# ARC: LEARNING SELECTIVE ANSWER REPAIR FROM MATCHED OUTCOMES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Small language models offer efficient inference, but errors in their initial answers can limit their reliability on reasoning tasks. Self-refinement and external tools can help correct these errors, although applying repairs to every answer adds computation and can damage correct answers. The challenge is to predict which repair, if any, will improve an answer before executing it. We introduce ARC (Anchor-conditioned Repair Controller), which learns this decision from matched repair outcomes. During training, we apply all available repairs to the same initial output, called the Anchor, and use their final-answer scores to supervise retention and repair selection. At deployment, an output-conditioned hierarchical router reads the question, generated answer, and public reasoning, then retains the Anchor or executes one repair with the base model fixed. On Qwen3-8B across five reasoning task families, the main evaluation yields a mean task score of 0.6915, a 6.75% improvement over retaining initial answers. It uses 0.155 additional LLM calls and 0.348 s of recorded LLM request time per example, reducing these costs by 84.5% and 90.9%, respectively, compared with one self-refinement call per answer. A separate deterministic evaluation over three seeds yields nearly equal mean in-distribution scores with 24.8% fewer additional calls than routing without the generated answer or reasoning. Experiments with the other evaluated base models also show improvements over answer retention and question-only routing. Our code is available at `https://anonymous.4open.science/r/ARC-474B/`.

## 1 INTRODUCTION

Small language models offer efficient inference, but their initial answers are not always reliable. Self-refinement can revise these answers using model-generated feedback (Madaan et al., 2023), while external tools provide task-specific computation or evidence. The benefit depends on where the computation is spent: repairing every answer adds model calls and can turn a correct answer into an incorrect one. Selective repair aims to improve answer quality while preserving the efficiency that motivates using a small model.

Prior work allocates computation by selecting models from the question, verifying generated answers, or directing difficult problems to refinement (Ong et al., 2025; Aggarwal et al., 2024; Chen et al., 2025). For a fixed base model with several repair procedures, the decision also depends on what each procedure will do to the completed answer. In our temporal-reasoning examples, the same repair corrects an incomplete answer, preserves a correct answer, or corrupts one (Section 3). Knowing the task family or whether the current answer is correct does not determine the effect of a repair. This motivates supervision based on the final-answer value of each available action, including retention.

We introduce ARC (Anchor-conditioned Repair Controller) to learn this decision from matched repair outcomes. For each labeled training example, we apply every available repair to the same initial output, called the Anchor, and compare the resulting final-answer scores with retention. These comparisons supervise both the preferred action and the gains of alternative repairs. An output-conditioned hierarchical router uses the question, generated answer, and public reasoning to score retention and repairs in a shared action space. At deployment, it retains the Anchor through NoSkill or executes one repair, without observing candidate repair outcomes in advance.



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

In the main evaluation across four base models and five reasoning task families, ARC improves mean task score by 4.4–7.2 points over Retention while repairing 10.7–19.6% of ID examples. With deterministic Qwen routers, a development-calibrated sweep identifies three settings with higher mean test scores and fewer calls than question-only routing. At the default threshold, the two routers obtain nearly equal mean ID scores, with 24.8% fewer additional calls for ARC.

Our contributions are:

- We formulate post-generation answer repair as selective routing over retention and a heterogeneous library of executable repair procedures.
- We introduce ARC, which evaluates all available repairs on the same Anchor offline and uses their final-answer scores to train an output-conditioned hierarchical router. At deployment, it retains the Anchor or executes one repair with a fixed base model.
- In the main evaluation across four 4B–8B models and five task families, we show higher mean task scores than retention and question-only routing at low intervention rates, with further quality-cost gains at development-calibrated Qwen operating points.

## 2 Related Work

Answer refinement studies how further inference can improve a generated response. Self-Refine uses feedback from the same model to revise its output (Madaan et al., 2023), while CRITIC obtains feedback from external tools to validate and correct an initial response (Gou et al., 2024). SCoRe trains self-correction through multi-turn RL, with reward shaping that favors improvements between attempts (Kumar et al., 2025). These approaches improve the revision process through feedback or model training. MAgICoRe also addresses over-correction by directing harder problems to fine-grained refinement (Chen et al., 2025). Our setting adds a choice among heterogeneous repair procedures, with retention included as an action. ARC keeps the base model fixed and trains a controller on the final-answer outcomes of alternative repairs applied to the same Anchor. This supervision connects the choice of a repair to its observed benefit over retaining the answer.

Model routing addresses inference cost by allocating queries across models. RouteLLM learns to choose between stronger and weaker models from preference data (Ong et al., 2025); AutoMix uses self-verification of a generated answer to guide escalation to a larger model (Aggarwal et al., 2024). BEST-Route jointly selects a model and the number of responses to sample, accounting for query difficulty and a target quality level (Ding et al., 2025). These methods allocate computation through model choice, escalation, or repeated sampling. ARC makes the decision after an answer has been generated: the controller reads that output and chooses among retention, self-refinement, and tool-assisted repairs. It estimates their relative value before executing a repair, so candidate repaired answers are needed for offline supervision but not for deployment-time selection.

Protocol routing chooses a collaboration strategy. Yang et al. (2026) compare four collaboration protocols with a fixed solver and find that failure risk can be easier to predict than the value of a particular protocol. This distinction also matters for answer repair: knowing that an answer is likely to be wrong does not establish which repair will improve it. ARC compares retention with heterogeneous repairs of the same realized Anchor and learns from their final-answer scores. Matching these outcomes holds the initial output fixed across actions and provides a common reference for learning repair preferences and gains. At deployment, the policy retains the answer or executes one selected repair. These are decisions over executable procedures; sparse mixture-of-experts (MoE) layers instead select expert parameters within a network (Shazeer et al., 2017; Fedus et al., 2022).

## 3 Motivation

A tool can be relevant to a task without improving the generated answer. Three CoTempQA training examples make this distinction concrete. The same Temporal action adds a missing employer to an incomplete answer ($0 \to 1$), preserves a correct employer answer ($1 \to 1$), and replaces a correct party-membership answer with an empty array ($1 \to 0$). Their gains are $+1$, $0$, and $-1$; Appendix B.4 gives the questions and recorded outputs. Across all 800 temporal training examples, this action improves 352 answers, leaves 438 scores unchanged, and harms 10 answers (Appendix B.5). Task relevance and current answer correctness alone therefore do not determine the outcome of a repair.

These observations motivate learning from alternative repairs applied to the same initial output. On labeled training examples, we collect these outcomes offline, score each final answer, and measure its gain over retention. Each example supplies a best-action label and gains for the available alternatives. The router learns from these targets using the generated answer and public reasoning available at decision time. At deployment, it retains the initial answer or selects a repair before executing any candidate.

## 4 Problem Setup

We select a repair after a base model generates its initial answer. For example $i$, let $x_i$ denote the input, $d_i$ the dataset, and $y_i^\star$ the gold answer. The fixed base model produces the Anchor $(a_{0i}, r_{0i}) = M(x_i)$, consisting of the initial answer and public reasoning text. The router reads $o_i = (x_i, a_{0i}, r_{0i})$ and chooses from eight actions: NoSkill, SelfRefine, and six specialized repairs for logic, time, mathematics using programs or symbolic solvers, spatial relations, and evidence retrieval. NoSkill retains $a_{0i}$ without an additional call. Availability is determined from the public question and context before execution, without reading the Anchor. We record it with a binary mask $b_{is} \in \{0, 1\}$ and choose from $\mathcal{F}_i = \{s \in \mathcal{A} : b_{is} = 1\}$. Each executed action returns a final answer $y_{is}$ and a status $q_{is} \in \{\texttt{ok}, \texttt{abstain}, \texttt{error}\}$. Appendix B.1 gives the availability and execution rules.

The dataset scorer $U_{d_i}(\widehat{y}, y_i^\star) \in [0, 1]$ measures answer quality. We define the action reward $R_{is}$ as the final answer score and the gain $G_{is}$ as the difference from the Anchor score:

$$R_{is} = U_{d_i}(y_{is}, y_i^\star), \qquad G_{is} = R_{is} - R_{i0}. \tag{1}$$

For NoSkill, $y_{i,\text{NoSkill}} = a_{0i}$ and $G_{i,\text{NoSkill}} = 0$. If an action abstains or fails, it returns the Anchor, and we include the cost of the attempt. The goal is to choose the action with the highest expected reward given the observed input, generated output, and available repairs:

$$s^\star(o, \mathcal{F}) = \arg\max_{s \in \mathcal{F}} \mathbb{E}[R_s \mid o, \mathcal{F}]. \tag{2}$$

Reference answers are used offline; deployment uses only $o$ and $\mathcal{F}$. We evaluate both answer quality and additional repair calls, using a retention threshold to control intervention.

## 5 Method

Figure 1 shows the three stages of ARC: offline outcome collection, router training, and selective repair at deployment. The base model remains fixed throughout.

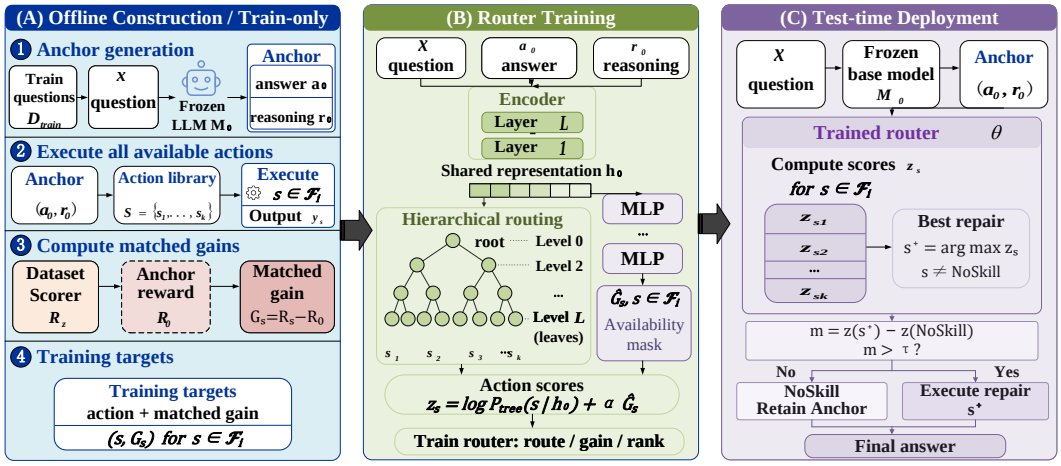



Figure 1: Offline repair evaluation and selective deployment in ARC. All available repairs start from the same Anchor during data collection. Their outcomes train a hierarchical router that, at inference time, retains the Anchor or selects one repair before observing its outcome.

### 5.1 Matched Repair Outcomes as Supervision

We supervise repair selection using final-answer scores to compare each action with retention. For each training Anchor, we apply every available repair to the same initial output. A row of the matched action matrix records the resulting answers, rewards, gains, statuses, and costs, providing targets even for repairs that the learned policy would not select. SelfRefine asks the base model to critique and revise the Anchor in one LLM call. The specialized backends perform logical inference (Datalog), temporal reasoning (Temporal), program execution (PythonMath), equation solving (SymPy), spatial reasoning (SpatialGraph), or evidence retrieval (EvidenceRetrieval). For a specialized action $s$, we obtain assistance $e_{is}$ and a revised answer:

$$e_{is} = B_s(o_i), \qquad y_{is} = C_M(o_i, e_{is}), \tag{3}$$

where $B_s$ is the backend and $C_M$ prompts the fixed base model with the original task, Anchor, assistance, and answer contract. The score of $y_{is}$ measures the effect of the complete repair, including both corrections and errors introduced during revision. Appendices B and C specify the backends and prompts.

The matrix has one column per action and records availability alongside execution outcomes. Each row supplies a best-action label:

$$\widetilde{s}_i = \mathrm{TB}\left(\arg\max_{s \in \mathcal{F}_i} R_{is}\right), \tag{4}$$

where TB breaks ties by preferring NoSkill, then fewer model calls, lower latency, and finally a fixed action order. Retention is therefore learned within the same action space and is the target whenever no repair improves the answer. Subtracting the Anchor score gives every action a common zero reference. This leaves the best-action set and all pairwise reward differences unchanged:

$$\arg\max_{s \in \mathcal{F}_i} G_{is} = \arg\max_{s \in \mathcal{F}_i} R_{is}, \qquad G_{is} - G_{it} = R_{is} - R_{it}. \tag{5}$$

### 5.2 Hierarchical Output-Conditioned Router

The router input contains the question, choices, a fixed context view, public reasoning, and the answer. The concatenated text is right-truncated to 512 tokens and encoded by DeBERTa-v3-small (He et al., 2021). We combine its `[CLS]` state with ten length and count features, then project the result to $h \in \mathbb{R}^{512}$. Dataset identity, family identity, the gold answer, answer type, and the availability mask are excluded from encoder features. Question-Only omits the answer and reasoning; Answer-Only omits reasoning. Appendix D specifies the features and reports field retention after truncation.

The router combines probabilities over a functional hierarchy with a gain prediction for each repair. Repairs with related functions share routing gates, as with program execution and symbolic equation solving. The root gate chooses among retention, self-refinement, structured reasoning, and evidence retrieval. The structured branch distinguishes logic, time, mathematics, and spatial actions, while the mathematics branch distinguishes PythonMath and SymPy. All gates and gain predictors use the same representation $h$. Let $\mathcal{V}(s)$ contain the gates on the path to action $s$, and let $k(v, s)$ denote the child selected at gate $v$. With gate logits $g_v(h)$, the path probability is

$$p_s^{\mathrm{path}}(h) = \prod_{v \in \mathcal{V}(s)} \left[\mathrm{softmax}(g_v(h))\right]_{k(v,s)}. \tag{6}$$

For Temporal, the path probability is the product of the root probability for structured reasoning and the conditional probability for the temporal branch. A separate head predicts $\widehat{G}_s$ for each repair. We combine both outputs with an optional cost term $c_s$:

$$z_s = \log p_s^{\mathrm{path}} + \alpha \widehat{G}_s - \lambda c_s, \qquad P_\theta(s \mid o) = \mathrm{softmax}_{s \in \mathcal{F}}(z_s). \tag{7}$$

The path term expresses a routing preference shared within repair groups; the gain term adds an action-specific estimate of improvement. NoSkill has its own root branch and $\widehat{G}_{\mathrm{NoSkill}} = 0$. We use $\alpha = 1$ and $\lambda = 0$ throughout; cost affects training labels only when rewards tie. Unavailable actions receive a score of $-\infty$.

