Robust Failure Conservative Repair for Textual Knowledge Distillation from Cross-Model Failures

Anonymous ACL submission

Abstract

Failure-based textual knowledge distillation is promising in that a model's mistakes can reveal missing task knowledge, which can potentially improve a variety of model families. We show that this signal is ambiguous. Building on a cheat sheet generation pipeline which distills training examples into cheat sheet that provides task-specific guidance to the model, we ask whether failure-derived rules can improve the generated cheat sheet. We define a rule atom to be a standalone rule entry in a cheat sheet, which may encode transferable task knowledge or model-specific patches that can confuse other models. Early experiment shows rule distillation from single model failure underperforms the base cheat sheet on non-source model families. We further see that newly added rules can be accidentally used for reasoning on unrelated cases, causing the model to incorrectly flip its answer based on the irrelevant information it receives. This motivates Robust Failure Conservative Repair (RFCR), a textual distillation procedure that derives rules from failures shared by a source model and proxy models, sharpens their application boundaries using boundary cases, and abstains when no useful rule is found. On a 400-item BIG-Bench Hard (Suzgun et al., 2023) task set , RFCR improves the baseline cheat sheets from 68.50% to 71.25% (+2.75 pp; 95% CI [+1.25,+4.50]) without performance degradation on previously correct cases. Ablations and cross-model diagnostics supports our claim that accuracy gain comes from not only new knowledge injection but also strict rule application control.

1 Introduction

Large language models (LLMs) can distill demonstrations, feedback, and task heuristics into reusable textual artifacts. A cheat sheet is one such artifact: a compact set of task rules distilled from examples and prepended at inference time. cheat sheet in-context learning (CS-ICL) has shown promise because it compresses many-shot demonstrations into a reusable summary (Honda et al., 2025). This motivation also applies to general domains. For example, the SAIR Mathematics Distillation Challenge asks whether strong mathematical reasoning can be compressed into a compact, human-readable cheat sheet that improves a weaker open-source model on a large equational-theory problem set (SAIR Foundation, 2026). This framing highlights the practical value of textual distillation: if the distilled artifact is short, readable, and model-agnostic, it can act as a portable interface between strong-model reasoning and cheaper deployment models. A natural next question is whether additional task knowledge can be distilled from the failures of a source model.

This question is a form of textual knowledge distillation. The supervision signal is not a teacher distribution or a hidden representation, but a set of failures: examples on which a source model gives the wrong answer under an anchor cheat sheet (baseline cheat sheet we improve on through different pipelines). These failures may expose missing knowledge. For example, if a source model repeatedly confuses a geometric relation, a pronoun resolution pattern, or a logical form, those failure examples may motivate having concise rules addressing the particular subtype of problems. The difficulty here is that source model failures are not automatically task knowledge. A failure-derived rule may capture a transferable improvement, but it may also encode a model-specific patch. The latter improves one model's residual errors while degrading a target model that does not share those errors. Our Robust Failure Repair(RFR) diagnostic pipeline exhibits this behavior: source-only failure refinement underperforms a static CS-ICL anchor on every non-source model family we test.

We therefore reframe failure-based refinement as *selective textual distillation*. A candidate rule atom

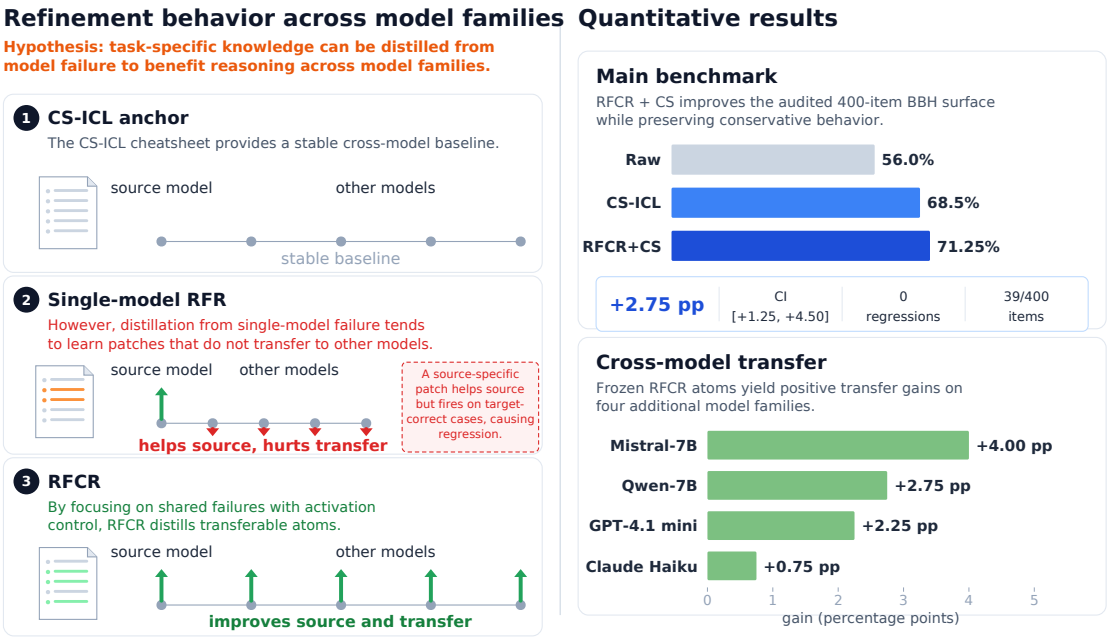

Figure 1: RFCR Pipeline and Result Overview

should not be accepted merely because it fixes a source-model failure. It should be accepted only when the model utilize it mainly on its targeted cases and when its boundary prevents its utilization on already correct examples. We define this by *activation precision*: A rule atom is activated when it is utilized by the model during reasoning for a specific task case. High activation precision means that the set of task cases that a rule atom activated on highly aligns with the target task cases it is created to solve. A low activation precision on the other hand indicates either the rule is activated on irrelevant cases or boundary cases which its activation description speaks specifically to avoid. The resulting procedure is conservative: distill a small atom when the evidence supports transfer and otherwise leave the anchor unchanged.

