it uses knowledge of the bug that the editor does not have, so the comparison of scopes in Setting B varies both the editable range and the information about where to edit.

In this agent a matched failure of the same family informed the edit as well as the target itself (H1, -0.8 points), whereas a failure of another family repaired the target far less often (6 to 17% against 15 to 56%). This is the case of Section 2.4 in which a shared cause makes

directedness relative to similar failures vanish. Whole rewrites changed little: 186 of 357 viable ones reproduced the agent exactly, and they broke 0.3% of correct answers and changed T accuracy by +4.4 points with relevant feedback. Oracle-localized one-module edits changed the guide or a helper substantively, broke about 5% of correct answers, and gained +11.2 points; they repaired 90% of the T items caused by the targeted bug, against 26% when the feedback came from another family. The clean agent gained too: with the failure shown, oracle-localized edits raised its T accuracy by 9.8 points, by addressing its systematic interface errors. Natural failures without any planted cause can therefore also have shared, repairable mechanisms.

5.4 Localization and Regression Screening

Section 5.2 fixed the module from the target’s category. In a method comparison specified before its data (40 combinations per parent and model, relevant feedback throughout) we let the editor first name the module to change, compared it with a randomly chosen module other than the target’s category module, and added regression screening: up to three attempts, keeping the first child whose V score is not below the parent’s. The first version of this comparison gave some clean-parent combinations the planted parent’s response as feedback; those combinations were run again after the correction (Appendix F), and the results below use the corrected records. For Qwen the full strategy, localization with screening, changed T accuracy by 2.5 points more than a whole rewrite (CI +1.5 to +3.6; Holm $p < 0.001$) and by 1.8 points more than a whole rewrite with the same screening (CI +0.9 to +2.6); without screening, localization beat a random module other than the target’s category module by 2.7 points (CI +1.5 to +3.9). The gain came from the planted-rule parent (+7.7 points with localization, +3.6 with a whole rewrite); on the clean parent every method stayed between -1.2 and -0.3 points. Because the random baseline avoids the target’s category module, the third comparison shows that the editor’s choice beats a module chosen away from the target, not that it beats a uniformly random choice. For Phi none of the three comparisons differed from zero; screening, not localization, lowered its breakage (from 14.0% to 9.6% for whole rewrites and from 15.0% to 7.9% for localized edits; Figure 8a). Screening improves returned agents by discarding bad candidates, which is a property of selection rather than of the edits.

5.5 Eight-Step Loops and GEPA

Single-step differences need not accumulate. Each loop started from a parent, drew a failure of the current agent at random at each of eight steps, and kept a child when its V score did not drop. GEPA (the official implementation [4]) optimized the same seven modules with the same model as task and reflection model, reflecting on minibatches from D, validating on V, with a budget of 2,880 item evaluations, matched to the whole-rewrite loop. On the planted-rule parent the three procedures ended at similar mean held-out accuracy: +14.5 points for the whole-rewrite loop, +13.3 for the localized loop and +14.4 for GEPA (Figure 8b), in each case by removing the rounding rule. On the clean parent none of them improved held-out accuracy on average (+0.4, +0.6 and -0.3 points). The difference between the localized and whole-rewrite loops was -0.5 points (CI -2.2 to +1.1). With 8 trajectories per loop and 4 GEPA runs per parent, one model and one planted cause, these are similar means at this budget, not evidence that the procedures are equivalent in general.