### 5.3 Training Objective

Each matched row supplies a best-action target. We also supervise the gains and pairwise ordering of alternative actions, which a single action label omits. A group-level auxiliary task predicts whether each of six repair groups contains a positive-gain action. The resulting objective is:

$$\mathcal{L} = \mathcal{L}_{\text{route}} + \beta\mathcal{L}_{\text{gain}} + \gamma\mathcal{L}_{\text{rank}} + \eta\mathcal{L}_{\text{aux}}, \qquad (\beta, \gamma, \eta) = (1, 0.5, 0.2), \tag{8}$$

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|} \sum_{(i,s,t)\in\mathcal{P}} \text{softplus}\Big(-(G_{is} - G_{it})(\widehat{G}_{is} - \widehat{G}_{it})\Big), \tag{9}$$

where $\mathcal{P}$ contains pairs of actions evaluated on the same example, pooled over a minibatch. The ranking term favors predicted differences with the same sign as the measured differences. $\mathcal{L}_{\text{route}}$ is class-weighted cross-entropy on $\widetilde{s}_i$. Gain regression uses the set $\mathcal{B}$ of available non-NoSkill actions in the minibatch:

$$\mathcal{L}_{\text{gain}} = \frac{1}{|\mathcal{B}|} \sum_{(i,s)\in\mathcal{B}} \text{Huber}_1(\widehat{G}_{is} - G_{is}). \tag{10}$$

$\mathcal{L}_{\text{aux}}$ is binary cross-entropy on the six group labels. Appendix D specifies class weights, pair sampling, and auxiliary targets. The Gain/Rank ablation removes the two gain-based losses jointly.

### 5.4 Thresholded Repair Selection

Given the generated $(a_0, r_0)$, the router identifies the best available repair $s^+$ and computes its score margin over retention:

$$s^+ = \arg\max_{s\in\mathcal{F}\setminus\{\text{NoSkill}\}} z_s, \qquad m = z_{s^+} - z_{\text{NoSkill}}, \qquad \widehat{s} = \begin{cases} s^+, & m > \tau, \\ \text{NoSkill}, & m \le \tau. \end{cases} \tag{11}$$

The repair scores determine which action is preferred, while the margin against NoSkill determines whether to execute it. Increasing $\tau$ reduces the set of repaired examples without retraining the router or changing its ordering over repair actions. The original policy uses $\tau = 0$; the budget sweep calibrates $\tau$ using development call counts (Appendix H). Cached evaluation applies this rule before retrieving the selected outcome; failed actions follow the rules in Section 4.

## 6 Experimental Setup

### 6.1 Models and Training

We evaluate Qwen3-8B (Yang et al., 2025), Llama-3.1-8B (Grattafiori et al., 2024), Gemma-3-4B (Gemma Team et al., 2025), and InternLM3-8B (InternLM Team, 2025). For each model, we generate Anchors and repair outcomes with deterministic decoding and use them to train a separate router. The routers share data splits, architecture, losses, hyperparameters, and seeds. Appendix D.1 lists model identities, quantization settings, and inference backends. We train DeBERTa-v3-small for up to six epochs with seeds 42–44, stopping after two epochs without a strict improvement in development task score and retaining the first checkpoint with the highest score. For Qwen, full ARC and Question-Only have both main and ablation-phase runs; Answer-Only uses ablation-phase runs. Component comparisons use the ablation-phase full model as the reference. Appendix F.2 also reports a separate deterministic evaluation of ARC and Question-Only over the same three seeds.

### 6.2 Tasks and Task-Cue Perturbations

The in-distribution (ID) test covers logical reasoning with ProofWriter (Tafjord et al., 2021), temporal reasoning with CoTempQA (Su et al., 2024), mathematics with GSM8K (Cobbe et al., 2021), spatial reasoning with StepGame at 1–5 hops (Shi et al., 2022), and evidence QA with HotpotQA distractor (Yang et al., 2018). The splits contain 4,010 training, 1,006 development, and 2,510 ID examples. We group ProofWriter queries by theory and other examples by source record (Appendix A). Hard Mixed adds irrelevant cross-family cues to 510 ProofWriter and 490 CoTempQA ID examples while preserving questions and reference answers. It tests task-cue sensitivity on Qwen and Llama; Gemma and InternLM are evaluated on ID only. Appendix A specifies the perturbations and source pairing.

### 6.3 BASELINES

Baselines test output information, intervention decisions, and repair choice within a shared action library. Policies select actions before accessing outcomes in the shared matrix (Appendix B.3). Retention keeps every answer, whereas Always Self-Refine applies one critique-and-revision LLM call to every Anchor. Question-Only and Answer-Only test the contribution of generated output fields; Flat Router retains full inputs and gain/rank supervision but removes the hierarchy. Random samples available actions uniformly, and Action Oracle selects the best recorded outcome. Always-on Family-Aware and Always-on Task-Type test family-based repair assignment using true and predicted family labels, respectively; Selective Task-Type adds a development-calibrated gate (Appendix E.1). Family-Mode chooses among all actions, including NoSkill, by development reward for each true benchmark family (Appendix I.1). On Qwen, Family-Aware When-Only independently learns a retain/repair gate for a development-selected repair assigned by the true benchmark family. ARC selects repairs without true family labels; Appendix I.2.2 specifies the control.

### 6.4 SCORES, COSTS, AND UNCERTAINTY

Macro Task Score averages scores within datasets and then weights datasets equally. Closed-answer tasks use normalized accuracy; HotpotQA uses answer F1, under the shared answer contracts (Appendix C). We compute differences from unrounded means; one point is 0.01. We report intervention rates, extra LLM calls, and LLM request times averaged over all examples, assigning zero cost to retention. Repair costs exclude initial generation, routing, and offline collection; request times also exclude local parsing and solver execution (Appendix D.5). Tables report mean $\pm$ sample standard deviation over seeds 42–44 for learned routers and 0–2 for Random; fixed policies report one value. Paired intervals use 10,000 source-group bootstrap resamples, weighting groups equally within datasets and datasets equally (Appendices F and G).

The Qwen budget sweep uses the deterministic checkpoints and calibrates thresholds to targets of 0.05, 0.10, 0.15, or 0.20 development calls per example, then reports realized ID calls at these fixed thresholds (Appendix H). We also separate realized repair gains, losses, and zero-gain interventions (Appendix F.3). The Family-Mode Challenge uses test outcomes to identify repairable examples where an available fixed family action is not oracle-best. It is used only for diagnosis; its full selection rule is in Appendix I.1.

## 7 RESULTS

### 7.1 RESULTS ACROSS FOUR MODELS

In the main evaluation across four base models, ARC improves mean ID task score by 4.4–7.2 points over Retention and 1.5–2.5 points over Question-Only (Table 1a). It repairs 10.7–19.6% of examples with 0.136–0.226 additional calls per example (Table 1b). Adding the generated answer improves mean score over Question-Only for every model; adding public reasoning yields the highest mean among input views in the main evaluation (Table 1a; Figure 3a). Appendix Table 19 reports the per-seed paired comparisons and intervals. In the separate deterministic Qwen evaluation, ARC and Question-Only have nearly equal mean ID scores (0.6929 vs. 0.6926), with 24.8% fewer additional calls for ARC (Appendix Table 14).

### 7.2 QUALITY-COST TRADEOFF

With deterministic checkpoints and development-calibrated thresholds, ARC achieves higher mean test scores than Question-Only with fewer calls at the three higher budget settings (Figure 2). At the 0.20 development call target, it scores 0.6881 vs. 0.6845, using 0.1416 vs. 0.1829 test calls: 22.6% fewer calls and 21.3% less recorded request time (Appendix H). At the default threshold, the routers improve similar proportions of ID answers (4.63% vs. 4.66%), while ARC makes fewer zero-gain interventions (6.22% vs. 10.60%; Appendix F.3). This accounts for most of the reduction in intervention frequency.

At its original operating point, ARC uses 0.155 additional calls and 348 ms of recorded LLM request time per ID example. Compared with Always Self-Refine, it raises task score from 0.5957 to

Table 1: Comparison of repair quality, cost, and action selection across four base models. Gains are in points.

| Method / metric | Qwen3-8B | Llama-3.1-8B | Gemma-3-4B | InternLM3-8B |
|---|---|---|---|---|
| (a) ID Macro Task Score ↑ | | | | |
| Retention | 0.6478 | 0.4667 | 0.4326 | 0.4649 |
| Question-Only | 0.6765±0.0152 | 0.5025±0.0015 | 0.4870±0.0007 | 0.4850±0.0168 |
| Answer-Only | 0.6870±0.0032 | 0.5092±0.0043 | 0.4963±0.0027 | 0.4968±0.0059 |
| ARC (ours) | **0.6915**±0.0021 | **0.5254**±0.0002 | **0.5045**±0.0024 | **0.5100**±0.0042 |
| Action Oracle | 0.8628 | 0.7373 | 0.6351 | 0.6615 |
| (b) ARC gains and repair cost on all ID examples | | | | |
| Δ vs. Retention | +4.37 | +5.88 | +7.19 | +4.52 |
| Δ vs. Question-Only | +1.51 | +2.29 | +1.75 | +2.50 |
| Intervention rate (%) | 10.69±0.42 | 19.59±0.57 | 14.13±0.62 | 13.21±0.44 |
| Extra LLM calls | 0.1548±0.0076 | 0.2262±0.0056 | 0.1380±0.0054 | 0.1356±0.0040 |
| (c) Positive-gain selection on repairable examples (%) ↑ | | | | |
| Repairable subset ($n$) | 548 | 689 | 521 | 506 |
| Question-Only | 13.8±7.1 | 15.3±0.9 | 28.0±0.3 | 11.9±9.9 |
| Answer-Only | 18.7±1.6 | 17.3±1.9 | 32.5±1.3 | 18.1±3.7 |
| ARC (ours) | **20.7**±0.9 | **23.7**±0.3 | **36.1**±1.2 | **22.6**±2.1 |

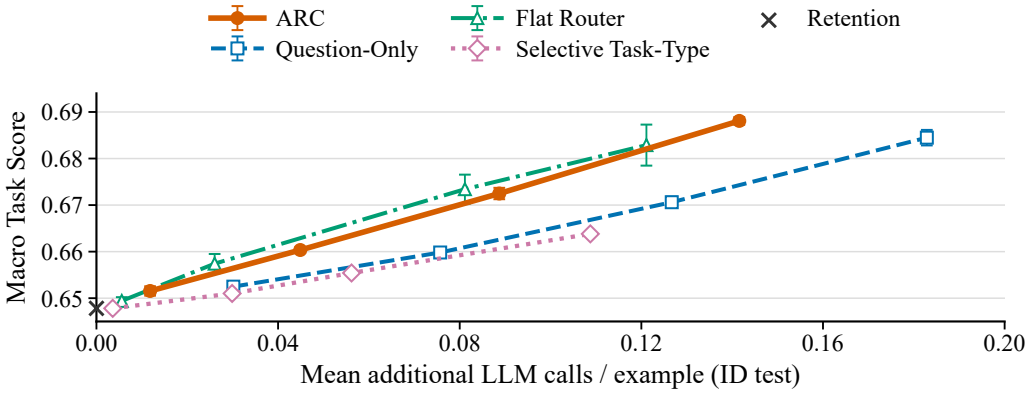



Figure 2: Qwen quality–cost curves at development-calibrated thresholds. ARC and Question-Only use deterministic retraining; Flat Router uses original checkpoints. Error bars show score standard deviation over three seeds; Selective Task-Type and Retention are fixed policies. Lines connect measured points. Appendix H gives the calibration and results.

0.6915 while reducing calls by 84.5% and request time by 90.9%. The always-on family policies achieve higher scores at about 1.18 calls and 2.74 s (Table 3); Section 7.5 discusses the family-aware diagnostic.

### 7.3 Component Ablations

At the default thresholds in the ablation phase, replacing the hierarchy with a flat router produces the largest decrease: 2.23 points on ID and 4.35 on Hard Mixed. Removing gain/rank supervision or retention also lowers both means. Table 2 reports the complete input-view, component, and action-library ablations. Intervention rates vary across ablations; Appendix Figure 4 compares the original routers at development-calibrated thresholds.

Table 2: Ablation study on Qwen3-8B. All rows use ablation-phase training runs; Δ is the change from full ARC in points. Main runs appear in Tables 1 and 3.

| | ID | | Hard Mixed | |
|---|---|---|---|---|
| Variant | Score ↑ | Δ | Score ↑ | Δ |
| ARC (full) | 0.6826±0.0170 | 0.00 | 0.5928±0.0382 | 0.00 |
| Input information | | | | |
| Question-Only | 0.6855±0.0015 | +0.29 | 0.5900±0.0071 | −0.27 |
| Answer-Only | 0.6870±0.0032 | +0.44 | 0.5938±0.0093 | +0.10 |
| Method components | | | | |
| w/o Gain/Rank | 0.6767±0.0123 | −0.59 | 0.5822±0.0286 | −1.05 |
| Flat Router (ablation) | 0.6603±0.0012 | −2.23 | 0.5492±0.0000 | −4.35 |
| w/o NoSkill | 0.6721±0.0159 | −1.05 | 0.5723±0.0324 | −2.04 |
| w/o Repairability Auxiliary | 0.6934±0.0020 | +1.08 | 0.6145±0.0062 | +2.18 |
| w/o SelfRefine | 0.6949±0.0065 | +1.23 | 0.6216±0.0114 | +2.89 |

### 7.4 Repair Choices across Models

Among examples with a positive-gain repair, ARC selects one at a higher mean rate than either reduced input view for all four base models. Such repairs are available for 20–28% of ID examples, and ARC selects one on 20.7–36.1% of this subset (Table 1c; Figure 3b). The selected repair types also vary across models: Qwen selects Temporal and SpatialGraph, while Llama also uses SelfRefine, Datalog, and EvidenceRetrieval. Gemma mainly uses SelfRefine and SpatialGraph; InternLM also uses PythonMath and EvidenceRetrieval (Figure 3c). Each base model generates both the initial answers and LLM-based repairs, which together determine the gains used for training. Appendix Table 20 also reports retention rates.

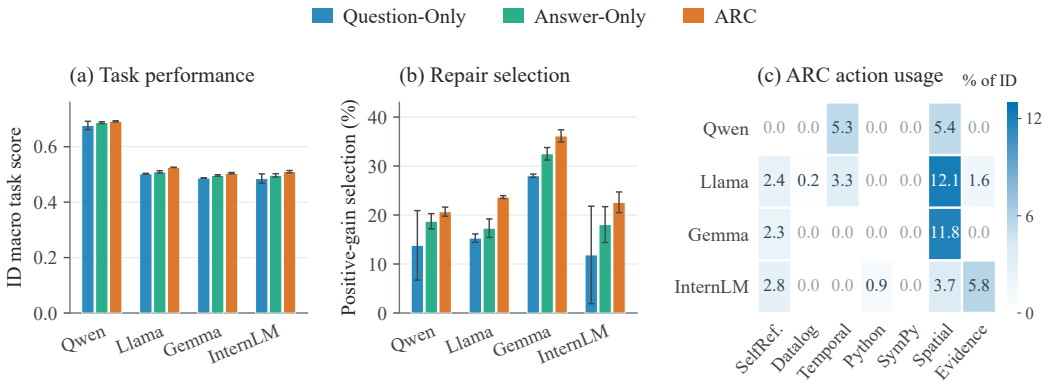



Figure 3: Inputs and repair choices across base models. (a) ID score by input view: Question-Only reads the question, choices, and context; Answer-Only adds the answer; ARC also adds public reasoning. (b) Rate of selecting a positive-gain repair among examples where one is available. Error bars show sample standard deviation over three seeds. (c) Mean selection rates for repairs, as percentages of all ID examples. SelfRef., Python, Spatial, and Evidence denote SelfRefine, PythonMath, SpatialGraph, and EvidenceRetrieval.