Contributions. First, we diagnose why source-only failure distillation can produce model-specific patches rather than transferable task knowledge. Second, we formalize transfer using activation precision, showing that source acceptance does not imply target transfer unless the atom activates mostly on failures of the evaluated model. Third, we introduce Robust Failure Conservative Repair (RFCR), a conservative textual distillation method that constructs rule atoms with proper activation control from shared failures and boundary cases and restrains from updating cheat sheet when no valid rule atom is found. On a 400-item BIG-Bench Hard (BBH) (Suzgun et al., 2023) task set, RFCR improves CS-ICL by +11 items (+2.75 pp) without performance degradations on previously correct cases. More broadly, RFCR suggests a scalable recipe for failure-based knowledge distillation: mine shared errors, refine when the resulting rules should apply, and evaluate them by whether they can help the failure cases without causing performance degradation.

2 Failure-Derived Distillation Can Produce Model-Specific Patches

We use a Robust Failure Repair (RFR) diagnostic pipeline to expose how failure-based textual distillation can fail. Phase 0 builds a global cheat sheet from examples. Phase 1 patches the cheat sheet using source-model errors. Phase 2 adds local ACTIVATE-IF/WHY rules targeting residual source failures. As distillation moves from global examples toward residual source errors, the artifact gradually becomes more model-specific (Table 1).

The early diagnostic shows why this distinction matters. On five non-ceiling BBH tasks, the full RFR pipeline falls below the static CS-ICL baseline for every non-source family (Table 2). This pattern shows that source-only refinement is not just hard to transfer, it can actively interfere with target models.

3 Transfer Theory for Failure-Derived Textual Knowledge Distillation

We define the anchor cheat sheet to be the baseline cheat sheet we aimed to improve on. An anchor-correct task case is a task case that is evaluated correctly under the anchor cheat sheet. Let C_0 be an anchor cheat sheet. For a model M, let $h_M^{C_0}(x)$ be its answer to input x under C_0. The anchor matters because the same atom may help under a raw prompt but harm on top of a stronger cheat sheet. We therefore define anchor-conditioned regions

$$F_M(C_0) = \{x : h_M^{C_0}(x) \neq y\}, \quad (1)$$

$$K_M(C_0) = \{x : h_M^{C_0}(x) = y\}. \quad (2)$$

A rule atom is a triple $m = (c, a_m, b_m)$: rule text c, a positive activation predicate $a_m(x)$, and a activation boundary predicate $b_m(x)$. Its effective activation is

$$\tilde{a}_m(x) = a_m(x)(1 - b_m(x)). \quad (3)$$

In our RFCR construction, we explicitly route rule atoms based on these predicates as part of the system. However, for globally appended natural-language rules, these predicates are often more implicit and too broad, causing occasional rule misfiring on cases outside the targeted subset.

Within the effective active region, three quantities determine the sign of the atom's effect:

$$\pi_M(m; C_0) = \Pr[x \in F_M(C_0) \mid \tilde{a}_m(x) = 1], \quad (4)$$

$$f_M(m; C_0) = \Pr[h_M^{C_0+m}(x) = y \mid \tilde{a}_m(x) = 1, \; x \in F_M(C_0)], \quad (5)$$

$$r_M(m; C_0) = \Pr[h_M^{C_0+m}(x) \neq y \mid \tilde{a}_m(x) = 1, \; x \in K_M(C_0)]. \quad (6)$$

Here $\pi_M(m; C_0)$ represents *activation precision*: among examples where the atom is used, how many are examples where the model needed help under the anchor? The terms f_M and r_M are conditional fix and performance degradation rates.

Component	Distillation role	Specificity risk
Phase 0	Build a global cheat sheet from examples	Low
Phase 1	Patch the cheat sheet using source failures	Medium
Phase 2	Add local memories for residual source failures	High
RFCR	Add routed atoms after shared-failure and activation boundary checks	Low–medium

Table 1: Failure-driven refinement can become riskier as the rule atom is being optimized for narrower source-model residuals.

Target model	CS-ICL	RFR	Δ
GPT-4.1	87.1	83.7	-3.4
Gemini-2.0	81.4	80.4	-1.0
Llama-3.3	78.3	75.7	-2.6
Claude-3.7	87.7	84.0	-3.7

Table 2: Early BBH diagnostic: mean accuracy over five non-ceiling tasks. The full source-refined RFR prompt underperforms CS-ICL on every non-source family.

Proposition 1 (Activation-precision threshold)**.** *Assume* $\Pr[\tilde{a}_m(x) = 1] > 0$ *and that* m *does not change predictions outside its effective active region. The marginal accuracy effect of* m *on model* M *under anchor* C_0 *is positive if and only if*

$$\pi_M(m; C_0) > \frac{r_M(m; C_0)}{f_M(m; C_0) + r_M(m; C_0)}, \quad (7)$$

with zero effect when $f_M(m; C_0) = r_M(m; C_0) = 0$.

Proof. Only effectively active examples can change. Conditional on activation, a fraction $\pi_M(m; C_0)$ are anchor failures and can be fixed with probability $f_M(m; C_0)$. The remaining fraction $1 - \pi_M(m; C_0)$ are anchor-correct examples and can be regressed with probability $r_M(m; C_0)$. Up to the positive factor $\Pr[\tilde{a}_m(x) = 1]$, the expected accuracy change is

$$\pi_M(m; C_0) f_M(m; C_0) - (1 - \pi_M(m; C_0)) r_M(m; C_0). \quad (8)$$

This quantity is positive exactly when the threshold above holds. □

This threshold explains why source gains do not imply transfer. A source-only gate can estimate whether m is useful for M_s, but transfer depends on whether the same atom activates precisely for a target M_t under the target anchor.

Proposition 2 (Source acceptance is not enough)**.** *For any* $f, r \in (0, 1]$*, there exist a source model* M_s*, a target model* M_t*, and a rule atom* m *such that* m *helps* M_s *and hurts* M_t*, even when both models have the same conditional fix rate* f *and performance degradation rate* r.