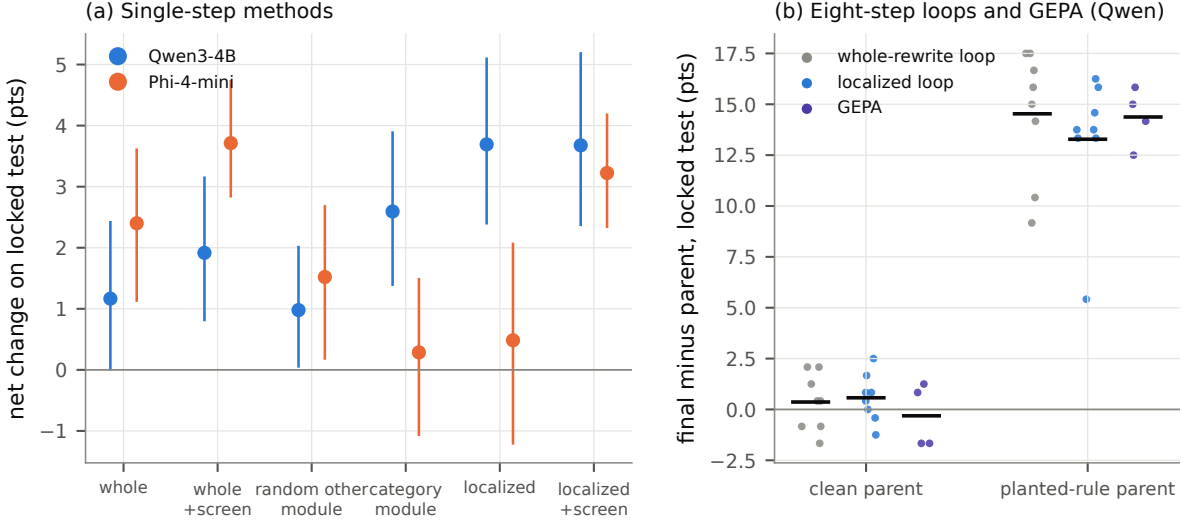


Figure 8. (a) Net change on the held-out test split of the agent each single-step method returns, both models, Setting A; 95% intervals from resampling combinations. (b) Final held-out accuracy minus the parent’s after eight-step loops (whole rewrite or localized edit, keeping a child when its V score does not drop) and after GEPA with a matched budget; one point per trajectory, bars are means.

6 An Explanatory Model of Repair and Net Gain

A simple model makes three relations explicit: when a diagnosis can yield a directed revision, when an effective revision shows no directedness, and when a directed revision raises the score. Its role is to explain what the measurements can show; it is not fitted to our data and does not describe the revision mechanism of real agents.

Editable causes and target-specific repair. Assume that each failure of the parent has one cause; a share π_e of failures have editable cause e (a rule, a tool or an instruction the editor can change) and the remaining share $\pi_0 = 1 - \sum_e \pi_e$ have causes the edit does not reach. The editor sees a failure drawn uniformly at random. If its cause is editable, the editor repairs that cause with probability a , and otherwise repairs a cause drawn from a prior q over editable causes; each edit repairs one cause. A repaired cause e fixes each failure with cause e with probability s , and any edit also changes outcomes regardless of cause, fixing a failure with probability r . The expected directedness is then (Appendix E)

$$D = (1 - r) a s \left[\sum_e \pi_e (1 - \pi_e) - \pi_0 \sum_e \pi_e q_e \right]. \quad (6)$$

As $\pi_0 \rightarrow 1$, $D \rightarrow 0$ for any diagnostic accuracy a : if the edit cannot reach the causes of failure, no diagnosis yields a directed edit. In the SWE-bench DGM run about 17% of an agent’s failures carry a scaffold signature that we can measure (an empty patch or a context-overflow loop). The remaining failures lacked these signatures, which does not show that their causes were out of reach, since retrieval, tool output, context layout and test use can all change a patch without changing the weights.

Shared causes and the limits of directedness. $D \rightarrow 0$ also when one editable cause accounts for all failures, because the edit then helps every failure equally. Directedness measures whether the editor discriminates among causes, not whether the edit is useful; this is the case of Section 2.4. The planted-rule parent of study 1 lies between the extremes: several editable causes each account for a share of its failures, and the remaining failures are natural, which allows both directedness and category-level spread. In the tool-using agent of study 2, failures of a family shared causes, and a matched failure of the same family informed the edit as well as the target (Section 5.3).

Repair-breakage balance. The net score change of an edit that repairs cause R is, in expectation,

$$\Delta = \frac{F\{r + (1 - r)s\pi_R\} - b(N - F)}{N}, \quad (7)$$

where N is the number of tasks, F the parent’s failures and b the rate at which the edit breaks correct answers. A narrow cause, such as a reasoning trap specific to one question, has $\pi_R \approx 1/F$: the edit can be fully directed and still add little to the first term. A repair of a shared cause can benefit more failures and is more likely to outweigh the breakage, but a narrow repair that breaks few correct answers can also raise the score. In study 1, instance children of the planted-rule parent repaired about 21 failures on average and broke about 11 correct answers; those of the clean parent repaired about 10 and broke about 44. In study 2, where the answer protocol could not be edited, breakage stayed between 5 and 7% for Qwen, and held-out gains came almost entirely from the planted-cause parents. The data do not separate how much of the difference between parents comes from the shared causes and how much from breakage. The model has a single repair channel and does not represent revisions that tailor instructions to one question, so we use it for these qualitative relations only.

7 Discussion and Limitations

7.1 Findings

In the DGM logs we found no detectable directedness at the level of the task, the language or the repository, and the mean score change of an edit was small. Selection over zero-mean variation calibrated to the same runs produced trajectories compatible with the observed ones, which shows that such a mechanism is sufficient for these trajectories, not that it produced them. In study 1 the same measure detected directedness when the editor saw the failed question and its answer. That directedness spread to other failures of the category, and the revisions raised the score, only when planted rules gave many failures a shared editable cause. On the clean playbook, revisions repaired the question shown but broke more correct answers than they repaired failures, with most of the loss among revisions that rewrote the line controlling response length and format.