### 7.5 POLICY DIAGNOSTICS

The Family-Mode Challenge tests action choices on repairable examples where the development-selected family action is available but not oracle-best. On this subset, ARC achieves higher mean scores than Family-Mode on Llama and InternLM, with similar mean scores on Qwen and Gemma. Appendix I.1 reports subset sizes, action coverage, and the complete per-seed comparisons.

On the full Qwen ID split, restricting the frozen router to a development-selected repair for each family yields nearly identical quality and cost (Appendix I.2.1). When repairs are assigned using true family labels, Family-Aware When-Only reaches 0.6997 at 0.1442 test calls, compared with 0.6886 at 0.1426 for the original ARC router (Appendix Figure 4; Appendix I.2.2).

Hard Mixed tests sensitivity to irrelevant task cues. On the perturbed logic and temporal examples, ARC improves over Question-Only on both evaluated models: 0.6118 vs. 0.5798 on Qwen (Table 3) and 0.3719 vs. 0.3420 on Llama (Appendix Table 22).

Table 3: Comparison with baselines on Qwen3-8B at the original operating points. Costs are per ID example.

| | Macro Task Score ↑ | | ID repair cost | | |
|---|---|---|---|---|---|
| Method | ID | Hard Mixed | Repair (%) | Extra calls | LLM time (ms) |
| Retention | 0.6478 | 0.5492 | 0.00 | 0.000 | 0.0 |
| Selective repair | | | | | |
| Selective Task-Type | 0.6554 | 0.5492 | 6.22 | 0.062 | 84.5 |
| Flat Router | 0.6606±0.0014 | 0.5492±0.0000 | 4.08 | 0.038 | 62.2 |
| Question-Only | 0.6765±0.0152 | 0.5798±0.0296 | 11.66 | 0.149 | 288.5 |
| Answer-Only | 0.6870±0.0032 | 0.5938±0.0093 | 13.45 | 0.185 | 394.1 |
| ARC (ours) | **0.6915**±0.0021 | **0.6118**±0.0087 | 10.69 | 0.155 | 347.9 |
| Random and always-on policies | | | | | |
| Random | 0.6541±0.0047 | 0.5593±0.0065 | 86.63 | 1.058 | 2674.5 |
| Always Self-Refine | 0.5957 | 0.5041 | 100.00 | 1.000 | 3829.9 |
| Always-on Task-Type | 0.7771 | 0.7694 | 100.00 | 1.183 | 2753.8 |
| Always-on Family-Aware | 0.7875 | 0.7694 | 100.00 | 1.174 | 2736.3 |
| Action Oracle | 0.8628 | 0.9034 | — | — | — |

## 8 DISCUSSION AND LIMITATIONS

ARC targets settings where labeled training data are available for offline repair evaluation before selective repair at deployment. Our evaluation uses quantized 4B–8B instruction models, five English reasoning task families, a fixed repair library, and $\lambda = 0$, with a separate router trained for each base model. Hard Mixed tests fixed task-cue perturbations of ID examples; transfer to unseen datasets or base models remains untested.

## 9 CONCLUSION

ARC learns when and how to repair an answer from matched outcomes of retention and heterogeneous repairs applied to the same Anchor. At deployment, the router uses the generated output to retain the answer or execute one repair with the base model fixed. The main evaluation across four small language models shows improved answer quality over retention at low intervention rates. The deterministic Qwen budget sweep also identifies settings with higher scores and fewer calls than question-only routing. These results support learning from offline repair comparisons to improve answer quality with selective additional computation.

### AI USE STATEMENT

We used ChatGPT to polish the language of this paper.

### REPRODUCIBILITY STATEMENT

The appendix gives the data splits, router inputs, repair procedures, prompts, training settings, scoring rules, and cost definitions, along with per-seed results and statistical comparisons. The accompanying code includes matched-outcome supervision, router training and evaluation, threshold calibration, and Self-Refine, Temporal, and SpatialGraph backends, with runnable examples and tests.

## REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4164.

Justin Chen, Archiki Prasad, Swarnadeep Saha, Elias Stengel-Eskin, and Mohit Bansal. MAgICoRe: Multi-agent, iterative, coarse-to-fine refinement for reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32663–32686, 2025. doi: 10.18653/v1/2025.emnlp-main.1660. URL https://aclanthology.org/2025.emnlp-main.1660/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. BEST-route: Adaptive LLM routing with test-time optimal compute. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13870–13884, 2025. URL https://proceedings.mlr.press/v267/ding25d.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://www.jmlr.org/papers/v23/21-0998.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.11738.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021. URL https://arxiv.org/abs/2111.09543.

InternLM Team. InternLM3-8B-Instruct model card. Hugging Face model repository, 2025. URL https://huggingface.co/internlm/internlm3-8b-instruct. Accessed 2026-09-03.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.12917.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-2019.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. StepGame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11321–11329, 2022. doi: 10.1609/aaai.v36i10.21383. URL https://ojs.aaai.org/index.php/AAAI/article/view/21383.

Zhaochen Su, Juntao Li, Jun Zhang, Tong Zhu, Xiaoye Qu, Pan Zhou, Bowen Yan, Yu Cheng, and Min Zhang. Living in the moment: Can large language models grasp co-temporal reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13014–13033. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.703. URL https://aclanthology.org/2024.acl-long.703/.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Chih-Hsuan Yang, Jingyan Jiang, Cheng-Hau Yang, Vikram Vasudevan, Huihuo Zheng, Venkatram Vishwanath, and Rajeev Thakur. LLMs can predict failure risk, but struggle to predict which collaboration protocol pays off: Cost-aware protocol routing across reasoning tasks. *arXiv preprint arXiv:2608.14927*, 2026. URL https://arxiv.org/abs/2608.14927.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

# APPENDIX

## A DATASETS AND SPLITS

Table 4 summarizes the data composition. We construct the splits before generating Anchors or executing actions, using sampling seed 42. All four base models share these sample identifiers and partitions. Appendix C specifies task scoring.

Table 4: Dataset composition after partitioning by source group.

| Split | Composition | $N$ |
|---|---|---|
| Router train | ProofWriter 810; each other ID dataset 800 | 4,010 |
| Router dev | ProofWriter 206; each other ID dataset 200 | 1,006 |
| ID test | ProofWriter 510; each other ID dataset 500 | 2,510 |
| Hard Mixed | ProofWriter 510; CoTempQA 490 (perturbed ID copies) | 1,000 |

### A.1 SOURCE RECORDS AND PARTITIONING

ProofWriter uses the open-world depth-5 files from release V2020.12.3. Router training and development use examples from its training file, and ID evaluation uses examples from its test file. Each theory expands into multiple queries; all queries retain the original theory identifier as their source group. GSM8K uses the `main` configuration of `openai/gsm8k`, with router training and development drawn from the official training split and ID from the test split. StepGame uses `michaelszx/StepGame` with the same split assignment, restricting ID test records to 1–5 hops. HotpotQA uses the `distractor` configuration of `hotpotqa/hotpot_qa`, with router training and development drawn from training and ID drawn from validation. CoTempQA is partitioned from the `Warrieryes/CotempQA` training records.

CoTempQA groups examples by source record. GSM8K and StepGame use the source split and row index when an original identifier is absent; HotpotQA uses its original record identifier. These four datasets have one processed example per source group, with a dataset prefix in each group identifier.

Within each source partition, we sort group identifiers, shuffle them with seed 42, and allocate whole groups sequentially toward the requested sample counts. Allocation stops when adding the next group would increase the absolute difference from the target; that group remains available for the next split. The targets are 800 training, 200 development, and 500 ID examples per dataset. Whole-theory grouping accounts for the slightly different ProofWriter counts. Training, development, and ID contain 3,239, 810, and 2,024 source groups, respectively. These splits share no group identifiers or exact public-input tuples (dataset, question, context, choices).

### A.2 HARD MIXED CONSTRUCTION

Hard Mixed uses the first 1,000 examples in the ID split, ordered by dataset. This gives 510 ProofWriter and 490 CoTempQA examples in 514 source groups. For ProofWriter, we append "The unrelated event occurred on 2001-01-01." to the context. For CoTempQA, we append "If the unrelated object is blue then it is quiet." The cue is separated from the original context by a newline. The question, choices, reference answer, dataset label, and source-group identifier remain unchanged. Each perturbed example is paired with its original ID example for task-cue sensitivity evaluation.

## B ACTIONS, INTERMEDIATE REPRESENTATIONS, AND SOLVERS

Each action-matrix row stores the sample and source-group identifiers, Anchor score, and a record for each action. Action records contain the final answer and reasoning, status, reward, gain, call and token counts, recorded latency, and execution trace. An unavailable action has `status=unavailable`; its record contains the unchanged Anchor, zero gain, and zero cost as placeholders. It is excluded from routing, gain regression, and ranking. Available actions have status

Table 5: Repair actions and their functions.

| ID | Action | Intervention |
|---|---|---|
| 0 | NoSkill | Return the Anchor without an additional call. |
| 1 | SelfRefine | Ask the base model to critique and revise the Anchor in one LLM call; return the revision directly. |
| 2 | Datalog | Extract facts and rules and run symbolic forward inference. |
| 3 | Temporal | Extract temporal entities and relations and perform temporal reasoning. |
| 4 | PythonMath | Extract and run a restricted arithmetic program. |
| 5 | SymPy | Extract up to eight equations and solve them symbolically. |
| 6 | SpatialGraph | Normalize and compose spatial relations over a graph. |
| 7 | EvidenceRetrieval | Rank context chunks by lexical relevance and retain the highest-ranked evidence. |

`ok`, `abstain`, or `error`. The mask encodes only pre-execution availability; the router does not read the recorded execution status. We train the router on targets from the training split and use development data for checkpoint selection and policy calibration.

### B.1 Availability and Execution

Availability is a local pre-execution check. NoSkill and SelfRefine are always available. Datalog requires nonempty question and context strings after whitespace removal. Temporal, PythonMath, SymPy, and SpatialGraph require a nonempty question. EvidenceRetrieval requires at least two context chunks. These checks use no generated answer, reasoning, task label, scorer, or LLM call. All input-view baselines use the same mask. The extractor may subsequently reject the input as unsuitable for an available action; this is an execution outcome, not a change to the pre-execution mask.

Evidence chunks are the sentences in the public paragraph records when available. Otherwise, we split the context at newlines or whitespace following sentence-ending punctuation and discard empty strings. BM25 uses word tokens normalized by case folding. It retrieves the top 12 chunks for the question concatenated with the rendered Anchor answer, breaking score ties by source order. We retain at most eight chunks and three per title, then restore their source order. Thus, retrieval uses the Anchor after selection, while its availability check does not.

### B.2 Structured Outputs and Failure Rules

Datalog extracts `a`, `f`, `r`, and `q`: an applicability flag, facts, rules, and query. An atom is `[predicate, arguments, negation]`, and a rule lists premises followed by its conclusion. The parser permits at most 128 facts and rules in total, requires the facts and query to be ground, and requires every conclusion variable to occur in a premise. Predicate constraints are derived from public task words. Forward chaining continues until no fact is added; it returns True if the query is derived, False if its negation is derived, and Unknown otherwise. Invalid or inapplicable representations cause abstention.

Temporal uses records of the form `[id, subject, predicate, object, start, end]` and a query specifying the target event, candidate filters, return field, and exclusions. When the public context matches the deterministic fact parser, those records are fixed and the model extracts only the query. The target and query slots are aligned with the public question before interval reasoning. Malformed representations, unresolved queries, and an Unknown solver result cause abstention. Dates retain their year, month, or day precision; the code specifies the normalization and interval rules.

PythonMath extracts `applicable`, `code`, `answer_variable`, and `explanation`. AST validation rejects imports, file or network access, function definitions, unsupported calls, and unsafe attributes. The program must assign `answer`; it runs in a separate process with a two-second timeout and must return a numeric scalar whose string representation has at most 128 characters. A

validation failure, runtime error, or timeout produces an error status. An extraction marked inapplicable causes abstention.

SymPy extracts declared symbols, equations, a target, and optional real-valued assumptions. The parser accepts at most eight equations, restricts identifiers and expression characters, and permits `sqrt` and `Abs`. Invalid expressions produce an error status; inapplicable requests or an empty solution set cause abstention. Multiple solutions are passed to the correction composer as structured assistance.

SpatialGraph first parses spatial constraints and the query from public text. If either is missing, the model extracts `applicable`, a relation list, and the query entities. Relations map to nine two-dimensional directions, including overlap, and are composed over the graph. Invalid extractions, missing query entities, disconnected paths, and unresolved conflicting constraints cause abstention. The deterministic path uses no extraction call.

For model-based extraction, temperature is zero, top-$p$ is one, and the decoding seed is 42. Output limits are 1,024 tokens for Datalog, 768 for Temporal, and 384 for PythonMath, SymPy, and the SpatialGraph fallback. The shared correction call also uses 384 tokens. The JSON parser first attempts to parse the complete response, then falls back to the first balanced JSON object. If the backend abstains or returns an error, we retain the Anchor and count the recorded cost of the attempt. A completed correction is scored under the answer contract even when its returned answer violates that contract.

### B.3 Cached Policy Evaluation

Cached outcomes are reused only when the inputs and execution settings match, including the model, generation settings, prompts, and solver versions. Anchor-dependent repairs also require the same Anchor; correction calls require the same assistance and answer contract.

Cached policy evaluation selects an action from public router inputs and the pre-execution mask before looking up its final answer and cost. Rewards, execution traces, and post-execution statuses are used only for evaluation. The input-view comparisons share the target Anchor and cached action outcomes, varying the information supplied to the router.

### B.4 Recorded Training Examples

Table 6 gives the Qwen CoTempQA training examples discussed in Section 3. For each score pattern, we take the first Temporal execution with status `ok` in file order.

Table 6: Recorded answers and scores for three Qwen training examples under the Temporal action. Questions are shortened.

| | |
|---|---|
| Question | Which employers did Pier Paolo Pandolfi work for while Christopher Celenza worked for Georgetown University? |
| Anchor answer | `["Beth Israel Deaconess Medical Center"]` |
| Repaired answer | `["Beth Israel Deaconess Medical Center", "Harvard Medical School"]` |
| Score change | $0 \rightarrow 1$; gain $+1$ |
| Question | Where did Samuel S. Wilks work while Peter Hilton attended the University of Oxford? |
| Anchor answer | `["Princeton University"]` |
| Repaired answer | `["Princeton University"]` |
| Score change | $1 \rightarrow 1$; gain $0$ |
| Question | Who belonged to the Democratic Labour Party at the same time as Amazonino Mendes? |
| Anchor answer | `["Ciro Gomes", "Cristovam Buarque"]` |
| Repaired answer | `[]` |
| Score change | $1 \rightarrow 0$; gain $-1$ |

### B.5 Gain Distribution within Task Families

Table 7 summarizes all seven repairs on the 4,010 Qwen training Anchors. For each family–action pair, we count positive, zero, and negative final-answer gains among available outcomes, including abstentions and execution errors. These counts describe the training outcomes used for supervision, before router selection.