Proof. Choose $\pi_s > r/(f + r)$ and $\pi_t < r/(f + r)$. Construct active regions where the fraction of baseline failures is π_s for the source and π_t for the target, while both models share the same f and r. Proposition 1 gives a positive effect for M_s and a negative effect for M_t. □

Proposition 3 (Anchor-specific validity)**.** *A rule atom can have positive marginal gain under one anchor C_0 and negative marginal gain under another anchor C_1 for the same model M.*

Proof sketch. Choose anchors such that the atom's effective active region contains a high fraction of failures under C_0 and a low fraction of failures under C_1. The activation-precision threshold gives opposite signs. This formalizes why atoms selected on a raw prompt must be revalidated before being deployed on top of a stronger CS-ICL anchor. □

Proposition 4 (No-op safety)**.** *We define no-op as the deliberate choice of the pipeline to not proceed with any modification of the anchor cheat sheet. If inactive examples inherit the anchor output exactly, then a routed rule atom cannot change predictions outside its effective activation region. If no atom activates,* RFCR *returns exactly the anchor output.*

Proof. By construction, the inference procedure uses $h_M^{C_0}(x)$ whenever no selected atom has $\tilde{a}_m(x) = 1$. Thus non-activated examples are unchanged. If no atom activates, all examples follow the anchor. □

Failure overlap as a risk screen. Let F_s and F_t be source and target failure sets under C_0. We define the source-private failure region as the set of task cases where only the source model made a failed prediction and the shared failure region as the set of task cases where both source and target model give false answers. The shared failure region is $F_s \cap F_t$; the source-private failure region is $F_s \setminus F_t$. We summarize overlap with

$$\rho_{s\to t} = \frac{|F_s \cap F_t|}{|F_s|}, \quad J(s,t) = \frac{|F_s \cap F_t|}{|F_s \cup F_t|}. \quad (9)$$

Low overlap warns that source-derived rules are likely to activate imprecisely for the target. High overlap is not a guarantee, because wording, routing, fix rate, and performance degradation risk still matter.

Lemma 1 (Overlap bound)**.** *Let $\tilde{A}_m = \{x : \tilde{a}_m(x) = 1\}$ be the effective active region of atom m. If $\tilde{A}_m \subseteq F_s$, then*

$$\pi_{M_t}(m; C_0) = \Pr[x \in F_t(C_0) \mid \tilde{a}_m(x) = 1] \le \frac{|F_s \cap F_t|}{|\tilde{A}_m|}.$$

If $\tilde{A}_m = F_s$, then $\pi_{M_t}(m; C_0) = \rho_{s\to t}$.

Proof. Since $\tilde{A}_m \subseteq F_s$, we have $\tilde{A}_m \cap F_t \subseteq F_s \cap F_t$. Dividing by $|\tilde{A}_m|$ gives the bound. If $\tilde{A}_m = F_s$, the expression becomes $|F_s \cap F_t|/|F_s|$. □

4 Robust Failure Conservative Repair (RFCR)

Robust Failure Conservative Repair (RFCR) is a conservative textual knowledge distillation procedure. It treats failure-derived knowledge distillation as a routed repair procedure rather than as a global prompt update.

Let P be a proxy panel of models used during refinement, excluding the held-out target family when such a target is defined. Under C_0, the set of validation examples V are divided into

$$V_{\text{shared}} = F_s \cap \left(\bigcup_{M_j \in P} F_j \right), \quad (10)$$

$$V_{\text{private}} = F_s \cap \left(\bigcap_{M_j \in P} K_j \right), \quad (11)$$

$$V_{\text{easy}} = K_s \cap \left(\bigcap_{M_j \in P} K_j \right). \quad (12)$$

Candidate atoms are generated from V_{shared}. Source-private failure examples (V_{private}) are not positive evidence but boundary cases that tell the system when not to apply a rule. Easy examples (V_{easy}) check ordinary prompt-induced performance degradation. We leave the subset $V \setminus (V_{\text{shared}} \cup V_{\text{private}} \cup V_{\text{easy}})$ that accounts for the cases where the source model provides correct reasoning but al least one proxy model fails are put aside intentionally as source model fix rate for cases in this set remains 0 regardless of the quality of candidates generated from this subset. This makes the conservative acceptance gate defined below too strict for any rule acceptance.

rule-atom format. Each accepted rule atom is compact and human-readable:

> **RULE:** one task-level decision principle.
> **USE WHEN:** the minimal structural trigger.
> **DO NOT USE WHEN:** source-private or confounding cases.
> **CHECK:** one final verification step.

At inference time, the model receives C_0 plus the atoms whose activation condition fires and whose activation boundary condition does not fire on the particular task case. If no atom activates, the input is passed through the unchanged CS-ICL anchor. Atom activation is determined by keyword matching between the activation predicates (USE WHEN / DO NOT USE WHEN section) of the rule atom

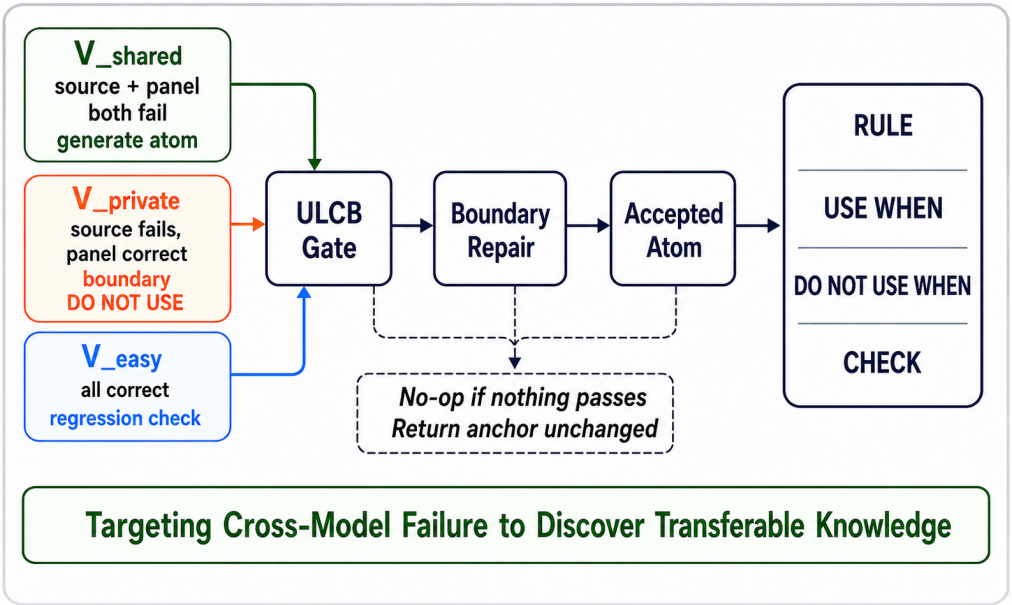

Figure 2: RFCR treats a failure-derived atom as rule text plus an activation predicate and a activation boundary predicate. If the gate fails or no route fires, the system returns the anchor output.

and task case question text. We denote this mode as routing in comparison to the global mode where all accepted atoms are shown to each task case with activation boundary check relies entirely on LLM's interpretation of the rule's activation predicates at inference time.