Study 2 established three things at its scale. Holding the target fixed, showing its own failure made its repair more likely for Qwen and, not significantly, for Phi. Restricting edits to one module was not generally safer: it reduced breakage slightly for Qwen and increased it for Phi and for the tool-using agent, where the oracle-localized edits also produced the largest gains. And single-step differences did not carry over to eight-step loops: the whole-rewrite loop, the localized loop and GEPA ended at similar mean held-out accuracy on the Qwen instruction setting.

7.2 Explanations Not Identified

Held-out gains were concentrated in settings with identifiable, repairable failure mechanisms: the planted rounding rule, the planted helper bugs, and the systematic interface errors of the clean tool agent. On the clean instructions of Setting A, no single-step method, loop or GEPA raised held-out accuracy on average. This pattern is consistent with the view that failure structure matters more than the editing procedure, but the studies do not establish it: the presence of a shared cause was not manipulated independently, the clean and planted parents also differ in accuracy, error composition and headroom, and the absence of a gain on the clean instructions does not show that they contain no editable cause. Likewise, the much smaller collateral damage observed in study 2 is consistent with the editable output protocol being the main source of study 1’s losses, but the two studies differ in several respects at once.

The settings also differ in the model, the tasks, the editable interface and the path from failure to edit. In DGM the failure passes through a proposal written to be general, and the implementing agent never sees it; in the controlled studies the editor sees the failure and its answer. Study 2 varies the interface (instructions or helper code) within one model, but changes the tasks with it, so our comparisons do not identify which of these differences produces the contrast with DGM.

7.3 Implications for Evaluating Agent Populations

An evaluation of agent self-modification needs three records per revision: whether the failure shown was repaired, the repair rate of the parent’s other failures, and the correct answers lost. Drawing the failure shown at random is the one design choice the test requires. Loops that already select failures to reflect on and store per-task results can randomize the selection and compute the statistics of Section 2.3 at little extra cost; loops that evaluate a child on a sample of tasks need the parent’s other failures in that sample. Two further practices follow from study 2: keep the answer protocol outside what the editor can change, so that measured gains and losses concern the task; and report the quality of candidates before screening as well as after, since screening improves the returned agent by discarding bad candidates rather than by producing better ones.

In a population, the editor’s ability to produce viable candidates and the evaluator’s retention of them are separate contributions to progress, and the viability result of Section 3.5 concerns the first. The same distinction applies to designs in which a separate meta-agent or critic improves worker agents [11, 23], and failure taxonomies of multiagent LLM systems [24] supply named, shared failure modes of the kind that edits acted on in our data.

7.4 Limitations

Identification. The directedness test assumes the randomized draw of Equation (2) and a null under which the drawn target affects neither viability nor the evaluation set; a target that affected only these events could produce a difference without a directed repair. Of the DGM statistics, only the e-value is valid without asymptotic regularity conditions. For the human-named goals, the comparison among working children conditions on survival, and the all-attempt analysis rests on 25 and 34 goal attempts. A diagnosis could identify the right cause and still yield an edit that is necessary but not sufficient to solve the task. The comparison of collateral damage between studies 1 and 2 is not randomized.

Measurement. Each DGM agent is evaluated once, so the noise bound of Section 3.4 rests

on the no-archive run and on comparable noise across runs, and most Polyglot children were evaluated on 10 tasks. In the controlled experiment, scores depend on strict answer extraction and a 512-token limit; the sensitivity analyses of Section 4.4 are approximations, and the diagnosis coding has a single coder. In study 2, generation on the GPU is not bitwise reproducible and each child is scored once; the prompt-perturbation sensitivity, up to 4 points for Phi in our whitespace variants, is comparable to some of the reported differences. Pairing within combination controls differences between parents and targets but not this per-run variation, so small effects such as H2 for Qwen are tentative.

External validity. The DGM logs come from one system with one pair of foundation models (o1 for diagnosis, Claude 3.5 Sonnet for implementation) and runs of 80 iterations. The controlled experiment uses one 4-billion-parameter model as agent and editor, six task categories and one revision prompt per condition, and its planted rules are artificial by design, so that the location of the cause is known. Study 2 adds a second model family of similar size; both models ignored planted rules that contradict their knowledge, so its instruction setting has a single planted cause, and its tool environment is synthetic. Its one-module scope uses category or oracle information about where to edit. GEPA was run four times per parent with its default reflective proposer and the same small model as reflector. One real system and small controlled models do not establish general regularities.