Table 7: Gain signs of all available repairs in the Qwen training action matrix.

| Family | Repair | Available | Positive | Zero | Negative |
|---|---|---|---|---|---|
| Logic | SelfRefine | 810 | 103 (12.7%) | 562 (69.4%) | 145 (17.9%) |
| | Datalog | 810 | 191 (23.6%) | 429 (53.0%) | 190 (23.5%) |
| | Temporal | 810 | 0 (0.0%) | 809 (99.9%) | 1 (0.1%) |
| | PythonMath | 810 | 46 (5.7%) | 702 (86.7%) | 62 (7.7%) |
| | SymPy | 810 | 0 (0.0%) | 810 (100.0%) | 0 (0.0%) |
| | SpatialGraph | 810 | 0 (0.0%) | 810 (100.0%) | 0 (0.0%) |
| | EvidenceRetrieval | 810 | 35 (4.3%) | 751 (92.7%) | 24 (3.0%) |
| Temporal | SelfRefine | 800 | 58 (7.2%) | 646 (80.8%) | 96 (12.0%) |
| | Datalog | 800 | 4 (0.5%) | 788 (98.5%) | 8 (1.0%) |
| | Temporal | 800 | 352 (44.0%) | 438 (54.8%) | 10 (1.2%) |
| | PythonMath | 800 | 29 (3.6%) | 735 (91.9%) | 36 (4.5%) |
| | SymPy | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | SpatialGraph | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | EvidenceRetrieval | 800 | 17 (2.1%) | 780 (97.5%) | 3 (0.4%) |
| Math | SelfRefine | 800 | 25 (3.1%) | 755 (94.4%) | 20 (2.5%) |
| | Datalog | 0 | – | – | – |
| | Temporal | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | PythonMath | 800 | 21 (2.6%) | 755 (94.4%) | 24 (3.0%) |
| | SymPy | 800 | 1 (0.1%) | 796 (99.5%) | 3 (0.4%) |
| | SpatialGraph | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | EvidenceRetrieval | 0 | – | – | – |
| Spatial | SelfRefine | 800 | 37 (4.6%) | 641 (80.1%) | 122 (15.2%) |
| | Datalog | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | Temporal | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | PythonMath | 800 | 3 (0.4%) | 794 (99.2%) | 3 (0.4%) |
| | SymPy | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | SpatialGraph | 800 | 191 (23.9%) | 596 (74.5%) | 13 (1.6%) |
| | EvidenceRetrieval | 647 | 7 (1.1%) | 638 (98.6%) | 2 (0.3%) |
| Evidence | SelfRefine | 800 | 24 (3.0%) | 686 (85.8%) | 90 (11.2%) |
| | Datalog | 800 | 3 (0.4%) | 789 (98.6%) | 8 (1.0%) |
| | Temporal | 800 | 0 (0.0%) | 797 (99.6%) | 3 (0.4%) |
| | PythonMath | 800 | 11 (1.4%) | 756 (94.5%) | 33 (4.1%) |
| | SymPy | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | SpatialGraph | 800 | 0 (0.0%) | 800 (100.0%) | 0 (0.0%) |
| | EvidenceRetrieval | 800 | 12 (1.5%) | 784 (98.0%) | 4 (0.5%) |

Each entry is a count (percentage of available outcomes for that family–action pair). The matrix contains 4,010 examples. Outcomes with status `ok`, `abstain`, or `error` are included; only `unavailable` is excluded. Gain is final task score minus Anchor score; HotpotQA uses F1 and the other tasks use binary scores.

## C Prompts and Answer Contracts

All four model configurations use the same prompt roles and answer contracts. Each base model generates its own Anchor, structured assistance, and final correction. The Anchor prompt requests a single public JSON object with `reasoning` and `answer` fields, without a Markdown wrapper. SelfRefine requests `critique`, `reasoning`, and `answer`. Depending on the action, specialized extractors produce intermediate representations in strict JSON, restricted arithmetic programs, or symbolic equation specifications. The shared correction composer receives the original task, public Anchor fields, selected assistance, and task answer contract, then returns `reasoning` and

`answer`. Hidden or private reasoning is discarded and never provided to the router. Appendix C.1 records the prompt versions and shared system prompts.

Closed-answer datasets use normalized 0–1 correctness scores computed by the corresponding evaluation adapters. ProofWriter maps equivalent labels to canonical forms before exact comparison. CoTempQA requires a nonempty JSON array of strings and checks exact equality between normalized entity sets. GSM8K compares normalized numeric answers for equality. StepGame maps spatial-relation aliases to canonical forms before exact comparison. Hard Mixed uses the scorer of the corresponding source ID example. HotpotQA uses normalized answer-span token F1 as its primary utility. Its binary correction and harm diagnostics use exact match, and F1 gain is reported separately.

CoTempQA requires a nonempty JSON array of strings; scalar strings and arrays serialized as strings do not satisfy the contract. For Gemma, 7 of 500 ID Anchors meet this requirement. The matched matrix contains 38 repairable examples and 37 Temporal executions with positive reward. The router does not select Temporal, and the final Temporal-family score under this contract is zero. All methods use the same parser and answer contract, so the scores jointly reflect answer correctness and format compliance.

### C.1 Shared System Prompts

The shared system prompts below define initial generation, self-refinement, and correction. Task-specific answer contracts and structured-assistance schemas are described above. The task input is provided with the applicable answer contract.

#### C.1.1 Initial generation

Prompt version: `anchor-v2.2-answer-contract`.

```
You are solving a reasoning problem. Use only the information given in the
task unless ordinary arithmetic or language understanding is required.
Return exactly one JSON object with keys:
- reasoning: a concise but explicit reasoning trace
- answer: the final answer only
Do not add markdown outside the JSON object.
```

#### C.1.2 Self-refinement

Prompt version: `self-refine-v2.2-answer-contract`.

```
You are revising your own previous solution. The previous answer may be
correct or incorrect.
Do not assume it is wrong. Check the reasoning against the original task,
identify the single most important possible flaw if one exists, then solve
again.
Return exactly one JSON object with keys:
- critique
- reasoning
- answer
Do not add markdown outside the JSON.
```

#### C.1.3 Correction

Prompt version: `correction-v2.2-answer-contract`.

```
You are correcting a previous solution using selected reasoning assistance.
The assistance may be useful, partially useful, or irrelevant. Check it
against the original task and do not blindly copy it.
Return exactly one JSON object with keys:
- reasoning
- answer
Do not add markdown outside the JSON.
```

# D Model and Router Implementation

## D.1 Base-Model Configurations

Table 8: Model identities and inference backends.

| Base LM | Repository and identity | Quantization | Backend |
|---|---|---|---|
| Qwen3-8B | `Qwen/Qwen3-8B-GGUF` | GGUF `Q4_K_M` | Local OpenAI-compatible endpoint |
| Llama-3.1-8B-Instruct | `meta-llama/Llama-3.1-8B-Instruct` | GGUF `Q4_K_M` | llama.cpp b10615 |
| Gemma-3-4B-it | `google/gemma-3-4b-it` | GGUF `Q4_K_M` | llama.cpp b10615 |
| InternLM3-8B-Instruct | `internlm/internlm3-8b-instruct` | GGUF `Q4_K_M` | llama.cpp b10615 |

Table 9: Runtime context and public reasoning interface for each base model.

| Base LM | Context | Public reasoning interface |
|---|---|---|
| Qwen3-8B | not recorded | Explicit JSON fields; native thinking off |
| Llama-3.1-8B-Instruct | 16,384 | Explicit JSON fields; native thinking off |
| Gemma-3-4B-it | 16,384 | Explicit JSON fields; native thinking off |
| InternLM3-8B-Instruct | 16,384 | Explicit JSON fields; native thinking off |

The Qwen configuration uses `Qwen3-8B-Q4_K_M.gguf`. All configurations use text-only inputs and the public-reasoning schema in Appendix C.

## D.2 Router Inputs

BaseView serializes the question, choices, and context view in that order, using `[QUESTION]`, `[CHOICES]`, and `[CONTEXT_VIEW]` delimiters. Absent choices are rendered as `NONE`. Contexts longer than 1,200 characters are replaced by their first and last 600 characters concatenated directly. ARC appends `[ANCHOR_REASONING]` and the public reasoning, then `[ANCHOR_ANSWER]` and the rendered answer. Answer-Only appends only the latter field; Question-Only uses BaseView alone. The complete string is tokenized and right-truncated to 512 tokens, including special tokens. No token quota is reserved for the appended Anchor fields.

The ten numeric features are the character counts of the question, full context, public reasoning, and rendered answer, the number of choices, and five cue counts. Cue counts use only the question concatenated with the full public context, before truncation; they exclude choices and all Anchor fields. The patterns count numbers, temporal expressions, rule words, spatial words, and negations. Length features for omitted Anchor fields are set to zero. These numeric features are computed before token truncation and supplied separately from the text encoder.

Numbers are signed integers or decimals. Temporal expressions include four-digit years with optional month and day, or the words before, after, during, year, month, and day. Rule words are if, then, implies, therefore, all, and every. Spatial words are left, right, above, below, upper, lower, north, south, east, west, and overlap. Negations are not, never, no, neither, and cannot. Word patterns are case-insensitive and use word boundaries. The reference source records the exact regular expressions.

## D.3 Router Training

The router uses DeBERTa-v3-small with a maximum input length of 512 tokens, a 512-dimensional projection, 256-dimensional hidden layers for the gates and gain predictors, and a dropout rate of 0.1.

We train with AdamW for at most six epochs, using a batch size of 16 and two gradient accumulation steps. The encoder and head learning rates are $2 \times 10^{-5}$ and $10^{-4}$, respectively. We use a weight decay of 0.01, a warmup ratio of 0.06, a gradient clipping threshold of 1.0, and an early-stopping patience of 2. The weight for route class $s$ is $\mathrm{clip}(\sqrt{N/(|\mathcal{A}| \max(n_s, 1))}, 0.5, 3)$, where $n_s$ is its training-label count. For weighted cross-entropy, we divide the weighted loss sum by the sum of the target-class weights in the minibatch. Gain regression averages Huber loss with threshold 1 over available non-NoSkill actions. For ranking, we form pairs of available actions whose gains differ by at least 0.25. We shuffle these pairs using seed $42+$ the minibatch row index and retain at most 12 pairs per example. The ranking loss is zero when no pair qualifies. The six auxiliary labels indicate whether each repair group contains an available positive-gain action: SelfRefine, Datalog, Temporal, either mathematics action, SpatialGraph, or EvidenceRetrieval.

All four configurations use the same router architecture, feature construction, training objective, hyperparameter schedule, and seed set. We train a separate router for each base model using the Anchors and matched action matrix generated for that model.

### D.4 Field Retention after Tokenization

Table 10 shows how much answer and reasoning text remains after tokenization and truncation to 512 tokens. A field token is counted when its character offsets overlap the field content; delimiters and special tokens are excluded. The denominator $n$ counts only examples with nonempty tokenized fields. Full and Zero report the percentages of examples retaining all or none of those field tokens, respectively. Tokens is the mean fraction of field tokens retained across these examples.

Table 10: Retention of generated-output fields in the ID router inputs.

| Model | View | Field | $n$ | Full (%) | Zero (%) | Tokens (%) |
|---|---|---|---|---|---|---|
| Qwen | ARC | Reasoning | 2,510 | 99.80 | 0.00 | 99.98 |
| Qwen | ARC | Answer | 2,510 | 99.56 | 0.24 | 99.64 |
| Qwen | Answer-Only | Answer | 2,510 | 100.00 | 0.00 | 100.00 |
| Llama | ARC | Reasoning | 2,215 | 96.66 | 0.00 | 99.32 |
| Llama | ARC | Answer | 2,480 | 95.85 | 3.10 | 96.56 |
| Llama | Answer-Only | Answer | 2,480 | 99.03 | 0.00 | 99.71 |
| Gemma | ARC | Reasoning | 2,396 | 99.71 | 0.00 | 99.95 |
| Gemma | ARC | Answer | 2,508 | 98.60 | 0.28 | 99.44 |
| Gemma | Answer-Only | Answer | 2,508 | 99.12 | 0.00 | 99.83 |
| InternLM | ARC | Reasoning | 1,990 | 99.10 | 0.00 | 99.83 |
| InternLM | ARC | Answer | 2,505 | 97.92 | 0.96 | 98.61 |
| InternLM | Answer-Only | Answer | 2,505 | 99.00 | 0.00 | 99.66 |

### D.5 Cost Accounting

We compute repair costs by averaging the recorded call counts, prompt tokens, completion tokens, and LLM request times of selected actions over all test examples, assigning zero to retention. Calls count logical completion requests, including extraction when needed and final correction. Each base model requires its own offline Anchor generation, action execution, and router training; these costs and the common initial generation are excluded from the reported repair costs. Request time is the elapsed time measured by the client, including waiting and transport retries. It excludes local parsing, symbolic or program execution, retrieval, availability checks, and router inference. Timings follow the model-specific inference configurations in Tables 8 and 9.

## E Qwen Baselines and Per-Family Results

Table 11 breaks down the Qwen comparison in Table 3 by task family. Always-on Family-Aware uses gold benchmark-family identity, whereas Always-on Task-Type predicts the family from the public input.

Table 11: Results on each ID benchmark family with Qwen3-8B.

| Family | Retention | ARC | Always-on Task-Type | Always-on Family-Aware | Oracle |
|---|---|---|---|---|---|
| Logic | .6627 | .6627±.0000 | .6863 | .6863 | .9392 |
| Temporal | .4140 | .5307±.0090 | .8460 | .8460 | .8720 |
| Math | .9100 | .9100±.0000 | .9040 | .9040 | .9520 |
| Spatial | .5460 | .6480±.0020 | .7320 | .7840 | .8000 |
| Evidence | .7063 | .7063±.0000 | .7173 | .7173 | .7510 |

### E.1 Selective Task-Type Calibration

The family-to-action map is logic→EvidenceRetrieval, temporal→Temporal, math→SelfRefine, spatial→SpatialGraph, and evidence→EvidenceRetrieval. For the predicted family $\widehat{f}$, Selective Task-Type assigns the priority score

$$b(x) = p(\widehat{f} \mid x) \max(\mu_{\widehat{f}}^{\mathrm{dev}}, 0), \tag{12}$$

where $\mu_f^{\mathrm{dev}}$ is the mean development gain of the mapped action. We target the development intervention rate of ARC, 0.159708 averaged over seeds 42–44. Calibration yields a threshold of 0.247618 and a development intervention rate of 0.160040. We fix this threshold for all test suites.

On ID, this fixed threshold yields an intervention rate of 0.0622 for Selective Task-Type, compared with 0.1069 for ARC. Selective Task-Type scores 0.6554. The paired source-group-weighted differences for ARC minus Selective Task-Type are $+0.0344$ $[0.0268, 0.0424]$, $+0.0356$ $[0.0280, 0.0436]$, and $+0.0384$ $[0.0304, 0.0468]$ for seeds 42/43/44. The Selective Task-Type scores are 0.6627 for Logic, 0.4140 for Temporal, 0.9100 for Math, 0.5840 for Spatial, and 0.7063 for Evidence; the gate intervenes only on 31.2% of Spatial ID examples.