Conservative acceptance. Each candidate atom m is evaluated based on a conservative utility score:

$$U(m) = \min_{j \in P} \mathrm{LCB}_{95}[\Delta_j(V_s; m)] \tag{13}$$

$$- \lambda \max_{j \in P} \mathrm{UCB}_{95}[R_j(V_p; m)] \tag{14}$$

$$- \mu \max_{j \in P} \mathrm{UCB}_{95}[R_j(V_e; m)] - \nu|c|. \tag{15}$$

Here V_s, V_p, V_e abbreviate the shared, private, and easy regions. $\Delta_j(Vs; m)$ is the fix rate of m on V_s. $R_j(V_p; m)$ and $R_j(Ve; m)$ are degradation rates on V_p and V_e. We use **Wilson score interval** with z=1.96 for the confidence bounds and λ=μ=1.0 and ν=0.05 throughout experiments. The first term requires shared-failure improvement. The next two penalize performance degradations on source-private failures and easy examples. The length penalty discourages broad memories. A rule is accepted only when $U(m) > 0$. In the strict main-table deployment we also require positive net improvement and no observed performance degradations against CS-ICL-correct items.

activation boundary repair and data separation. Candidate atoms can pass conservative validation while retaining an over-broad trigger. RFCR therefore includes a boundary-repair pass over activation predicates. This pass finetunes the triggers using source-private failures and anchor-correct examples as negative evidence. It is worth noting that this procedure does not change the item universe, anchor cache, parser, scorer, or candidate rule text. It changes only whether an accepted atom is allowed to fire. In the current evidence package, the 400-item result is a controlled development result rather than a fresh held-out test: activation boundary repair used item-effect evidence from the atom-enabled pilot before the routes were frozen and rerun under the strict evaluator. We report the original locked package and the repaired package as an ablation rather than treating the repair as an unobserved held-out estimate.

No-op deployment. If no atom passes route and activation boundary checks, RFCR returns exactly the CS-ICL output. Abstention is part of our method: the system should not add a risky rule merely because a source model improved during development.

5 Experiments and Results

Evaluation Task Sets. The main evaluation uses BBH tasks for which the RFCR pipeline produced main-eligible routed atoms: disambiguation

Task Set Evaluated	n	Raw	CS-ICL	RFCR+CS	Δ	95% CI	Reg.
BBH main eligible atom tasks	400	56.00	68.50	71.25	+2.75	[+1.25,+4.50]	0
BBH full18, supplementary	1716	70.75	75.58	76.22	+0.64	[+0.29,+1.05]	0

Table 3: Strict unified-evaluator results after focused disambiguation activation boundary repair. Δ is absolute accuracy improvement in percentage points over the GPT-4.1-mini CS-ICL anchor. The main row is a controlled development result; the full18 row is supplementary because most tasks are inactive passthrough.

Model	n	CS-ICL	RFCR+CS	Fix	Reg.	Net	Δ pp / 95% CI
`anthropic/claude-haiku-4.5`	400	75.75	76.50	4	1	+3	+0.75 [-0.25, 2.00]
`google/gemini-2.5-flash-lite`	400	64.25	63.75	3	5	-2	-0.50 [-2.00, 1.00]
`mistral-7b-instruct-v0.3`	400	37.25	41.25	16	0	+16	+4.00 [2.25, 6.00]
`qwen2.5-7b-instruct`	400	51.00	53.75	13	2	+11	+2.75 [1.00, 4.75]

Table 4: Cross-model rerun with the repaired atom set. Atoms are frozen from the source protocol and applied without target-model route tuning (i.e. visible during all task case evaluations with soft activation boundary check by LLM at inference time). The evidence supports positive-net transfer on several families, but not a cross-model zero-degradation guarantee.

QA, formal fallacies, geometric shapes, and object counting. This main task set contains 400 examples. We also report a supplementary 18-task BBH task set as a sanity check on bigger scale.

Anchor and protocol. The anchor is a GPT-4.1-mini-generated CS-ICL (Honda et al., 2025) cache, scored by GPT-4.1-mini in the unified evaluator. The current artifact set does not contain a GPT-4.1-generated CS-ICL based result, so we do not claim comparison against that stronger-generator anchor. For every inactive item, the RFCR condition copies the CS-ICL output exactly. The strict rerun uses the same task set, candidate rule atoms, prompt builder, parser, retry path, scoring guard, low-concurrency call schedule, and 10,000 paired-bootstrap samples across conditions.

Models and access. The CS-ICL anchor and atom generation use `gpt-4.1-mini` via the OpenAI API directly. All other cloud-hosted models are accessed through OpenRouter: `claude-haiku-4.5`, `gemini-2.5-flash-lite`, and the early diagnostic models (GPT-4.1, Gemini-2.0-Flash, Llama-3.3-70B, Claude-3.7-Sonnet). Proprietary model parameter counts are not publicly disclosed. Open-weight models (`mistral-7b-instruct-v0.3` and `qwen2.5-7b-instruct`, $\approx$ 7B parameters each) are self-hosted via vLLM on a shared compute cluster with no per-token cloud cost.

Main controlled result. Table 3 gives the final strict source-protocol result. On the 400-item BBH task set, RFCR improves the CS-ICL anchor from 68.50% to 71.25%, a gain of +11 items or +2.75 pp, with a 95% confidence interval of [+1.25,+4.50] pp and no paired performance degradations. The supplementary 18-task check improves from 75.58% to 76.22%, also without performance degradations. It is worth noting that the 18-task run is a sanity check for inactive passthrough rather than a broad benchmark-wide improvement claim.

Cross-model diagnostics. Table 4 evaluates the repaired atom set on additional model families. The repaired atoms have positive net transfer on Claude Haiku, Mistral, and Qwen. We further observe that models with relatively weaker reasoning capability, such as Qwen and Mistral, show larger net gains with relatively low performance degradation, supporting the interpretation that RFCR distills task-specific knowledge that transfers broadly rather than merely patching source-model errors. However, Gemini Flash Lite has negative net transfer and several targets incur nonzero performance degradations, indicating that even task-specific rules may not universally apply across all model families.

Task-level decomposition. Table 5 shows that the gain is sparse. Geometric shapes contributes +7 items and disambiguation QA contributes +4. Formal fallacies is neutral; object counting is a no-

Task	CS-ICL	RFCR+CS	Act.	Fix	Reg.	Net
Disambig. QA	83	87	15	4	0	+4
Formal fallacies	67	67	4	0	0	0
Geometric shapes	56	63	20	7	0	+7
Object counting	68	68	0	0	0	0
Total	274	285	39	11	0	+11

Table 5: Task-level decomposition on the 400-item main task set. Counts are out of 100 items per task. "Act." is the number of activated routed-rule applications.