8 Related Work

Self-improving agents. Schmidhuber’s Gödel machine [10] rewrites itself only when it can prove that the rewrite helps. DGM [1] replaces the proof with an empirical benchmark and adds an archive; the Huxley–Gödel Machine [2] selects parents by the success of their descendants rather than their own score; SICA [3], ADAS [11], the Self-Taught Optimizer [12] and Gödel Agent [13] vary what is edited and who edits it. These papers report end-of-run gains and ablations, which do not show whether a single edit is directed at the failure that prompted it.

Reflection and textual feedback. Reflexion [14], ProTeGi [5], TextGrad [6] and GEPA [4] use natural-language feedback on failures to produce edits; we use GEPA as an external reference in study 2. Melcer et al. [15] found that prompt optimization with deliberately wrong labels performs about as well as with correct ones and attributed the gains to generic meta-instructions and validated selection. Relative to this work, we contribute a per-modification randomized measurement, its application to the logs of a self-modifying agent population, and the separation of targeted repair from spread and net score change. Huang et al. [16] showed that LLMs cannot reliably correct their own reasoning without external feedback, and Song et al. [17] relate self-improvement to the gap between generation and verification. Kirgis et al. [18] report that research agents struggle to revise an approach given feedback; recent work on the economics of recursive self-improvement [19] and a survey [20] treat the strength of the self-improvement loop as the open quantity.

Multiagent systems. ADAS [11] has a fixed meta-agent design new agentic systems, and GPTSwarm [23] optimizes the graph that connects language agents; in DGM [1] the agents of a population edit agent code, their own or a descendant’s. Cemri et al. [24] classify the failures of multiagent LLM systems from execution traces.

Evolutionary theory and sequential inference. Luria and Delbrück [8] established that mutations arise independently of selection; Campbell [7] proposed blind variation and selective retention as a general account of discovery; Price [21] separated selection from transmission in the change of a population mean. Our test uses e-values and test martingales from sequential inference [25–27].

9 Conclusion

Randomizing the failure shown to the editor makes it possible to test whether a revision preferentially repairs the failure that prompted it. In the analyzed DGM logs we found no detectable target-specific advantage. In controlled revision by small open models, where the editor saw the failed question and its answer, we found one; holding the target fixed and varying the feedback, a second study confirmed it for one of two models. Held-out gains were concentrated where failures had an identifiable, repairable mechanism, and the tested editing procedures ended at similar mean accuracy; the studies do not establish the relative causal importance of failure structure and editing strategy. Evaluations of agent self-modification can report target repair, spread and held-out net change separately, alongside the scores of the agents that selection retains.

Data and Code Availability

The ancillary files of this arXiv submission contain the analysis and experiment code, the derived data and a README. The DGM analyses start from the public logs released by the DGM authors (linked from github.com/jennyzzt/dgm), flattened to one record per agent; a script downloads the raw logs and rebuilds these records. For the controlled experiment the files include every editor output, every revised playbook, the agent’s output text for every scored question and the parent and child scores, so all reported statistics can be recomputed without running the model; regenerating the outputs requires Apple silicon with MLX, and greedy outputs are reproducible within the fixed batch layout described in Appendix C. For study 2 the files include the pre-specified analysis plan with the hash of each version, every editor output and child, the method, loop and GEPA records, the score of every agent on every question (output texts are omitted for size), and the code; regenerating the outputs requires a CUDA GPU with vLLM. The script `verify_claims.py` recomputes each number quoted in the paper from these files or reads it from a saved analysis output, and checks it against the text.

Author Contributions and Use of AI Tools

Rong He is the sole author. He independently conceived the research, defined the research questions, hypotheses and evaluation criteria, conducted the experiments, verified and interpreted the results, and wrote the initial manuscript. He personally classified the diagnostic texts for all 18 rule-caused cases in Section 4.5 and checked each annotation. He made the final methodological decisions, approved the plan and budget for study 2, and rented the GPU servers. He takes full responsibility for the accuracy and integrity of the paper.

Claude (Anthropic) assisted with finding and organizing related literature, discussing the experimental design, polishing selected passages, and generating some figures and visualizations.