## F Training and Statistical Details

### F.1 Scores and Paired Comparisons

Tables 18 and 19 report the ID input-view scores and paired comparisons for every model and seed. Table 12 adds Qwen Flat Router ID scores and Hard Mixed scores.

For paired comparisons, we average score differences within each source group, then weight groups equally within each dataset and datasets equally. We resample source groups within datasets 10,000 times to obtain 95% intervals. This group-weighted quantity can differ from the sample-weighted Macro Task Score when source groups contain different numbers of examples.

Table 12: Qwen3-8B per-seed scores supplementing Table 18.

| Suite | Method | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|---|
| ID | Flat Router | .6614 | .6590 | .6614 |
| Hard Mixed | Flat Router | .5492 | .5492 | .5492 |
| Hard Mixed | Question-Only | .5819 | .6084 | .5492 |
| Hard Mixed | ARC | .6053 | .6084 | .6217 |

Table 13 reports Qwen Hard Mixed comparisons for seed 42 under the same bootstrap procedure.

Table 13: Qwen3-8B paired Hard Mixed score differences for seed 42, with source-group-weighted 95% intervals.

| Comparison | Suite | Difference [95% interval] |
|---|---|---|
| ARC−Retention | Hard Mixed | $+.0561$ $[.0418, .0704]$ |
| ARC−Question-Only | Hard Mixed | $+.0235$ $[.0092, .0378]$ |

### F.2 Training Runs and Deterministic Evaluation

We trained full ARC and Question-Only in both the main Qwen experiments and the ablation phase, using seeds 42–44 and the same data and training configuration. Tables 1 and 3 report the main runs. Table 2 uses the ablation-phase full model as the reference for every variant. Checkpoints are selected by development score under the settings in Appendix D. In the ablation phase, ARC scores 0.6630/0.6910/0.6938 on ID and 0.5492/0.6084/0.6206 on Hard Mixed. Question-Only scores 0.6838/0.6862/0.6866 on ID and 0.5819/0.5951/0.5931 on Hard Mixed. Values follow seed order 42/43/44; Table 2 reports their means and standard deviations.

For a separate deterministic evaluation, we retrain ARC and Question-Only with seeds 42–44 using fixed copies of the data, pretrained encoder, and training code. The architecture, losses, optimizer, and hyperparameters remain as specified in Appendix D. We use deterministic PyTorch and cuDNN operations, disable TF32 and cuDNN benchmarking, and set `CUBLAS_WORKSPACE_CONFIG=:4096:8`. For seed $s$, the training and development data loaders use separate generators with seeds $s$ and $s+1$, respectively, and no worker processes. Runs use PyTorch 2.11.0, CUDA 13.0, bfloat16 mixed precision, four CPU threads, and one interop thread. Both methods train for at most six epochs, stop after two epochs without a strict improvement in development score, and select the first checkpoint attaining the highest development score. Test evaluation follows checkpoint selection.

Two independent full runs of each method at seed 42 produce identical numerical training histories, model parameters, selected checkpoints, and test predictions. Table 14 reports scores, costs, and selected epochs for one run per method and seed.

Table 14: Deterministic Qwen retraining. Epoch denotes the selected checkpoint; summary rows report mean ± sample standard deviation over three seeds.

(a) Macro Task Score

| Method | Seed | Epoch | ID | Hard Mixed |
|---|---|---|---|---|
| ARC | 42 | 4 | 0.6910 | 0.6186 |
| ARC | 43 | 6 | 0.6942 | 0.6217 |
| ARC | 44 | 3 | 0.6934 | 0.6237 |
| Question-Only | 42 | 6 | 0.6890 | 0.5972 |
| Question-Only | 43 | 6 | 0.6962 | 0.6288 |
| Question-Only | 44 | 6 | 0.6926 | 0.6053 |
| ARC | Mean | – | 0.6929±0.0017 | 0.6213±0.0026 |
| Question-Only | Mean | – | 0.6926±0.0036 | 0.6104±0.0164 |

(b) Costs and intervention rates

| | Extra calls per example | | Repair rate (%) | |
|---|---|---|---|---|
| Method | ID | Hard Mixed | ID | Hard Mixed |
| ARC | 0.1614±0.0107 | 0.2693±0.0232 | 11.00±0.82 | 13.47±1.16 |
| Question-Only | 0.2145±0.0170 | 0.2473±0.0599 | 15.46±1.09 | 12.37±3.00 |

### F.3 Decomposition of Realized Repair Gains

We evaluate the deterministic Qwen routers at the default threshold on all 2,510 ID examples. For the selected action $a_i$, let $d_i = R_i(a_i) - R_i(\text{NoSkill})$. We partition examples into retention and interventions with positive, zero, or negative gain. These frequencies use all ID examples as the denominator and are averaged over seeds 42–44. A zero-gain intervention can change the answer while leaving its task score unchanged.

Let $\mathcal{M}$ denote the dataset-macro averaging operator used for task score. Positive gain and loss are

$$G^+ = \mathcal{M}[\max(d_i, 0)], \qquad G^- = \mathcal{M}[\max(-d_i, 0)]. \tag{13}$$

The score change over retention is exactly $G^+ - G^-$. Table 15 reports these terms in score points alongside the outcome frequencies. ARC and Question-Only obtain similar positive gains, while

ARC executes fewer zero-gain repairs. Always Self-Refine produces more positive gain but a larger loss, giving a negative net change.

Table 15: Realized Qwen repair outcomes at the default threshold.

| Method | Improve | Same | Harm | Retain | $G^+$ | $G^-$ | Net |
|---|---|---|---|---|---|---|---|
| Always Self-Refine | 6.29 | 81.99 | 11.71 | 0.00 | 6.028 | 11.236 | -5.208 |
| Question-Only | 4.66 | 10.60 | 0.20 | 84.54 | 4.680 | 0.200 | +4.480 |
| ARC | 4.63 | 6.22 | 0.15 | 89.00 | 4.653 | 0.147 | +4.507 |

The first four columns partition all ID examples and are percentages. Same means an executed repair with zero score change; Retain means NoSkill. $G^+$ and $G^-$ are dataset-macro positive gain and loss in score points; Net $= G^+ - G^-$. Learned policies use the deterministic runs, averaged over three seeds; Always Self-Refine is fixed.

Table 16 separates correction and harm on the four binary-score tasks. Correction rate is the fraction of initially incorrect answers made correct; harm rate is the fraction of initially correct answers made incorrect. HotpotQA uses continuous answer F1 and is reported separately as a mean F1 change.

Table 16: Correction and harm on binary-score tasks, with HotpotQA reported separately.

| Method | $0 \to 1$ | Rate (%) | $1 \to 0$ | Rate (%) | HotpotQA $\Delta$F1 |
|---|---|---|---|---|---|
| Always Self-Refine | 136.00 | 18.45 | 233.00 | 18.30 | -0.0679 |
| Question-Only | 117.00 | 15.88 | 5.00 | 0.39 | +0.0000 |
| ARC | 116.33 | 15.78 | 3.67 | 0.29 | +0.0000 |

Binary tasks contain 2,010 ID examples: 737 initially incorrect and 1,273 initially correct. The two rates use these respective denominators. HotpotQA contains 500 examples and uses mean answer F1 change. Counts and rates for learned policies are averaged over three seeds.

# G Additional Results across Base Models

All configurations share the same ID sample identifiers, source-group partition, router architecture, objectives, feature construction, hyperparameters, and seeds. Each configuration uses its own base-model inference profile, generated Anchors, and matched action matrix.

## G.1 Incremental Repair Cost

Table 17 adds token counts and recorded LLM request times to the repair costs in Table 1.

Table 17: ID repair costs of ARC across four base models. Latency is recorded LLM request time.

| Base LM | Intervention rate | Mean extra LLM calls | Prompt tokens | Completion tokens | Added latency (ms) |
|---|---|---|---|---|---|
| Qwen3-8B | 10.69±0.42% | .1548±.0076 | 130.1±7.2 | 15.3±0.7 | 347.9±16.0 |
| Llama-3.1-8B | 19.59±0.57% | .2262±.0056 | 143.9±2.4 | 24.2±1.9 | 271.8±18.0 |
| Gemma-3-4B | 14.13±0.62% | .1380±.0054 | 65.5±3.4 | 15.3±0.6 | 133.8±5.2 |
| InternLM3-8B | 13.21±0.44% | .1356±.0040 | 66.4±4.8 | 30.7±5.9 | 344.9±61.7 |

### G.2 Per-Seed Scores and Paired Bootstrap

Table 18: Results on the common ID test set by input view, base model, and seed.

| Base LM | Seed | Question-Only | Answer-Only | ARC |
|---|---|---|---|---|
| Qwen3-8B | 42 | .6834 | .6834 | .6898 |
| Qwen3-8B | 43 | .6870 | .6894 | .6910 |
| Qwen3-8B | 44 | .6590 | .6882 | .6938 |
| Llama-3.1-8B | 42 | .5019 | .5101 | .5257 |
| Llama-3.1-8B | 43 | .5043 | .5129 | .5253 |
| Llama-3.1-8B | 44 | .5014 | .5045 | .5253 |
| Gemma-3-4B | 42 | .4874 | .4934 | .5070 |
| Gemma-3-4B | 43 | .4874 | .4970 | .5022 |
| Gemma-3-4B | 44 | .4862 | .4986 | .5042 |
| InternLM3-8B | 42 | .4657 | .5028 | .5146 |
| InternLM3-8B | 43 | .4961 | .4966 | .5064 |
| InternLM3-8B | 44 | .4933 | .4910 | .5091 |

Table 19: Paired ID score differences by model and seed, with source-group-weighted 95% intervals.

| Base LM | Comparison | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|---|
| Qwen3-8B | ARC−Retention | +0.0420 [+0.0340, +0.0500] | +0.0432 [+0.0352, +0.0512] | +0.0460 [+0.0380, +0.0544] |
| Qwen3-8B | ARC−Question-Only | +0.0064 [-0.0004, +0.0132] | +0.0040 [-0.0024, +0.0104] | +0.0348 [+0.0268, +0.0428] |
| Qwen3-8B | ARC−Answer-Only | +0.0064 [-0.0004, +0.0132] | +0.0016 [-0.0048, +0.0080] | +0.0056 [-0.0004, +0.0116] |
| Llama-3.1-8B | ARC−Retention | +0.0585 [+0.0483, +0.0689] | +0.0586 [+0.0488, +0.0688] | +0.0587 [+0.0490, +0.0685] |
| Llama-3.1-8B | ARC−Question-Only | +0.0232 [+0.0152, +0.0317] | +0.0216 [+0.0147, +0.0289] | +0.0239 [+0.0168, +0.0313] |
| Llama-3.1-8B | ARC−Answer-Only | +0.0153 [+0.0096, +0.0211] | +0.0126 [+0.0067, +0.0185] | +0.0208 [+0.0141, +0.0279] |
| Gemma-3-4B | ARC−Retention | +0.0744 [+0.0648, +0.0844] | +0.0696 [+0.0600, +0.0792] | +0.0719 [+0.0624, +0.0818] |
| Gemma-3-4B | ARC−Question-Only | +0.0196 [+0.0140, +0.0256] | +0.0148 [+0.0092, +0.0204] | +0.0183 [+0.0128, +0.0243] |
| Gemma-3-4B | ARC−Answer-Only | +0.0132 [+0.0081, +0.0187] | +0.0048 [0.0000, +0.0095] | +0.0055 [+0.0011, +0.0100] |
| InternLM3-8B | ARC−Retention | +0.0524 [+0.0401, +0.0653] | +0.0431 [+0.0316, +0.0552] | +0.0460 [+0.0344, +0.0579] |
| InternLM3-8B | ARC−Question-Only | +0.0516 [+0.0393, +0.0645] | +0.0095 [-0.0040, +0.0238] | +0.0142 [+0.0014, +0.0273] |
| InternLM3-8B | ARC−Answer-Only | +0.0113 [-0.0020, +0.0247] | +0.0088 [-0.0048, +0.0231] | +0.0185 [+0.0086, +0.0298] |

### G.3 Model-Specific Repairable Subsets

A repairable example has at least one available positive-gain action other than NoSkill in the matched test action matrix. These subsets are defined separately for each base model and used for evaluation. Table 20 gives their sizes, positive-gain selection rates, and retention rates.

Table 20: Results on model-specific repairable subsets. Selection and retention rates use the repairable subset as the denominator.

| Base LM | Repairable $n$ (rate) | Question-Only positive-gain selection | Answer-Only positive-gain selection | ARC positive-gain selection | ARC retention rate |
|---|---|---|---|---|---|
| Qwen3-8B | 548 (21.8%) | 13.8±7.1% | 18.7±1.6% | 20.7±0.9% | 78.2±1.0% |
| Llama-3.1-8B | 689 (27.5%) | 15.3±0.9% | 17.3±1.9% | 23.7±0.3% | 70.5±1.4% |
| Gemma-3-4B | 521 (20.8%) | 28.0±0.3% | 32.5±1.3% | 36.1±1.2% | 61.7±1.3% |
| InternLM3-8B | 506 (20.2%) | 11.9±9.9% | 18.1±3.7% | 22.6±2.1% | 64.2±1.2% |

### G.4 Action Distributions

Table 21: ID action-selection rates of ARC (%). Python, Spatial, and Evidence denote PythonMath, SpatialGraph, and EvidenceRetrieval.

| Base LM | NoSkill | SelfRefine | Datalog | Temporal | Python | SymPy | Spatial | Evidence |
|---|---|---|---|---|---|---|---|---|
| Qwen3-8B | 89.31±0.42 | 0.00±0.00 | 0.00±0.00 | 5.34±0.34 | 0.00±0.00 | 0.00±0.00 | 5.35±0.08 | 0.00±0.00 |
| Llama-3.1-8B | 80.41±0.57 | 2.42±0.99 | 0.23±0.20 | 3.29±0.17 | 0.00±0.00 | 0.00±0.00 | 12.10±0.13 | 1.55±0.67 |
| Gemma-3-4B | 85.87±0.62 | 2.32±0.05 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 0.00±0.00 | 11.78±0.58 | 0.03±0.05 |
| InternLM3-8B | 86.79±0.44 | 2.83±3.54 | 0.00±0.00 | 0.00±0.00 | 0.92±0.07 | 0.00±0.00 | 3.69±0.39 | 5.78±3.04 |

Table 21 provides the values plotted in Figure 3c.

### G.5 LLAMA HARD MIXED RESULTS

Table 22: Results on Hard Mixed with Llama-3.1-8B.

| Suite | Retention | Flat Router | Question-Only | Answer-Only | ARC | Oracle |
|---|---|---|---|---|---|---|
| Hard Mixed | .3406 | .3488±.0072 | .3420±.0024 | .3532±.0046 | .3719±.0026 | .7679 |

## H QUALITY-COST CALIBRATION

We calibrate Qwen retention thresholds separately for each router and seed. The decision variable is the score of the best available non-NoSkill action minus the NoSkill score. We execute that repair only when the difference exceeds the threshold. Calibration changes retention decisions while keeping the router parameters, action ranking, and $\lambda = 0$ configuration fixed. Candidate thresholds are the decision-variable values observed on the development set. We select the threshold that minimizes the absolute difference between realized development calls and the target (0.05, 0.10, 0.15, or 0.20 calls per example). Ties favor fewer calls and then the larger threshold. Calibration does not use development task scores. All selected thresholds fall within the 0.01-call feasibility tolerance and are frozen before ID evaluation.