Version	n	Activated	Fix	Reg.	Net	Acc.	Δ vs CS
Original locked package	400	59	9	0	+9	70.75	+2.25
Repaired route	400	39	11	0	+11	71.25	+2.75

Disambiguation route version	Activated	Fix	Reg.	Net
Original broad route	35	1	0	+1
Repaired narrow route: `needed_one OR told_that`	15	4	0	+4

Table 6: CS-ICL-anchored same-cache route ablation. The repaired activation boundary activates fewer items but fixes more errors, supporting the activation-precision interpretation rather than a protocol-change explanation.

Condition	n	Acc.	Correct	Fix	Deg.	Net	Activated
No atom	400	56.00	224	0	0	0	0
Global atom	400	66.00	264	78	38	+40	400
Raw-Routed atom	400	66.00	264	43	3	+40	132
Boundary-routed atom	400	69.50	278	54	0	+54	132

Table 7: Raw-anchor exposure ablation on the same 400-item task set. Global source-only exposure repairs many failures but regresses 38 raw-correct items. Raw routing reduces performance degradations to 3, and activation boundary routing removes observed performance degradations.

op under the repaired CS-ICL anchor. This sparse pattern is consistent with our methodology: RFCR improves only where a routed repair is available and otherwise leaves the anchor unchanged.

Boundary-repair ablation. Because the main result uses a narrowed disambiguation route, Table 6 compares the original locked atom package with the repaired package on the same cache and item universe. The original package obtains 70.75% (+2.25 pp) without performance degradations. The repaired route obtains 71.25% (+2.75 pp) also without performance degradations, meaning that narrowing the activation space improves model performance by directing each atom only to cases where its encoded knowledge is relevant. This supports the activation-precision view: sharpening a rule's activation boundary can increase fixes without increasing performance degradations.

Global versus routed exposure. Table 7 shows the raw-anchor counterpart to the CS-ICL-anchored result. It tests what happens when the same kind of atoms are exposed without the conservative activation boundary control. It appears that global exposure leads to high coverage of the atoms but with a large negative performance degradation tail. On the other hand, we see that the raw-routing mode (LLM is presented the full atom list with cheat sheet at inference time for all task case, but task result of cases not activating any atoms falls back to the anchor CS-ICL result) recovers the degradation rate drastically. Boundary-routed mode (standard RFCR) further move the system

Variant	Acc.	Fix	Reg.	Net	Act.	frac. of Net Gain
CS anchor	68.50	0	0	0	0	–
Gen. only	70.75	14	5	+9	79	17.7
Gen.+ULCB	70.75	9	0	+9	59	15.3
+Boundary	71.25	11	0	+11	39	28.2
Oracle	72.00	14	0	+14	14	100.0

Table 8: Generated-only versus activation-boundary-repaired atoms.

toward a safer point on the coverage–safety frontier by boosting activated task case accuracy and eliminating degradation.

Generated rules versus activation boundary repair. Table 8 separates candidate generation from conservative validation. Generation-only approach yields more repairs but introduces five paired performance degradations. On the other hand, The upper/lower confidence bounds (ULCB) filtering show no performance degradations with the repaired activation boundary further improves on both net gain and activation precision. This echoes with our methodological claim that the main result comes from sparse conservative routing, not from indiscriminately adding more generated text.

6 Analysis

Sparse gains. The main gain of RFCR is sparse and routed across specific subsets of cases with similar demands for specific knowledge, rather than a global marginal improvement over the entire task. Only 39 item-rule applications activate on the 400-item task set, yet they account for +11 net fixes without any source-model performance degradations. This aligns with our Proposition 1: a valu-

Atom	Act.	Act. prec.	Fail. prec.	Fix	Net
Disambig-v1.2	15	26.7	40.0	4	+4
FF-neg.	4	0.0	50.0	0	0
Geo-1	8	12.5	12.5	1	+1
Geo-6	12	50.0	91.7	6	+6

Table 9: Activation-precision diagnostics. Higher activation precision aligns with larger net gain; the small atom count makes this a mechanism audit.

able atom needs high activation precision on the subset where it fires, not coverage of every item.

Activation precision. Table 9 reports atom-level activation precision and net gain. Activation precision here refers to number of fixed cases among all activated cases while activated failure precision here measures the fraction of CS-ICL failures among activated cases. The strongest geometric atom has both high activated failure precision and the largest net gain; the formal-fallacies atom is safe but underactive. The useful object is not a globally better prompt, but a rule atom whose activation predicate concentrates on examples the anchor is likely to fail.

Abstention is part of the method. Object counting has eligible task coverage but zero active applications in the repaired main table. This is a no-op decision, not a failure to deploy. The supplementary full-18 result likewise shows that inactive passthrough does not introduce a negative performance degradation tail, but it should not be treated as benchmark-wide improvement.

Cross-model performance degradations are informative. Cross-model performance degradations show the remaining weakness of source-derived routes. The same atom can have positive net transfer and still regress target-correct items if the target activation boundary differs from the source activation boundary. This motivates leave-one-family-out proxy gating and per-target activation boundary diagnostics.

7 Related Work

Textual knowledge distillation and cheat sheets. CS-ICL distills many-shot in-context examples into a compact textual summary, reducing inference-time prompt cost while preserving much of the benefit of long demonstrations (Honda et al., 2025). Dynamic Cheatsheet studies persistent, evolving rule for black-box language models at test time (Suzgun et al., 2026). Our work is complementary: rather than asking only whether a distilled rule is useful in aggregate, we ask when failure-derived textual knowledge transfers across model families without becoming a model-specific patch.

Prompt optimization and reflective search. Automatic prompt optimization methods rewrite prompts using data, feedback, search, or textual gradients (Pryzant et al., 2023; Fernando et al., 2024; Yang et al., 2023; Khattab et al., 2024; Yuksekgonul et al., 2024; Agrawal et al., 2025). These methods usually optimize a prompt for a measured development distribution. RFCR focuses on a narrower distillation risk: a failure-derived atom can be accepted by the source model yet hurt another model if it activates on examples that the target already solves.

Reasoning benchmarks. We evaluate on BIG-Bench Hard(BBH), a challenging subset of BIG-Bench designed to stress reasoning behavior (Srivastava et al., 2022; Suzgun et al., 2023). Because several BBH tasks are sensitive to prompt wording and reasoning format, they are useful for testing whether distilled textual repairs are robust rather than source-specific. We discuss AGIEval as a diagnostic benchmark, but do not use it as a main claim until a locked nonnegative-anchor protocol is available (Zhong et al., 2024).