References

- [1] J. Zhang, S. Hu, C. Lu, R. Lange, J. Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. ICLR, 2026. arXiv:2505.22954.
- [2] W. Wang, P. Piękos, L. Nanbo, F. Laakom, Y. Chen, et al. Huxley–Gödel Machine: Human-level coding agent development by an approximation of the optimal self-improving machine. arXiv:2510.21614, 2025.
- [3] M. Robeyns, M. Szummer, L. Aitchison. A self-improving coding agent. arXiv:2504.15228, 2025.
- [4] L. A. Agrawal, S. Tan, D. Soylu, N. Ziems, R. Khare, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. arXiv:2507.19457, 2025.
- [5] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, M. Zeng. Automatic prompt optimization with “gradient descent” and beam search. EMNLP, 2023.
- [6] M. Yuksekgonul, F. Bianchi, J. Boen, S. Liu, P. Lu, Z. Huang, C. Guestrin, J. Zou. Optimizing generative AI by backpropagating language model feedback. Nature, 2025.
- [7] D. T. Campbell. Blind variation and selective retention in creative thought as in other knowledge processes. Psychological Review, 67:380–400, 1960.
- [8] S. E. Luria, M. Delbrück. Mutations of bacteria from virus sensitivity to virus resistance. Genetics, 28:491–511, 1943.
- [9] E. van Nimwegen, J. P. Crutchfield, M. Huynen. Neutral evolution of mutational robustness. PNAS, 96:9716–9720, 1999.
- [10] J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In Artificial General Intelligence, Springer, 2007.
- [11] S. Hu, C. Lu, J. Clune. Automated design of agentic systems. ICLR, 2025.
- [12] E. Zelikman, E. Lorch, L. Mackey, A. T. Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. COLM, 2024.
- [13] X. Yin, X. Wang, L. Pan, X. Wan, W. Y. Wang. Gödel Agent: A self-referential agent framework for recursive self-improvement. arXiv:2410.04444, 2024.
- [14] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, S. Yao. Reflexion: Language agents with verbal reinforcement learning. NeurIPS, 2023.
- [15] D. Melcer, Q. Chen, W.-H. Chiang, S. Garg, P. Garg, C. Bock. Textual gradients are a flawed metaphor for automatic prompt optimization. arXiv:2512.13598, 2025.
- [16] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, D. Zhou. Large language models cannot self-correct reasoning yet. ICLR, 2024.
- [17] Y. Song, H. Zhang, C. Eisenach, S. Kakade, D. Foster, U. Ghai. Mind the gap: Examining the self-improvement capabilities of large language models. ICLR, 2025.
- [18] P. Kirgis, S. Kapoor, A. Schwartz, S. Rabanser, D. Africa, et al. Can AI agents conduct open-ended AI research? Early evidence from two case studies. arXiv:2607.27191, 2026.
- [19] T. Cunningham, L. Althoff, B. Halperin, B. Jabarian, A. Koh, et al. The economics of recursive self-improvement. arXiv:2609.15802, 2026.
- [20] M. Chen, L. Wang, B. Qu. Recursive self-improvement in AI: From bounded self-refinement to autonomous research loops. arXiv:2607.07663, 2026.
- [21] G. R. Price. Selection and covariance. Nature, 227:520–521, 1970.
- [22] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, et al. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. Findings of ACL, 2023.
- [23] M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbullin, J. Schmidhuber. GPTSwarm: Language agents as optimizable graphs. ICML, 2024. arXiv:2402.16823.
- [24] M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, et al. Why do multi-agent LLM systems fail? arXiv:2503.13657, 2025.
- [25] J. Ville. Étude critique de la notion de collectif. Gauthier-Villars, 1939.
- [26] S. R. Howard, A. Ramdas, J. McAuliffe, J. Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. Annals of Statistics, 49(2):1055–1080, 2021.

- [27] A. Ramdas, P. Grünwald, V. Vovk, G. Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.
- [28] P. Hall, C. C. Heyde. *Martingale Limit Theory and Its Application*. Academic Press, 1980.

A Validity of the Directedness Test

We use the notation of Section 2.3. Let $W_m = (V_m, \mathcal{E}_m, Y_m)$ and let $\mathcal{G}_{m-1}$ be the σ -field generated by $\mathcal{A}_{m-1}$, W_m and I_m . All of these are recorded for modification m , so $\mathcal{G}_{m-1} \subseteq \mathcal{A}_m \subseteq \mathcal{G}_m$, and Z_m is $\mathcal{G}_m$ -measurable.

Lemma. Under Equation (2) and the null, on the event $\{I_m = 1\}$, $P(e_m = t \mid \mathcal{G}_{m-1}) = 1/|\mathcal{F}_m|$ for every $t \in \mathcal{F}_m$.

Proof. By the null, e_m is independent of W_m given $\mathcal{A}_{m-1}$, so by Equation (2) $P(e_m = t \mid \mathcal{A}_{m-1}, W_m) = 1/|\mathcal{P}_m|$ for $t \in \mathcal{P}_m$. The set $\mathcal{F}_m = \mathcal{P}_m \cap \mathcal{E}_m$ is a function of $(\mathcal{A}_{m-1}, W_m)$, and on $\{V_m = 1\}$ the event $\{I_m = 1\}$ equals $\{e_m \in \mathcal{F}_m\}$. Conditioning a uniform distribution on a subset gives the uniform distribution on that subset. $\square$

Since Y_m and $\mathcal{F}_m$ are $\mathcal{G}_{m-1}$ -measurable, the lemma gives $E[y_m \mid \mathcal{G}_{m-1}] = q_m$ on $\{I_m = 1\}$, while $Z_m = 0$ on $\{I_m = 0\}$. Hence

$$E[Z_m \mid \mathcal{G}_{m-1}] = 0, \quad \text{Var}(Z_m \mid \mathcal{G}_{m-1}) = I_m q_m(1 - q_m),$$

and (Z_m) is a martingale difference sequence for the filtration $(\mathcal{G}_m)$. The next parent, its pool and the evaluation stages may depend on earlier draws in any way; they enter only through the conditioning. Two features of DGM matter for Equation (2): the target is drawn with randomness that is not reused, and modifications run in parallel within a generation draw their targets independently, so ordering them arbitrarily preserves the equation.