### H.1 DETERMINISTIC ROUTER COMPARISON

The ARC and Question-Only curves in Figure 2 and the results in Table 23 use the deterministic checkpoints from Appendix F.2. We evaluate each checkpoint on the same 1,006 development examples and 2,510 ID examples, with the original frozen action matrices. Evaluation uses full precision and batches of 16. Table 24 lists each selected threshold and its realized development and ID costs. Means and sample standard deviations use three seeds; the two identical repetitions at seed 42 contribute only once. Figure 2 also includes the original Flat Router and fixed Selective Task-Type results from Table 25, using the same action matrices, development targets, and calibration rule.

Table 23: Development-calibrated quality and cost for deterministic Qwen routers.

| Dev. target | Method | ID score | ID calls | Request time (ms) |
|---|---|---|---|---|
| 0.05 | ARC | $0.6515 \pm 0.0010$ | $0.0118 \pm 0.0020$ | $18.4 \pm 3.7$ |
| 0.05 | Question-Only | $0.6525 \pm 0.0009$ | $0.0301 \pm 0.0053$ | $48.8 \pm 14.8$ |
| 0.10 | ARC | $0.6603 \pm 0.0006$ | $0.0449 \pm 0.0046$ | $87.1 \pm 10.6$ |
| 0.10 | Question-Only | $0.6598 \pm 0.0008$ | $0.0757 \pm 0.0029$ | $158.7 \pm 6.6$ |
| 0.15 | ARC | $0.6725 \pm 0.0012$ | $0.0887 \pm 0.0041$ | $187.0 \pm 6.3$ |
| 0.15 | Question-Only | $0.6706 \pm 0.0012$ | $0.1267 \pm 0.0048$ | $274.1 \pm 9.9$ |
| 0.20 | ARC | $0.6881 \pm 0.0008$ | $0.1416 \pm 0.0069$ | $316.1 \pm 13.8$ |
| 0.20 | Question-Only | $0.6845 \pm 0.0017$ | $0.1829 \pm 0.0022$ | $401.4 \pm 11.4$ |

Values are mean ± sample standard deviation over seeds 42–44. Targets specify development calls per example; ID costs are realized at the frozen thresholds. Request time covers recorded repair LLM requests.

Table 24: Thresholds and per-seed ID results for deterministic Qwen routers.

| Method | Seed | Target | $\tau$ | Dev. calls | ID score | ID calls | Time (ms) |
|---|---|---|---|---|---|---|---|
| ARC | 42 | 0.05 | 0.746166 | 0.0497 | 0.6506 | 0.0096 | 14.7 |
| ARC | 42 | 0.10 | 0.293573 | 0.1004 | 0.6602 | 0.0426 | 79.5 |
| ARC | 42 | 0.15 | 0.175035 | 0.1501 | 0.6722 | 0.0876 | 186.9 |
| ARC | 42 | 0.20 | 0.059139 | 0.1998 | 0.6874 | 0.1390 | 310.0 |
| ARC | 43 | 0.05 | 0.906239 | 0.0497 | 0.6514 | 0.0124 | 18.4 |
| ARC | 43 | 0.10 | 0.520108 | 0.1004 | 0.6598 | 0.0418 | 82.5 |
| ARC | 43 | 0.15 | 0.331411 | 0.1491 | 0.6714 | 0.0853 | 180.8 |
| ARC | 43 | 0.20 | 0.177961 | 0.1998 | 0.6878 | 0.1363 | 306.4 |
| ARC | 44 | 0.05 | 0.749894 | 0.0497 | 0.6526 | 0.0135 | 22.1 |
| ARC | 44 | 0.10 | 0.318334 | 0.0994 | 0.6610 | 0.0502 | 99.1 |
| ARC | 44 | 0.15 | 0.205028 | 0.1501 | 0.6738 | 0.0932 | 193.4 |
| ARC | 44 | 0.20 | 0.105410 | 0.1998 | 0.6890 | 0.1494 | 331.9 |
| Question-Only | 42 | 0.05 | 0.556767 | 0.0497 | 0.6514 | 0.0243 | 33.2 |
| Question-Only | 42 | 0.10 | 0.287173 | 0.1004 | 0.6598 | 0.0789 | 166.1 |
| Question-Only | 42 | 0.15 | 0.136242 | 0.1491 | 0.6718 | 0.1299 | 284.3 |
| Question-Only | 42 | 0.20 | 0.030424 | 0.1998 | 0.6850 | 0.1825 | 408.6 |
| Question-Only | 43 | 0.05 | 0.504066 | 0.0497 | 0.6530 | 0.0315 | 50.5 |
| Question-Only | 43 | 0.10 | 0.355291 | 0.0994 | 0.6606 | 0.0749 | 153.3 |
| Question-Only | 43 | 0.15 | 0.233307 | 0.1491 | 0.6706 | 0.1291 | 273.5 |
| Question-Only | 43 | 0.20 | 0.138035 | 0.2008 | 0.6826 | 0.1809 | 388.2 |
| Question-Only | 44 | 0.05 | 0.508224 | 0.0497 | 0.6530 | 0.0347 | 62.8 |
| Question-Only | 44 | 0.10 | 0.351633 | 0.1004 | 0.6590 | 0.0733 | 156.8 |
| Question-Only | 44 | 0.15 | 0.199050 | 0.1501 | 0.6694 | 0.1211 | 264.5 |
| Question-Only | 44 | 0.20 | 0.063171 | 0.1998 | 0.6858 | 0.1853 | 407.5 |

### H.2 Original Multi-Policy Comparison

The original comparison evaluates ARC, Question-Only, Flat Router, and Selective Task-Type with the same threshold-selection rule. Selective Task-Type applies a threshold to the family-based priority score in Appendix E.1. Figure 4 also includes Family-Aware When-Only, which assigns repairs using true family labels (Appendix I.2.2). These original router runs are separate from the deterministic comparison above.

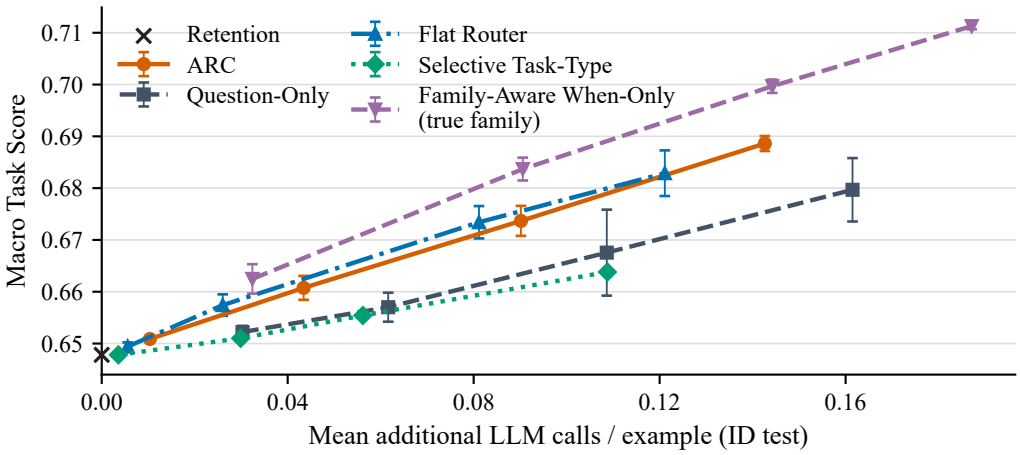



Figure 4: Original Qwen multi-policy quality-cost comparison. Points show mean ID scores and calls; error bars show score standard deviation over three seeds. Family-Aware When-Only uses true family labels. Selective Task-Type and Retention are deterministic.

Table 25: Qwen3-8B quality-cost sweep. Each cell reports mean ID calls / Macro Task Score; columns are development call targets.

| Method | Target 0.05 | Target 0.10 | Target 0.15 | Target 0.20 |
|---|---|---|---|---|
| ARC | 0.0104 / 0.6509 | 0.0434 / 0.6607 | 0.0902 / 0.6737 | 0.1426 / 0.6886 |
| Question-Only | 0.0303 / 0.6522 | 0.0616 / 0.6570 | 0.1086 / 0.6675 | 0.1615 / 0.6797 |
| Flat Router | 0.0056 / 0.6494 | 0.0260 / 0.6574 | 0.0811 / 0.6734 | 0.1211 / 0.6829 |
| Selective Task-Type | 0.0036 / 0.6478 | 0.0299 / 0.6510 | 0.0562 / 0.6554 | 0.1088 / 0.6638 |

Appendix H.4 gives the development thresholds and per-seed results for the original policies. Their original operating points, including both always-on policies, appear in Table 3.

### H.3 Paired Source-Group-Weighted Comparisons

Table 26 reports all paired comparisons with Question-Only, Selective Task-Type, and Flat Router using the source-group weighting in Appendix F.1. Each comparison uses the thresholds selected for the same development call target; the corresponding realized test costs appear in Table 25.

Table 26: Qwen3-8B budget sweep: ARC-minus-baseline differences with source-group-weighted 95% intervals.

| Target | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|
| ARC minus Question-Only | | | |
| 0.05 | $-0.0008$ $[-0.0036, 0.0020]$ | $-0.0028$ $[-0.0056, 0.0000]$ | $-0.0004$ $[-0.0032, 0.0024]$ |
| 0.10 | $+0.0064$ $[0.0012, 0.0116]$ | $-0.0004$ $[-0.0056, 0.0044]$ | $+0.0052$ $[0.0004, 0.0100]$ |
| 0.15 | $+0.0080$ $[0.0012, 0.0148]$ | $-0.0032$ $[-0.0092, 0.0028]$ | $+0.0136$ $[0.0076, 0.0200]$ |
| 0.20 | $+0.0064$ $[-0.0004, 0.0136]$ | $+0.0040$ $[-0.0020, 0.0104]$ | $+0.0164$ $[0.0088, 0.0240]$ |
| ARC minus Flat Router | | | |
| 0.05 | $+0.0012$ $[-0.0008, 0.0032]$ | $+0.0012$ $[-0.0008, 0.0032]$ | $+0.0020$ $[-0.0004, 0.0044]$ |
| 0.10 | $+0.0048$ $[0.0012, 0.0084]$ | $+0.0008$ $[-0.0020, 0.0036]$ | $+0.0044$ $[0.0008, 0.0080]$ |
| 0.15 | $+0.0020$ $[-0.0028, 0.0068]$ | $-0.0036$ $[-0.0080, 0.0008]$ | $+0.0024$ $[-0.0028, 0.0076]$ |
| 0.20 | $+0.0024$ $[-0.0020, 0.0068]$ | $+0.0044$ $[-0.0008, 0.0096]$ | $+0.0104$ $[0.0052, 0.0160]$ |
| ARC minus Selective Task-Type | | | |
| 0.05 | $+0.0036$ $[0.0012, 0.0064]$ | $+0.0028$ $[0.0008, 0.0052]$ | $+0.0028$ $[0.0008, 0.0052]$ |
| 0.10 | $+0.0124$ $[0.0076, 0.0176]$ | $+0.0084$ $[0.0040, 0.0132]$ | $+0.0084$ $[0.0040, 0.0128]$ |
| 0.15 | $+0.0216$ $[0.0148, 0.0284]$ | $+0.0164$ $[0.0104, 0.0228]$ | $+0.0168$ $[0.0104, 0.0236]$ |
| 0.20 | $+0.0260$ $[0.0184, 0.0340]$ | $+0.0232$ $[0.0160, 0.0308]$ | $+0.0252$ $[0.0176, 0.0332]$ |

### H.4 Per-Seed Calibration and Test Results

Tables 27 and 28 report the development thresholds and ID results for all original policies, seeds, and call targets. Calls, request time, and token counts are incremental costs averaged over all examples; intervention rate is the fraction receiving a repair. Seed 0 identifies the fixed Selective Task-Type policy. Budget error is the absolute difference between realized development calls and the target.

Table 27: Development thresholds and realized calibration costs by policy and seed.

| Seed | Target | Threshold | Dev calls | Intervention rate | Budget error |
|---|---|---|---|---|---|
| ARC | | | | | |
| 42 | 0.05 | 0.914339 | 0.049702 | 0.049702 | 0.000298 |
| 42 | 0.10 | 0.312011 | 0.099404 | 0.095427 | 0.000596 |
| 42 | 0.15 | 0.151796 | 0.149105 | 0.122266 | 0.000895 |
| 42 | 0.20 | 0.005082 | 0.200795 | 0.150099 | 0.000795 |
| 43 | 0.05 | 0.871865 | 0.049702 | 0.049702 | 0.000298 |
| 43 | 0.10 | 0.384711 | 0.099404 | 0.097416 | 0.000596 |
| 43 | 0.15 | 0.168603 | 0.149105 | 0.126243 | 0.000895 |
| 43 | 0.20 | 0.050366 | 0.199801 | 0.153082 | 0.000199 |
| 44 | 0.05 | 0.892860 | 0.049702 | 0.049702 | 0.000298 |
| 44 | 0.10 | 0.475771 | 0.099404 | 0.097416 | 0.000596 |
| 44 | 0.15 | 0.259540 | 0.149105 | 0.126243 | 0.000895 |
| 44 | 0.20 | 0.111026 | 0.200795 | 0.153082 | 0.000795 |
| Question-Only | | | | | |
| 42 | 0.05 | 0.487659 | 0.049702 | 0.049702 | 0.000298 |
| 42 | 0.10 | 0.292011 | 0.100398 | 0.094433 | 0.000398 |
| 42 | 0.15 | 0.128022 | 0.150099 | 0.128231 | 0.000099 |
| 42 | 0.20 | 0.006237 | 0.200795 | 0.160040 | 0.000795 |
| 43 | 0.05 | 0.451597 | 0.049702 | 0.048708 | 0.000298 |
| 43 | 0.10 | 0.292002 | 0.099404 | 0.090457 | 0.000596 |
| 43 | 0.15 | 0.137074 | 0.150099 | 0.125249 | 0.000099 |
| 43 | 0.20 | 0.039944 | 0.200795 | 0.157058 | 0.000795 |
| 44 | 0.05 | 0.477449 | 0.049702 | 0.050696 | 0.000298 |
| 44 | 0.10 | 0.316954 | 0.100398 | 0.101392 | 0.000398 |
| 44 | 0.15 | 0.006888 | 0.150099 | 0.151093 | 0.000099 |
| 44 | 0.20 | -0.525263 | 0.200795 | 0.194831 | 0.000795 |
| Flat Router | | | | | |
| 42 | 0.05 | 0.899757 | 0.049702 | 0.049702 | 0.000298 |
| 42 | 0.10 | 0.266423 | 0.100398 | 0.100398 | 0.000398 |
| 42 | 0.15 | -0.518437 | 0.149105 | 0.139165 | 0.000895 |
| 42 | 0.20 | -0.580065 | 0.200795 | 0.166004 | 0.000795 |
| 43 | 0.05 | 0.676267 | 0.049702 | 0.049702 | 0.000298 |
| 43 | 0.10 | 0.059562 | 0.100398 | 0.100398 | 0.000398 |
| 43 | 0.15 | -0.638881 | 0.150099 | 0.143141 | 0.000099 |
| 43 | 0.20 | -0.673644 | 0.199801 | 0.168986 | 0.000199 |
| 44 | 0.05 | 0.782037 | 0.049702 | 0.049702 | 0.000298 |
| 44 | 0.10 | 0.411093 | 0.100398 | 0.101392 | 0.000398 |
| 44 | 0.15 | -0.374256 | 0.150099 | 0.150099 | 0.000099 |
| 44 | 0.20 | -0.457626 | 0.200795 | 0.177932 | 0.000795 |
| Selective Task-Type | | | | | |
| 0 | 0.05 | 0.317338 | 0.049702 | 0.049702 | 0.000298 |
| 0 | 0.10 | 0.298544 | 0.100398 | 0.101392 | 0.000398 |
| 0 | 0.15 | 0.258874 | 0.150099 | 0.151093 | 0.000099 |
| 0 | 0.20 | 0.218049 | 0.200795 | 0.189861 | 0.000795 |