8 Conclusion

Failure-based textual distillation can produce either transferable task support or model-specific patches. We formalize this distinction through activation precision and show why source acceptance does not imply target transfer. RFCR operationalizes the analysis by using shared failures for rule atom construction, source-private failures and anchor-correct examples for boundaries, and conservative checks for deployment. On a 400-item BBH task set, RFCR achieves +2.75 pp over CS-ICL without any performance degradations. The RFCR success argues that textual knowledge distillation under the context of cheat sheet improvement should take into consideration both mining new knowledge and find the best activation boundary for the acquired new knowledge. Future work should also focus on experimenting with different form of cheat sheet modification beyond simple rule appending to isolate the effect of cheat sheet structuring from actual knowledge-induced performance variance.

Ethical Considerations

This work studies textual distillation on reasoning benchmarks. The main risk is overclaiming reliability: a rule artifact that improves one model may harm another. We therefore report paired performance degradations, preserve a no-op fallback, and emphasize that source-model gains alone are not evidence of general improvement. The experiments do not require personal data.

Limitations

The main improvement is modest and concentrated in geometric shapes and disambiguation QA. The evidence supports conservative routed textual distillation as a sparse no-degradation improvement over a strong same-generator CS-ICL anchor instead of a broad benchmark optimization claim.

The zero-degradation property holds only for the locked source-model protocol. Cross-model performance degradations remain nonzero for some targets, reflecting that activation predicates were derived from source-model failures and not validated against every target-model activation boundary. The current paper includes a retrospective leave-one-family-out proxy-gating diagnostic, but it does not yet regenerate atoms with each family fully excluded. A stronger test should perform end-to-end leave-one-family-out generation, gating, freezing, and target-only evaluation.

The main evaluation task set contains four BBH tasks with eligible routed atoms. Broader benchmark-scale claims require more tasks, learned routers, and stronger held-out validation. The supplementary full-18 result is useful as a sanity check, but it should not be interpreted as evidence that every task benefits from RFCR, because most items use inactive passthrough.

The main 400-item result is a controlled development result, not a fresh held-out test after activation boundary repair. The final disambiguation activation boundary was repaired using item-effect evidence from the atom-enabled pilot before the repaired route was frozen and rerun under a strict evaluator. A fresh held-out confirmation would strengthen the empirical claim.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*.

Ukyo Honda, Soichiro Murakami, and Peinan Zhang. 2025. Distilling many-shot in-context learning into a cheat sheet. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17158–17178.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri R. Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.

SAIR Foundation. 2026. Mathematics distillation challenge: Equational theories. https://competition.sair.foundation/competitions/. Accessed 24 May 2026.

Aarohi Srivastava and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2026. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, pages 7080–7106.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic differentiation via text. *arXiv preprint arXiv:2406.07496*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314.

A Reproducibility Statement

The final run uses a fixed task set, fixed CS-ICL cache, fixed selected-atom file, the same parser and prompt builder across conditions, a scoring guard, low-concurrency calls, inactive passthrough for non-activated items, and a paired bootstrap with 10,000 samples. The reported CS-ICL column is the unified-evaluator GPT-4.1-mini CS-ICL anchor/cache condition and should not be mixed with separately reported deployment numbers from earlier runs. The main 400-item result and all ablation tables are from a single locked run. The exploratory holistic rewrite table (Appendix E) reports means over three seeds (1000, 2000, 3000). Cross-model 95% CIs are computed per model using the same paired-bootstrap procedure.

B RFCR Implementation Details

Shared-failure identification. A failure is classified as *shared* if at least one proxy panel model $M_j \in P$ also fails the item under C_0. This is a union criterion: $V_{\text{shared}} = F_s \cap (\bigcup_j F_j)$. A failure is *source-private* if *every* panel model gets it right: $V_{\text{private}} = F_s \cap (\bigcap_j K_j)$. Items that all models including the source get right form V_{easy}. Cluster size within V_{shared} serves as a soft priority signal for which failure patterns to write atoms for first.

Atom generation prompt. Atoms are generated by presenting the source model with a curated batch of shared-failure examples together with a selection of source-private boundary examples, then issuing the following template:

```
You are given examples where a
language model failed despite
having a general cheat sheet.
Identify the common structural
pattern and write a repair rule.
Failed examples (model got
wrong):
[SHARED_FAILURE_EXAMPLES]
Boundary examples (model gets
right  the rule must NOT fire
here):
[SOURCE_PRIVATE_EXAMPLES]
Write the rule in this exact
format:
RULE: <one task-level decision
principle>
USE WHEN: <minimal structural
trigger>
DO NOT USE WHEN: <conditions
from the boundary examples>
CHECK: <one final verification
step>
Be as specific as possible.  Do
not write a rule that applies to
all task examples.
```

The **DO NOT USE WHEN** field is seeded from V_{private} so the model is nudged toward writing a narrow activation boundary condition rather than a broad topic match. Multiple candidates are drawn per cluster using temperature sampling; all candidates are then scored by the ULCB gate.

Upper/Lower confidence bounds. Confidence bounds use a normal-approximation 95% interval: $\text{LCB}_{95}[\Delta_j] = \hat{\Delta}_j - 1.96\,\hat{\sigma}_j$ and $\text{UCB}_{95}[R_j] = \hat{R}_j + 1.96\,\hat{\sigma}_j$, where $\hat{\Delta}_j$ and $\hat{R}_j$ are the observed improvement and performance degradation rates on the respective validation subsets and $\hat{\sigma}$ is the standard error of the proportion. An atom passes the gate only when $U(m) > 0$. The strict deployment condition additionally requires zero *observed* performance degradations against CS-ICL-correct items across all panel models (not just a UCB bound).

Activation Boundary repair prompt. After an atom passes the ULCB gate, its activation predicate is inspected for over-triggering. The repair prompt presents the source model with over-triggered cases and asks for a narrower condition:

```
An accepted rule fires on the
following examples but should
not (anchor-correct or source-
private cases):
[OVER_TRIGGERED_EXAMPLES]
The rule correctly fires on
these examples:
[CORRECTLY_TRIGGERED_EXAMPLES]
Current USE WHEN:
[CURRENT_TRIGGER]
Write a narrower USE WHEN
condition that still covers
the correct examples and
excludes the over-triggered
ones.  Express it as a minimal
keyword or structural pattern
(e.g., "contains X OR contains
Y").
Narrowed USE WHEN:
```

Each candidate narrowing is evaluated on the locked task set. The narrowing that maximizes net gain while maintaining zero observed performance degradations is selected.