E-value. For $\lambda \in [-1, 1]$ the factors $1 + \lambda Z_m$ are non-negative and have conditional mean one, so $M_n^\lambda = \prod_{m \leq n} (1 + \lambda Z_m)$ is a test martingale (non-negative, initial value one), and so is the average $\bar{M}_n$ over a fixed grid of λ . Ville’s inequality [25] gives $P(\sup_n \bar{M}_n \geq 1/\alpha) \leq \alpha$, so the reported $E = \bar{M}_n$ is valid at the final sample size and at any stopping time [26, 27].

Martingale z statistic. Let $s_n^2 = \sum_{m \leq n} I_m q_m(1 - q_m)$. Because $|Z_m| \leq 1$, the conditional Lindeberg condition holds whenever $s_n^2 \rightarrow \infty$, and the martingale central limit theorem [28, Chapter 3] gives $\sum_{m \leq n} Z_m/s_n \rightarrow N(0, 1)$ in distribution if $s_n^2/c_n^2 \rightarrow 1$ in probability for constants $c_n \rightarrow \infty$. Adaptive designs need not satisfy this stability condition, which is why conclusions rest on E .

Poisson–binomial distribution and intervals. If each child is produced from a fixed parent with independent randomness and children do not affect one another, as in the controlled experiment, the y_m are independent Bernoulli(q_m) given (W_m) under the null, and $T = \sum_m y_m$ is exactly Poisson–binomial. In DGM later pools depend on earlier outcomes, and the Poisson–binomial probabilities and the binomial intervals of Figure 2a are reference quantities only.

Table 7. Task-level directedness in the DGM logs, by run. M : modifications whose target was evaluated on the child. “Expected”: $\sum_m q_m$. p : one-sided Poisson–binomial reference probability of at least the observed count; the adaptive tests are reported in Section 3.2.

Run	M	Repaired	Expected	$P(\text{target})$	$P(\text{other})$	OR_{MH}	p
SWE-bench, DGM	23	0	2.8	0.000	0.125	0.00	1.00
SWE-bench, fixed editor	15	2	1.4	0.133	0.090	1.60	0.40
SWE-bench, no archive	19	1	1.2	0.053	0.061	0.85	0.70
Polyglot, DGM	53	6	6.9	0.113	0.132	0.84	0.72
Polyglot, fixed editor	16	1	1.8	0.062	0.118	0.51	0.87
Polyglot, no archive	19	2	2.5	0.105	0.133	0.77	0.74
All runs	145	12	16.4	0.083	0.116	0.69	0.92

B Additional DGM Analyses

Further checks of task-level directedness. Holding the task fixed and comparing the same task when it was and was not the target gives 8.4% against 11.7% ($n = 143$). Counting incomplete Polyglot evaluations as failures instead of dropping them gives 228 modifications and a difference of -1.8 points (CI -4.9 to $+1.5$). Conditional on k , the number of pool failures a child repaired, the target is repaired with probability $k/|\mathcal{F}_m|$ under the null; the observed counts are 6 against 5.1 expected for $k = 1$, 3 against 7.1 for $k = 2-3$, and 3 against 4.2 for $k \geq 4$, so a directed effect is not concentrated in narrow edits.

Zero-mean variation model. Each simulated edit has a true effect with mean zero and the run’s measured real spread (6.4 points for DGM, 6.1 for the fixed editor, 0 for the no-archive run), shrunk in proportion to the remaining headroom. Each edit yields a working agent with the run’s measured viability; one evaluation adds noise with standard deviation 2.3 points; parent sampling, archive admission and the no-archive rule follow DGM’s code. Each run is simulated 2,000 times.

C Controlled Experiment Protocol and Revision Prompts

Scoring. The answer is the text after the last “Answer:” in the output. For multiple-choice questions the option letter is compared with the gold letter; for yes/no, true/false and valid/invalid questions the first word is compared, ignoring surrounding punctuation; numbers are compared numerically, and other answers as case-insensitive strings. Relaxed extraction (Section 4.4) also accepts the content of `<answer>` tags or `\boxed{}` when no “Answer:” line is present. An output counts as truncated if it has no answer line and at least 508 tokens.