Table 28: ID scores and incremental repair costs by policy, seed, and development call target.

| Seed | Target | ID calls | Macro | Intervention rate | Request time (ms) | Prompt tokens | Output tokens |
|---|---|---|---|---|---|---|---|
| ARC | | | | | | | |
| 42 | 0.05 | 0.010359 | 0.651416 | 0.0104 | 15.4056 | 5.1040 | 0.7586 |
| 42 | 0.10 | 0.050199 | 0.663416 | 0.0478 | 90.9931 | 30.9450 | 4.3355 |
| 42 | 0.15 | 0.093227 | 0.677016 | 0.0733 | 198.1079 | 70.5562 | 8.8952 |
| 42 | 0.20 | 0.145817 | 0.689816 | 0.1020 | 328.0943 | 122.2080 | 14.4677 |
| 43 | 0.05 | 0.010359 | 0.650616 | 0.0104 | 15.7026 | 4.9936 | 0.7705 |
| 43 | 0.10 | 0.042231 | 0.659416 | 0.0406 | 86.5721 | 25.7048 | 4.1275 |
| 43 | 0.15 | 0.088845 | 0.671816 | 0.0693 | 182.7119 | 65.2175 | 8.1697 |
| 43 | 0.20 | 0.137052 | 0.687016 | 0.0968 | 305.4216 | 110.6510 | 13.4964 |
| 44 | 0.05 | 0.010359 | 0.650616 | 0.0104 | 15.4706 | 5.0363 | 0.7625 |
| 44 | 0.10 | 0.037849 | 0.659416 | 0.0367 | 70.0994 | 22.8633 | 3.3745 |
| 44 | 0.15 | 0.088446 | 0.672216 | 0.0673 | 184.6273 | 66.7195 | 8.2092 |
| 44 | 0.20 | 0.145020 | 0.689016 | 0.1004 | 321.1912 | 120.4394 | 14.1988 |
| Question-Only | | | | | | | |
| 42 | 0.05 | 0.0295 | 0.6522 | 0.0287 | 43.8347 | 13.5837 | 2.1255 |
| 42 | 0.10 | 0.0653 | 0.6570 | 0.0578 | 122.0147 | 35.8351 | 5.7139 |
| 42 | 0.15 | 0.1147 | 0.6690 | 0.0940 | 238.3967 | 71.5335 | 10.5980 |
| 42 | 0.20 | 0.1745 | 0.6834 | 0.1307 | 373.6823 | 122.8761 | 16.5311 |
| 43 | 0.05 | 0.0347 | 0.6534 | 0.0307 | 65.2086 | 18.8112 | 3.0944 |
| 43 | 0.10 | 0.0733 | 0.6598 | 0.0614 | 143.5091 | 43.4857 | 6.2673 |
| 43 | 0.15 | 0.1335 | 0.6750 | 0.1020 | 263.7370 | 89.9737 | 11.5625 |
| 43 | 0.20 | 0.1717 | 0.6830 | 0.1255 | 341.3894 | 123.3789 | 14.8183 |
| 44 | 0.05 | 0.0267 | 0.6510 | 0.0267 | 36.0362 | 11.0777 | 1.7514 |
| 44 | 0.10 | 0.0462 | 0.6542 | 0.0462 | 63.6528 | 19.6159 | 3.0578 |
| 44 | 0.15 | 0.0777 | 0.6586 | 0.0777 | 105.3513 | 34.7207 | 5.1446 |
| 44 | 0.20 | 0.1382 | 0.6726 | 0.1283 | 261.5176 | 75.4578 | 12.0155 |
| Flat Router | | | | | | | |
| 42 | 0.05 | 0.0072 | 0.6502 | 0.0076 | 11.2532 | 3.4112 | 0.5661 |
| 42 | 0.10 | 0.0267 | 0.6586 | 0.0283 | 43.7224 | 14.1398 | 2.1614 |
| 42 | 0.15 | 0.0821 | 0.6750 | 0.0769 | 137.2925 | 51.1781 | 6.4143 |
| 42 | 0.20 | 0.1291 | 0.6874 | 0.1016 | 246.1509 | 94.9745 | 10.8263 |
| 43 | 0.05 | 0.0056 | 0.6494 | 0.0060 | 9.3781 | 2.6988 | 0.4713 |
| 43 | 0.10 | 0.0291 | 0.6586 | 0.0311 | 46.8019 | 15.0303 | 2.2861 |
| 43 | 0.15 | 0.0904 | 0.6754 | 0.0845 | 156.9586 | 55.1892 | 7.3064 |
| 43 | 0.20 | 0.1231 | 0.6826 | 0.1024 | 227.3482 | 85.0737 | 10.2299 |
| 44 | 0.05 | 0.0040 | 0.6486 | 0.0044 | 6.9168 | 1.8705 | 0.3446 |
| 44 | 0.10 | 0.0223 | 0.6550 | 0.0235 | 36.9650 | 11.2645 | 1.8219 |
| 44 | 0.15 | 0.0709 | 0.6698 | 0.0757 | 107.4297 | 37.2430 | 5.2267 |
| 44 | 0.20 | 0.1112 | 0.6786 | 0.0984 | 196.5918 | 71.7355 | 8.9622 |
| Selective Task-Type | | | | | | | |
| 0 | 0.05 | 0.0036 | 0.6478 | 0.0036 | 5.0388 | 1.3837 | 0.2414 |
| 0 | 0.10 | 0.0299 | 0.6510 | 0.0299 | 40.6350 | 12.2964 | 1.9665 |
| 0 | 0.15 | 0.0562 | 0.6554 | 0.0562 | 76.8616 | 24.1980 | 3.7116 |
| 0 | 0.20 | 0.1088 | 0.6638 | 0.0952 | 209.9851 | 61.5458 | 9.8629 |

# I Repair Selection Diagnostics

## I.1 Family-Mode Challenge

For each base model and benchmark family, Family-Mode selects the action with the highest mean development reward from the available actions, including NoSkill. When an action is unavailable for a development example, its evaluation uses the retention outcome. Ties favor fewer mean calls, then lower mean latency, and finally the fixed action order. Table 29 lists the selected actions.

Table 29: Family-Mode actions selected on the development set.

| Family | Qwen | Llama | Gemma | InternLM |
|---|---|---|---|---|
| Logic | EvidenceRetrieval | Datalog | NoSkill | SelfRefine |
| Temporal | Temporal | Temporal | Temporal | Temporal |
| Math | SelfRefine | NoSkill | SelfRefine | PythonMath |
| Spatial | SpatialGraph | SpatialGraph | SpatialGraph | SpatialGraph |
| Evidence | EvidenceRetrieval | NoSkill | EvidenceRetrieval | NoSkill |

The Family-Mode Challenge includes examples with at least two available non-NoSkill repairs, at least one available positive-gain repair, and a fixed family action that is available but outside the oracle-best action set. Test rewards determine only subset membership and diagnostic metrics. Family actions are selected on development data; the challenge does not affect training, checkpoint selection, or budget calibration.

Table 30: Comparison with Family-Mode on the challenge subset. Coverage measures oracle-best hits on all repairable ID examples.

| Base LM | $n$ (% ID) | Coverage | ARC Macro | Fixed Macro | $\Delta$ Macro |
|---|---|---|---|---|---|
| Qwen3-8B | 158 (6.29%) | 71.17% | 0.0265 | 0.0265 | 0.0000 |
| Llama-3.1-8B | 196 (7.81%) | 71.55% | 0.0403 | 0.0131 | +0.0272 |
| Gemma-3-4B | 172 (6.85%) | 66.99% | 0.0136 | 0.0152 | −0.0015 |
| InternLM3-8B | 113 (4.50%) | 77.67% | 0.1400 | 0.0252 | +0.1148 |

Coverage in Table 30 uses all repairable examples as its denominator. In the per-seed tables below, Macro averages sample scores within each dataset and then weights datasets equally; Reward averages selected rewards over examples. Positive-gain selection, best-action hit, retention (Keep), and intervention (Repair) rates use the challenge subset as their denominator. Regret is the mean difference between the oracle-best and selected rewards. Table 30 reports averages over the three router seeds; fixed policies share one result across seeds.

Paired comparisons against Question-Only, Answer-Only, Flat Router, and Family-Mode use the source-group weighting defined in Appendix F.1.

Table 31: Results on the Family-Mode Challenge with Qwen3-8B for each seed.

| Method | Seed | Macro | Reward | Positive gain | Best hit | Regret | Keep | Repair |
|---|---|---|---|---|---|---|---|---|
| ARC | 42 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9873 | 0.0127 |
| ARC | 43 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9557 | 0.0443 |
| ARC | 44 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9620 | 0.0380 |
| Question-Only | 42 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Question-Only | 43 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Question-Only | 44 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Answer-Only | 42 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Answer-Only | 43 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Answer-Only | 44 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9430 | 0.0570 |
| Flat Router | 42 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9937 | 0.0063 |
| Flat Router | 43 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9873 | 0.0127 |
| Flat Router | 44 | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.9873 | 0.0127 |
| Family-Mode | – | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 0.0000 | 1.0000 |
| Retention | – | 0.0265 | 0.0185 | 0.0000 | 0.0000 | 0.9654 | 1.0000 | 0.0000 |
| Oracle | – | 0.9769 | 0.9839 | 1.0000 | 1.0000 | 0.0000 | 0.0000 | 1.0000 |

Table 32: Paired comparisons on the Qwen3-8B Family-Mode Challenge. Cells show ARC-minus-baseline differences with source-group-weighted 95% intervals.

| Baseline | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|
| Question-Only | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ |
| Answer-Only | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ |
| Flat Router | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ |
| Family-Mode | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ |

Table 33: Results on the Family-Mode Challenge with Llama-3.1-8B for each seed.

| Method | Seed | Macro | Reward | Positive gain | Best hit | Regret | Keep | Repair |
|---|---|---|---|---|---|---|---|---|
| ARC | 42 | 0.0451 | 0.0668 | 0.0561 | 0.0561 | 0.9068 | 0.8418 | 0.1582 |
| ARC | 43 | 0.0327 | 0.0566 | 0.0459 | 0.0459 | 0.9170 | 0.8827 | 0.1173 |
| ARC | 44 | 0.0432 | 0.0719 | 0.0612 | 0.0612 | 0.9017 | 0.8265 | 0.1735 |
| Question-Only | 42 | 0.0131 | 0.0107 | 0.0000 | 0.0000 | 0.9629 | 0.9592 | 0.0408 |
| Question-Only | 43 | 0.0131 | 0.0107 | 0.0000 | 0.0000 | 0.9629 | 0.9541 | 0.0459 |
| Question-Only | 44 | 0.0218 | 0.0311 | 0.0204 | 0.0204 | 0.9425 | 0.9337 | 0.0663 |
| Answer-Only | 42 | 0.0429 | 0.0617 | 0.0510 | 0.0510 | 0.9119 | 0.8878 | 0.1122 |
| Answer-Only | 43 | 0.0305 | 0.0515 | 0.0408 | 0.0408 | 0.9221 | 0.8878 | 0.1122 |
| Answer-Only | 44 | 0.0283 | 0.0464 | 0.0357 | 0.0357 | 0.9272 | 0.8980 | 0.1020 |
| Flat Router | 42 | 0.0516 | 0.0821 | 0.0714 | 0.0714 | 0.8915 | 0.8469 | 0.1531 |
| Flat Router | 43 | 0.0153 | 0.0158 | 0.0051 | 0.0051 | 0.9578 | 0.9694 | 0.0306 |
| Flat Router | 44 | 0.0432 | 0.0719 | 0.0612 | 0.0612 | 0.9017 | 0.8571 | 0.1429 |
| Family-Mode | – | 0.0131 | 0.0107 | 0.0000 | 0.0000 | 0.9629 | 0.3571 | 0.6429 |
| Retention | – | 0.0131 | 0.0107 | 0.0000 | 0.0000 | 0.9629 | 1.0000 | 0.0000 |
| Oracle | – | 0.9677 | 0.9736 | 1.0000 | 1.0000 | 0.0000 | 0.0000 | 1.0000 |

Table 34: Paired comparisons on the Llama-3.1-8B Family-Mode Challenge. Cells show ARC-minus-baseline differences with source-group-weighted 95% intervals.

| Baseline | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|
| Question-Only | $+0.0304\ [0.0107, 0.0549]$ | $+0.0204\ [0.0072, 0.0368]$ | $+0.0200\ [0.0005, 0.0421]$ |
| Answer-Only | $+0.0012\ [0.0000, 0.0036]$ | $+0.0030\ [-0.0024, 0.0101]$ | $+0.0144\ [0.0024, 0.0301]$ |
| Flat Router | $-0.0116\ [-0.0304, 0.0000]$ | $+0.0183\ [0.0069, 0.0312]$ | $-0.0009\ [-0.0063, 0.0036]$ |
| Family-Mode | $+0.0304\ [0.0107, 0.0549]$ | $+0.0204\ [0.0072, 0.0368]$ | $+0.0290\ [0.0120, 0.0498]$ |

Table 35: Results on the Family-Mode Challenge with Gemma-3-4B for each seed.