Scoring guard and retry logic. Each model call goes through a scoring guard that checks whether the returned answer can be parsed to a valid option or numeric value. Failed parses trigger up to two retries with a slightly relaxed extraction prompt before the item is marked as a parse failure. Items with parse failures are excluded from paired comparisons but logged. The low-concurrency call schedule (sequential rather than batch-parallel) was adopted to reduce inter-call variance across conditions; all conditions for a given item are scored in the same session where possible.

Paired bootstrap. The 95% CI and performance degradation counts are computed over 10,000 paired bootstrap resamples of the item-level effect vector $e_i = \mathbf{1}[\text{RFCR correct}] - \mathbf{1}[\text{CS-ICL correct}]$. A performance degradation is defined as $e_i = -1$; a fix as $e_i = +1$; a no-op as $e_i = 0$. The reported CI is the percentile interval of the bootstrap distribution of $\sum_i e_i/n$.

C RFR Diagnostic Pipeline

The RFR pipeline is the source-only refinement baseline we used to produce the diagnostic results in Table 2. It runs in three phases on a source model over a training split, with each phase accepting patches only when they pass source-model quality gates with no multi-model evaluation at any stage. This is the fundamental reason its output becomes model-specific. Below we describe each phase in detail.

Phase 0: Bootstrap cheat sheet. The pipeline samples the first 75 training examples (configurable). For examples lacking an explicit chain-of-thought, it first generates zero-shot reasoning traces via the source model. The examples are then formatted as Question / Reasoning / Answer triplets and passed to the following prompt:

```
Create a cheat sheet based on
the examples below.  You will
be asked to answer questions
similar to these examples during
the test, without being allowed
to refer to the examples at
that time.  Your task here is
to make a cheat sheet that will
help you answer such problems
correctly.  First, carefully
read the examples below and
identify which ones you find
most difficult to answer.
[EXAMPLES]
Now, create a cheat sheet to
help you solve the difficult
examples.  Exclude any
content that is easy for you,
and only include specific,
detailed points to address the
challenging ones.
```

The output is a `prior_knowledge` text block that forms the starting anchor for Phase 1.

Phase 1: Iterative prior-knowledge patching. Phase 1 runs an iterative repair loop over source-model failures under the current `prior_knowledge`. Each iteration (up to a configurable maximum, default 4) samples up to 15 failures and constructs a *failure block* for each: the input, the expected and predicted answers, the source model's wrong reasoning trace, and when oracle traces are available the correct reasoning from a stronger reference model. The patch prompt is:

```
You are refining a knowledge
guide that helps a model answer
questions correctly.
=== CURRENT KNOWLEDGE GUIDE ===
[PRIOR_KNOWLEDGE]
=== END KNOWLEDGE GUIDE ===
The model is making the
following errors:
[FAILURE_BLOCKS]
Produce an IMPROVED version of
the knowledge guide that helps
avoid these mistakes.  You may:
- ADD a new rule, clarification,
or concrete example
- MODIFY an existing rule to be
more precise or correct
- REMOVE a rule that is actively
causing errors
Requirements:  preserve rules
that are working correctly;
focus on ABSTRACT REASONING
PRINCIPLES, not memorization of
specific examples; output ONLY
the improved knowledge guide
text.
```

A candidate patch is committed only when it passes two source-model gates: (a) it fixes at least 20% of the sampled failures (*fix-rate gate*), and (b) it does not regress more than 20% of a sample of currently-correct items (*performance degradation gate*). The loop exits when source accuracy reaches 85% or when two consecutive iterations produce no accepted patch. The oracle contrast showing the correct reasoning from a stronger model alongside the wrong reasoning provides a concrete target for the patcher, but is itself derived from source-model failures and carries no cross-model validity signal.

Phase 2: Case study generation. Phase 2 operates on residual failures after Phase 1. It partitions failures by structural feature signature (e.g., argument form, entity type, geometric relation) and, for each partition with at least 3 failures, generates up to 3 candidate case studies using a task-specific structured prompt. Each generated case study has the following fields:

```
=== CASE STUDY: [title] ===
FAILURE_TYPE: A (missing
knowledge) | B (wrong pattern)
ACTIVATE IF:
- [structural condition 1]
- [structural condition 2]
DO NOT ACTIVATE IF: [boundary
condition]
COMMON WRONG MOVE: [what the
model typically does wrong]
NEXT CHECK: [mechanical
verification step]
WHY THIS WORKS: [1-2 sentences
explaining the fix]
SUPPORT:
 E1 = ...  | E2 = ...  | Answer:
...   [note]
```

The ACTIVATE-IF conditions become the activation predicate; DO NOT ACTIVATE IF becomes the boundary predicate. A candidate case study is accepted when it (a) fixes at least 30% of the partition failures, (b) does not regress more than 15% of correct items, and (c) is not a duplicate or near-duplicate of an existing case study through an LLM judge. Accepted case studies are appended to the cheat sheet and re-scored; partitions are refreshed after each flush, and the loop runs for up to 5 iterations.

Why model-specificity accumulates. All three gates in Phases 1 and 2 are evaluated against the *source model only*. A Phase 1 patch is accepted if it fixes source failures and does not break source-correct items; a Phase 2 case study is accepted if its partition fix rate and performance degradation rate are satisfactory on the source. Neither the fix-rate threshold nor the degradation threshold is measured against any other model other than the source model at any point during development. As a result, the accepted patches and case studies are precisely those that explain the source model's residual error profile, which may not align with the shared error profile across models. Through this mechanism, the full RFR artifact accumulates model-specific content as it progresses through Phase 1 and Phase 2, matching the specificity gradient observed in Table 1.

D Supplementary Audits

Protocol. The repaired rerun used the same item universe, CS-ICL cache, parser, prompt builder, retry path, scoring guard, and inactive passthrough rule as the locked protocol. It activated 39 RFCR+CS-ICL item-rule applications and had zero missing effect rows, baseline mismatches, parse errors, or failed calls (Table 10).

Audit field	Value
Main eligible items	400
Full18 items	1716
Activated applications	39
Missing effect rows	0
Baseline mismatches	0
Parse errors	0
Failed calls	0
Bootstrap samples	10,000

Table 10: Protocol audit for the repaired rerun.