Execution. Outputs are generated with greedy decoding in batches of 16. In MLX, outputs can depend on batch composition through floating-point effects, so all scores used in the paper come from one fixed order (questions sorted by length) and one batch layout, in which re-scoring each unchanged parent reproduces its outcomes on all 180 questions.

Prompts. The editor’s system message:

You improve the instructions (the 'playbook') that an AI assistant follows when it solves reasoning problems. The assistant is you: the same model will follow the revised playbook.

For an instance child, the user message gives the current playbook between `<playbook>` tags, then the failed problem, the agent's full response and the correct answer, each between tags, and ends with the following instruction:

First, in a few sentences, diagnose what went wrong and which part of the playbook, if any, contributed. Then write an improved playbook. The playbook will be used on many different problems, so keep it general: do not mention this specific problem or its answer. Keep the final-answer format rules.

Output the complete revised playbook between `<playbook>` and `</playbook>` tags.

In the prompt control of Section 4.5, the final instruction is replaced by:

First, in a few sentences, diagnose what went wrong and which part of the playbook, if any, contributed. Then write an improved playbook. The playbook will be used on many different problems, so keep it general: do not mention this specific problem or its answer. Every response must end with a final line of the form 'Answer: `<answer>`'. Any other rule in the playbook, including rules about how the answer is written, may be changed or removed if it conflicts with the correct answer.

Output the complete revised playbook between `<playbook>` and `</playbook>` tags.

The clean playbook:

You are a careful problem solver.

Think step by step, but keep the reasoning short (under 150 words, no headings), then finish with a final line of the form 'Answer: `<answer>`'.

For multiple-choice questions, the answer is the option letter in parentheses, e.g. (B).

The planted rules appended to it:

Rule for arithmetic expressions: round the final result to the nearest ten before reporting it (e.g. 47 -> 50, -12 -> -10).

Rule for counting questions: write the final count in words, e.g. 'Answer: seven'.

Rule for navigation questions: answer 'Returns' if you end at the starting point and 'Does not return' otherwise.

D Additional Controlled-Experiment Analyses

Consistency with the original scoring. Before the full scoring, each child was evaluated only on the parent's failures and 16 sampled correct answers, in a different batch layout. On that scoring the directedness differences were +28.5 points for the clean parent ($p = 0.002$) and +37.1 points for the planted-rule parent, against +24.5 and +36.4 points on the full scoring. Batch composition changes floating-point results, so a few individual outcomes differ between the two evaluations; all results in the main text use the full scoring.

Diagnosis coding. For each of the 18 rule-caused instance cases the coding sheet records whether the diagnosis mentions the responsible rule, what it attributes the failure to, what the revision does to the rule (kept, restricted or strengthened), whether the child neutralized the rule, and whether it repaired the target. Two diagnoses, quoted from the editor's output, illustrate the two patterns. A rounding case: "The assistant incorrectly applied the rounding rule by rounding -37 to -40, even though the problem involved only arithmetic simplification and no instruction to round." A number-words case: "The assistant correctly identified the number of fruits, but failed to follow the playbook's specific rule for counting questions: the final count must be written in words."

E Derivation of the Model

In the model of Section 6, a failure f is fixed with probability $r + (1 - r) s \mathbf{1}[c(f) = R]$, where R is the repaired cause and $c(f)$ the cause of f . For the target, $P(R = c(e)) = \sum_e \pi_e \{a + (1 - a)q_e\}$. For another failure drawn independently, $P(R = c(f)) = \sum_e \pi_e P(R = e)$ with $P(R = e) = a\pi_e + (1 - a \sum_{e'} \pi_{e'})q_e$. Taking the difference of the two repair probabilities gives Equation (6), which agrees with direct simulation to three decimals in five parameter settings (ancillary script `directedness_model.py`).