Task eligibility. Table 11 audits the full18 task set. Only main-paper-eligible tasks, plus object-counting no-op eligibility, define the main task set; Web-of-Lies is exploratory.

AGIEval anchor diagnostic and rebuild check. AGIEval is excluded from the main claim because the available CS-ICL anchor underperforms raw prompting on all three tasks (Table 12, top). A CS-ICL anchor that is worse than raw prompting is not a valid starting point for a distillation improvement claim: any RFCR gain measured against it is at high risk of being confounded by the anchor deficit rather than attributable to the atoms. We therefore ran a development-only anchor rebuild check (E8) to determine whether a nonnegative anchor could be recovered for any task (Table 12, bottom). Two tasks (LogiQA and LSAT-LR) produced nonnegative development anchors, but LSAT-AR found no variant that satisfied the nonnegative-anchor requirement. As a result, AGIEval remains diagnostic and is not promoted to the main benchmark task set.

Leave-one-family-out proxy diagnostic. Table 13 evaluates whether the repaired atom set remains useful when a target model family is treated as held out during proxy gating. This is a retrospective diagnostic: the atom set is fixed from the source protocol and is not regenerated with each family excluded. It should therefore not be read as a full end-to-end leave-one-family-out experiment, but rather as a check on whether the accepted atoms are source-specific to the extent of actively

Task	Status	n	Task	Status	n
boolean	no atom	100	causal judgement	no atom	87
date understanding	no atom	100	disambig. QA	main	100
formal fallacies	main	100	geometric shapes	main	100
LD five	no atom	100	LD seven	no atom	100
LD three	no atom	100	navigate	no atom	100
object counting	main no-op	100	penguins table	no atom	58
colored objects	no atom	100	snarks	no atom	71
sports	no atom	100	temporal seq.	no atom	100
tracking shuffled	no atom	100	web of lies	exploratory	100

Table 11: Full18 eligibility audit. The main result uses only tasks with main-paper-eligible atoms or no-op eligibility; Web-of-Lies remains exploratory.

Task	n	Raw	CS-ICL	Net
LogiQA	300	57.67	57.36	-1
LSAT-AR	23	40.00	30.43	-11
LSAT-LR	373	81.57	79.22	-6
Aggregate	696	63.51	60.92	-18

Task	Raw dev	Anchor dev	Fix	Reg.
LogiQA	50.00	70.00	4	0
LSAT-AR	–	–	–	–
LSAT-LR	85.00	90.00	1	0

Table 12: Top: AGIEval rejected-anchor diagnostic. Bottom: development-only nonnegative-anchor rebuild check (E8).

hurting other families even under a favorable gate. The results show positive net transfer for Anthropic, Mistral, and Qwen, while Gemini incurs a net performance degradation. The performance degradation for Gemini is consistent with the cross-model result in Table 4 and reflects that the atom activation boundaries were derived from source-model failures without Gemini-specific validation.

Held-out family	CS-ICL	RFCR+CS	Fix	Reg.	Net
Anthropic	75.75	76.25	3	1	$+2$
Google/Gemini	64.25	63.75	3	5	-2
Mistral	37.25	40.75	14	0	$+14$
Qwen	51.00	53.75	13	2	$+11$

Table 13: Retrospective leave-one-family-out proxy-gating diagnostic. The atom set is frozen from the source protocol; this is not a full end-to-end LOO experiment. Positive net transfer holds for three of four families, with Gemini as the exception.

Per-item mechanism examples. Table 14 gives compact examples of fixes, no-op passthrough, and boundary blocking. These examples are illustrative and are not used as additional evaluation evidence.

E Exploratory Holistic Rewrite Results

Iterative holistic rewrite is an exploratory alternative to atom appending. Rather than appending a

Task	ID	CS	RFCR	Gold	Atom	Blocked	Outcome
geo	0155	B	G	G	geo-6	no	fix
geo	0203	A	K	K	geo-1	no	fix
disambig	0171	C	B	B	disambig	no	fix
disambig	0242	C	A	A	disambig	no	fix
obj. count	0150	9	9	9	none	n/a	no-op
disambig	0154	A	A	A	disambig	yes	boundary no-op

Table 14: Qualitative per-item mechanism examples. The no-op rows illustrate inactive passthrough and boundary blocking rather than global prompt expansion.

	Causal judgment			Object counting		
Target model	App.	Hol.	Δ	App.	Hol.	Δ
`gpt-4.1`	**62.8**	61.3	-1.5	**83.7**	68.0	-15.7
`llama-3.3-70b`	**66.3**	64.4	-1.9	60.3	**62.0**	$+1.7$
`gemini-2.5-fl`	61.3	**65.9**	$+4.6$	**99.0**	97.0	-2.0
`claude-3.5-hk`	64.0	**65.1**	$+1.1$	89.7	**92.0**	$+2.3$
`mistral-7b`	**61.3**	54.8	-6.5	35.3	**47.0**	$+11.7$
`qwen2.5-7b`	57.9	57.9	0.0	36.3	**60.3**	$+24.0$

Table 15: Exploratory cross-model transfer accuracy (%, mean over seeds 1000/2000/3000) for append vs. holistic rewrite on causal judgment and object counting. Bold marks the better mode per cell; Δ = holistic $-$ append (pp).

routed atom alongside the CS-ICL anchor, holistic rewrite integrates the repair directly into the cheat sheet text, restructuring or merging rules where needed. It may help when a repair requires structural integration rather than a localized addition, but it is not yet robust enough to be a main contribution. Table 15 reports held-out accuracy for append versus holistic rewrite on causal judgment (CJ) and object counting (OC) across six target models (mean over seeds 1000/2000/3000).

On OC, holistic rewrite yields large gains for the two small open-weight models (`mistral-7b`: $+11.7$ pp, `qwen2.5-7b`: $+24.0$ pp) and modest gains for `claude-3.5-haiku` ($+2.3$ pp) and `llama-3.3-70b` ($+1.7$ pp), but substantially hurts `gpt-4.1` (-15.7 pp). On CJ, the picture is more mixed: holistic rewrite helps Gemini and Claude but hurts Mistral and is neutral on Qwen. The pattern suggests that holistic rewrite benefits

models whose reasoning is less sensitive to prompt structure, particularly smaller open-weight models that may rely more on the integrated rule text. On the other hand, stronger proprietary models are harmed when the cheat sheet structure is disrupted. These results are promising for structured-integration use cases but should be treated as exploratory until replicated with tighter controls.