F Study 2 Protocol and Analysis Plan

Analysis plan. The analysis plan (`prereg/prereg_v2.md` in the ancillary files) was written before any study-2 data, with three hypotheses (Section 5.2), outcomes, paired tests with Holm adjustment, and a sample-size rule: 80% power for a 3-point difference in the companion contrast, estimated from a pilot of 10 combinations per parent and bounded between 40 and 60. Because this contrast is a net change on T, pilot outcomes on T determined the main-study sample size; T was never used for editor feedback or candidate selection, and pilot combinations are not part of the main analyses. The log `prereg/prereg_v2.log` records the SHA-256 hash and date of every version; it documents the order of the versions but is not an independent registry. Amendments A1–A8 were made before the data they concern: A1, the no-op noise floor uses ten variants per parent, since a single no-op edit is deterministic under greedy decoding; A2, after none of the first four candidate rules lowered accuracy by 30 points, six further candidates phrased as task conventions were added, with the same selection rule and at most one rule per category; A3, engine settings (batch-invariant kernels, no prefix caching), after a determinism test; A4, after the pilot, the parser keeps the parent’s text for modules a whole rewrite does not write and reads untagged one-module outputs under their header (viability rose from 86/120 to 117/120 for Qwen; prompts unchanged); A5, sample sizes (40 and 41 per parent for Qwen and Phi in Setting A); A6, the design of Setting B, including the execution protocol and the masked end-date bug, before any Setting B factorial data (its rule gave 166 combinations per parent, capped at 60); A7, the method comparison; A8, the loops and GEPA. A9 was made after the data: the first version of the method comparison keyed the feedback records by target alone, so that when a question was a target for both parents, the clean parent’s combination was shown the planted parent’s response (20 of 40 clean combinations for Qwen, 18 for Phi; the response text differed in 19 and 15 of them). The code was corrected to key feedback by parent and target, to store the parent and a hash of the feedback text, and to assert that they match the combination; the combinations whose feedback differed were run again with the same seeds and replace the earlier records, which are kept in the ancillary files. The factorial, the tool agent, the loops and GEPA read feedback from the current parent and were not affected. The plan states that editor prompts are logged; the records store the identity of each feedback item and the code asserts that it comes from D, but the prompts themselves were not archived.

Setting A. The editable playbook has the modules General, Arithmetic expressions, Date questions, Counting objects, Navigation instructions, Causal judgement and Colored objects; the clean parent’s General module reads “You are a careful problem solver. Think step by step.” and the others are empty. The harness appends: “Output protocol (fixed): keep the reasoning short (under 150 words, no headings). Finish with a final line of the form ‘Answer: <answer>’. For

multiple-choice questions, the answer is the option letter in parentheses, e.g. (B).” The editor prompt follows study 1 (Appendix C) and ends, for a whole rewrite, with an instruction to output every module under its header between playbook tags, and, for a one-module edit, with “Then improve the module ‘### <header>’. You may change only this module; the other modules stay as they are. Output the new content of this module, without its header, between <module> and </module> tags.” The localization step asks which single module should be changed and reads the answer from a final line “Module: <header>”; an invalid answer selects General. The editor samples at temperature 0.7 with up to 1,500 tokens (2,000 in Setting B); the agent decodes greedily with up to 512 tokens.

Setting B. The database has 48 customers, 72 products in six categories and 387 orders with statuses delivered, returned or cancelled; the API returns five records per page. The agent’s system message gives the guide text, the signatures and docstrings of the five helpers, and the keys of order and product records, and asks for one Python code block. Programs run with restricted builtins, no imports and a two-second limit. Each task family has 70 templated questions: 20 in D, 20 in V and 30 in T. The editor sees the API docstrings, the guide and the full helper code, and the failed question with the agent’s program, its output or error, and the correct answer. The planted bugs replace the helper bodies as follows: B1 returns only the first page of orders; B2 returns the subtotal instead of the total paid; B3 uses a strict comparison for the end date.

Compute. All runs used vLLM 0.11.0 on rented RTX 4090 GPUs, about 22 GPU-hours in total including setup and pilots.

G Additional Study 2 Results

Table 8. Study 2 factorial, by condition: target repair, breakage of the parent’s correct held-out answers (%), and net change on the held-out test split (points), averaged over both parents.

Feedback, scope	Qwen3-4B (A)			Phi-4-mini (A)			Tool agent (B)		
	Target	Break	Net	Target	Break	Net	Target	Break	Net
Shown, whole	0.60	6.4	+1.3	0.28	12.8	+2.1	0.15	0.3	+4.4
Same category, whole	0.47	6.0	+1.4	0.18	14.4	+1.2	0.18	0.4	+4.9
Other category, whole	0.26	5.9	+1.0	0.26	12.9	+2.2	0.06	0.3	+5.1
Shown, one module	0.69	5.9	+1.8	0.35	16.7	−0.4	0.56	4.7	+11.2
Same category, one module	0.41	5.4	+1.6	0.32	15.3	+0.6	0.54	5.2	+11.1
Other category, one module	0.31	5.5	+1.9	0.29	14.6	+2.1	0.17	5.2	+6.6

Single-step methods (Section 5.4), net change on T in points, Qwen / Phi: whole rewrite +1.2 / +2.4; with screening +1.9 / +3.7; random other module +1.0 / +1.5; category module +2.6 / +0.3; localized +3.7 / +0.5; localized with screening +3.7 / +3.2. The localization answer was valid in 69% (Qwen) and 99% (Phi) of cases and named the target’s category module in 63% and 66%. Loops (Section 5.5) kept 43% (whole rewrite) and 38% (localized) of the children they evaluated.