| Method | Seed | Macro | Reward | Positive gain | Best hit | Regret | Keep | Repair |
|---|---|---|---|---|---|---|---|---|
| ARC | 42 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9360 | 0.0640 |
| ARC | 43 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9419 | 0.0581 |
| ARC | 44 | 0.0149 | 0.0213 | 0.0058 | 0.0058 | 0.9325 | 0.9244 | 0.0756 |
| Question-Only | 42 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9651 | 0.0349 |
| Question-Only | 43 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9651 | 0.0349 |
| Question-Only | 44 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9709 | 0.0291 |
| Answer-Only | 42 | 0.0187 | 0.0329 | 0.0174 | 0.0174 | 0.9209 | 0.9128 | 0.0872 |
| Answer-Only | 43 | 0.0187 | 0.0329 | 0.0174 | 0.0174 | 0.9209 | 0.9128 | 0.0872 |
| Answer-Only | 44 | 0.0187 | 0.0329 | 0.0174 | 0.0174 | 0.9209 | 0.8837 | 0.1163 |
| Flat Router | 42 | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 0.9419 | 0.0581 |
| Flat Router | 43 | 0.0187 | 0.0329 | 0.0174 | 0.0174 | 0.9209 | 0.9128 | 0.0872 |
| Flat Router | 44 | 0.0187 | 0.0329 | 0.0174 | 0.0174 | 0.9209 | 0.9186 | 0.0814 |
| Family-Mode | – | 0.0152 | 0.0181 | 0.0058 | 0.0000 | 0.9357 | 0.6163 | 0.3837 |
| Retention | – | 0.0130 | 0.0155 | 0.0000 | 0.0000 | 0.9383 | 1.0000 | 0.0000 |
| Oracle | – | 0.9612 | 0.9538 | 1.0000 | 1.0000 | 0.0000 | 0.0000 | 1.0000 |

Table 36: Paired comparisons on the Gemma-3-4B Family-Mode Challenge. Cells show ARC-minus-baseline differences with source-group-weighted 95% intervals.

| Baseline | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|
| Question-Only | $0.0000\ [0.0000, 0.0000]$ | $0.0000\ [0.0000, 0.0000]$ | $+0.0028\ [0.0000, 0.0083]$ |
| Answer-Only | $-0.0058\ [-0.0131, 0.0000]$ | $-0.0058\ [-0.0131, 0.0000]$ | $-0.0031\ [-0.0078, 0.0000]$ |
| Flat Router | $0.0000\ [0.0000, 0.0000]$ | $-0.0052\ [-0.0119, 0.0000]$ | $-0.0024\ [-0.0107, 0.0060]$ |
| Family-Mode | $-0.0022\ [-0.0065, 0.0000]$ | $-0.0022\ [-0.0065, 0.0000]$ | $+0.0006\ [-0.0065, 0.0083]$ |

Table 37: Results on the Family-Mode Challenge with InternLM3-8B for each seed.

| Method | Seed | Macro | Reward | Positive gain | Best hit | Regret | Keep | Repair |
|---|---|---|---|---|---|---|---|---|
| ARC | 42 | 0.0763 | 0.0887 | 0.0619 | 0.0619 | 0.8444 | 0.6106 | 0.3894 |
| ARC | 43 | 0.1661 | 0.2834 | 0.2566 | 0.2566 | 0.6497 | 0.6106 | 0.3894 |
| ARC | 44 | 0.1774 | 0.2834 | 0.2566 | 0.2566 | 0.6497 | 0.6195 | 0.3805 |
| Question-Only | 42 | 0.0252 | 0.0267 | 0.0000 | 0.0000 | 0.9063 | 0.9912 | 0.0088 |
| Question-Only | 43 | 0.0252 | 0.0267 | 0.0000 | 0.0000 | 0.9063 | 0.8496 | 0.1504 |
| Question-Only | 44 | 0.0293 | 0.0356 | 0.0088 | 0.0088 | 0.8975 | 0.7876 | 0.2124 |
| Answer-Only | 42 | 0.0559 | 0.0444 | 0.0177 | 0.0177 | 0.8886 | 0.7788 | 0.2212 |
| Answer-Only | 43 | 0.0406 | 0.0356 | 0.0088 | 0.0088 | 0.8975 | 0.8496 | 0.1504 |
| Answer-Only | 44 | 0.0569 | 0.0710 | 0.0442 | 0.0442 | 0.8621 | 0.8319 | 0.1681 |
| Flat Router | 42 | 0.1272 | 0.2480 | 0.2212 | 0.2212 | 0.6851 | 0.6283 | 0.3717 |
| Flat Router | 43 | 0.1385 | 0.2480 | 0.2212 | 0.2212 | 0.6851 | 0.6549 | 0.3451 |
| Flat Router | 44 | 0.0406 | 0.0356 | 0.0088 | 0.0088 | 0.8975 | 0.6372 | 0.3628 |
| Family-Mode | – | 0.0252 | 0.0267 | 0.0000 | 0.0000 | 0.9063 | 0.2124 | 0.7876 |
| Retention | – | 0.0252 | 0.0267 | 0.0000 | 0.0000 | 0.9063 | 1.0000 | 0.0000 |
| Oracle | – | 0.9369 | 0.9330 | 1.0000 | 1.0000 | 0.0000 | 0.0000 | 1.0000 |

Table 38: Paired comparisons on the InternLM3-8B Family-Mode Challenge. Cells show ARC-minus-baseline differences with source-group-weighted 95% intervals.

| Baseline | Seed 42 | Seed 43 | Seed 44 |
|---|---|---|---|
| Question-Only | $+0.0565\ [0.0154, 0.1075]$ | $+0.1277\ [0.0745, 0.1843]$ | $+0.1371\ [0.0805, 0.1994]$ |
| Answer-Only | $+0.0258\ [0.0030, 0.0530]$ | $+0.1124\ [0.0652, 0.1631]$ | $+0.1096\ [0.0591, 0.1661]$ |
| Flat Router | $-0.0344\ [-0.0848, 0.0211]$ | $+0.0230\ [0.0000, 0.0598]$ | $+0.1262\ [0.0745, 0.1843]$ |
| Family-Mode | $+0.0565\ [0.0154, 0.1075]$ | $+0.1277\ [0.0745, 0.1843]$ | $+0.1416\ [0.0848, 0.2042]$ |

### I.2 Qwen Fixed-Action Controls

Both controls use the true benchmark family to assign one development-selected repair (Table 29). Restricted ARC retains the trained multi-action router; Family-Aware When-Only trains a separate binary retain/repair gate. The fixed repairs are EvidenceRetrieval for Logic and Evidence, SelfRefine for Math, Temporal for Temporal, and SpatialGraph for Spatial. An unavailable repair leads to retention. Table 39 compares the policies at all four development call targets; FA When-Only abbreviates Family-Aware When-Only in tables. Scores and costs follow Section 6; realized test costs can differ at the same development target.

Table 39: Qwen fixed-action controls, averaged over three seeds.

| Dev target | Policy | Macro | Calls | Repair (%) | Time (ms) |
|---|---|---|---|---|---|
| 0.05 | ARC | 0.6509 | 0.0104 | 1.04 | 15.53 |
| 0.05 | Restricted | 0.6510 | 0.0105 | 1.05 | 15.71 |
| 0.05 | FA When-Only | 0.6625 | 0.0324 | 3.55 | 53.30 |
| 0.10 | ARC | 0.6607 | 0.0434 | 4.17 | 82.55 |
| 0.10 | Restricted | 0.6607 | 0.0434 | 4.17 | 82.55 |
| 0.10 | FA When-Only | 0.6837 | 0.0906 | 8.91 | 201.46 |
| 0.15 | ARC | 0.6737 | 0.0902 | 7.00 | 188.48 |
| 0.15 | Restricted | 0.6743 | 0.0914 | 7.07 | 191.20 |
| 0.15 | FA When-Only | 0.6997 | 0.1442 | 12.47 | 333.20 |
| 0.20 | ARC | 0.6886 | 0.1426 | 9.97 | 318.24 |
| 0.20 | Restricted | 0.6887 | 0.1430 | 9.99 | 319.59 |
| 0.20 | FA When-Only | 0.7113 | 0.1871 | 15.09 | 435.04 |

#### I.2.1 Frozen Restricted ARC

Restricted ARC limits the candidates to NoSkill and the assigned family repair, retaining the original action scores without retraining or renormalization. We calibrate each seed and policy using Appendix H; thresholds and test decisions are fixed before retrieving outcomes.

Restricted ARC and ARC have nearly identical quality and cost. The two policies select identical actions for all 2,510 test examples in seven of twelve seed–budget pairs; they differ on one to four examples in the remaining pairs. Every disagreement concerns whether to retain or repair; when both policies intervene, they select the same repair type. Table 40 gives the Restricted ARC thresholds and test points; the shared ARC reference appears in Tables 27 and 28.

Table 40: Development calibration and per-seed Qwen Restricted ARC results.

| Seed | Target | $\tau$ | Dev calls | Test Macro | Test calls |
|---|---|---|---|---|---|
| 42 | 0.05 | 0.914339 | 0.049702 | 0.651416 | 0.010359 |
| 42 | 0.10 | 0.312011 | 0.099404 | 0.663416 | 0.050199 |
| 42 | 0.15 | 0.148892 | 0.150099 | 0.678216 | 0.095618 |
| 42 | 0.20 | 0.005082 | 0.199801 | 0.689816 | 0.145817 |
| 43 | 0.05 | 0.871865 | 0.049702 | 0.650616 | 0.010359 |
| 43 | 0.10 | 0.382565 | 0.100398 | 0.659416 | 0.042231 |
| 43 | 0.15 | 0.166793 | 0.150099 | 0.671816 | 0.088845 |
| 43 | 0.20 | 0.049313 | 0.200795 | 0.687416 | 0.138645 |
| 44 | 0.05 | 0.877504 | 0.049702 | 0.651016 | 0.010757 |
| 44 | 0.10 | 0.474439 | 0.100398 | 0.659416 | 0.037849 |
| 44 | 0.15 | 0.256008 | 0.150099 | 0.673016 | 0.089641 |
| 44 | 0.20 | 0.111026 | 0.199801 | 0.689016 | 0.144622 |

Table 41 reports paired source-group-weighted intervals with 10,000 resamples and fixed thresholds (Appendix F.1). An interval of $[0, 0]$ indicates identical paired rewards.

Table 41: Paired Qwen differences: ARC minus Restricted ARC.

| Seed | Dev target | Group-weighted $\Delta$ | 95% interval | Action disagreements |
|---|---|---|---|---|
| 42 | 0.05 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 42 | 0.10 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 42 | 0.15 | $-0.0012$ | $[-0.0028, +0.0000]$ | 3 |
| 42 | 0.20 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 43 | 0.05 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 43 | 0.10 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 43 | 0.15 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 43 | 0.20 | $-0.0004$ | $[-0.0012, +0.0000]$ | 4 |
| 44 | 0.05 | $-0.0004$ | $[-0.0012, +0.0000]$ | 1 |
| 44 | 0.10 | $+0.0000$ | $[+0.0000, +0.0000]$ | 0 |
| 44 | 0.15 | $-0.0008$ | $[-0.0020, +0.0000]$ | 4 |
| 44 | 0.20 | $+0.0000$ | $[+0.0000, +0.0000]$ | 1 |

### I.2.2 Family-Aware When-Only

Family-Aware When-Only uses the same public input, numeric features, and pretrained DeBERTa-v3-small encoder as ARC, followed by a $512 \to 256 \to 2$ MLP head that predicts whether to retain the Anchor or execute the assigned repair. The target label is repair when the available fixed action has a higher final-answer reward than retention, and retention otherwise. We train with class-weighted cross-entropy, using the class-weight formula in Appendix D with two classes. The action is assigned using family labels outside the gate; these labels are excluded from encoder features. The remaining training settings follow Appendix D, including seeds 42–44, at most six epochs, and checkpoint selection by development score with an early-stopping patience of two. Each run starts from the pretrained encoder; the ARC and Restricted ARC references use the existing checkpoints evaluated above.

We calibrate a threshold on the difference between the repair and retention logits for the same four development call targets. Table 42 reports the thresholds and test results. Table 43 uses the paired procedure above to compare policies at the same development target; their realized test costs differ.

Table 42: Development calibration and per-seed Qwen Family-Aware When-Only results.

| Seed | Target | $\tau$ | Dev calls | Test Macro | Test calls |
|---|---|---|---|---|---|
| 42 | 0.05 | 1.085951 | 0.049702 | 0.659416 | 0.028685 |
| 42 | 0.10 | 0.622739 | 0.100398 | 0.683816 | 0.087251 |
| 42 | 0.15 | 0.403074 | 0.150099 | 0.698216 | 0.136255 |
| 42 | 0.20 | 0.169589 | 0.199801 | 0.711416 | 0.189641 |
| 43 | 0.05 | 0.901299 | 0.049702 | 0.665016 | 0.037849 |
| 43 | 0.10 | 0.636328 | 0.099404 | 0.681416 | 0.084861 |
| 43 | 0.15 | 0.393121 | 0.149105 | 0.700616 | 0.141833 |
| 43 | 0.20 | 0.240585 | 0.199801 | 0.711816 | 0.181275 |
| 44 | 0.05 | 0.909561 | 0.049702 | 0.663016 | 0.030677 |
| 44 | 0.10 | 0.560863 | 0.099404 | 0.685816 | 0.099602 |
| 44 | 0.15 | 0.303346 | 0.149105 | 0.700216 | 0.154582 |
| 44 | 0.20 | 0.155596 | 0.199801 | 0.710616 | 0.190438 |

Table 43: Paired Qwen differences against Family-Aware When-Only.

| Seed | Target | ARC−FA When-Only Δ [95% interval] | Restricted−FA When-Only Δ [95% interval] |
|---|---|---|---|
| 42 | 0.05 | $-0.0080\,[-0.0120, -0.0040]$ | $-0.0080\,[-0.0120, -0.0040]$ |
| 42 | 0.10 | $-0.0204\,[-0.0272, -0.0140]$ | $-0.0204\,[-0.0272, -0.0140]$ |
| 42 | 0.15 | $-0.0212\,[-0.0292, -0.0136]$ | $-0.0200\,[-0.0280, -0.0124]$ |
| 42 | 0.20 | $-0.0216\,[-0.0292, -0.0140]$ | $-0.0216\,[-0.0292, -0.0140]$ |
| 43 | 0.05 | $-0.0144\,[-0.0196, -0.0096]$ | $-0.0144\,[-0.0196, -0.0096]$ |
| 43 | 0.10 | $-0.0220\,[-0.0284, -0.0156]$ | $-0.0220\,[-0.0284, -0.0156]$ |
| 43 | 0.15 | $-0.0288\,[-0.0364, -0.0208]$ | $-0.0288\,[-0.0364, -0.0208]$ |
| 43 | 0.20 | $-0.0248\,[-0.0320, -0.0176]$ | $-0.0244\,[-0.0316, -0.0172]$ |
| 44 | 0.05 | $-0.0124\,[-0.0168, -0.0080]$ | $-0.0120\,[-0.0164, -0.0080]$ |
| 44 | 0.10 | $-0.0264\,[-0.0336, -0.0196]$ | $-0.0264\,[-0.0336, -0.0196]$ |
| 44 | 0.15 | $-0.0280\,[-0.0360, -0.0200]$ | $-0.0272\,[-0.0352, -0.0196]$ |
| 44 | 0.20 | $-0.0216\,[-0.0288, -0.0144]$ | $-0.0216\,[-0.0288, -0.0144]$ |

### I.3 Within-Dataset Anchor Mismatch

For each target example, we rotate donor Anchors within the same dataset, requiring the donor to come from a different source group. This changes only the answer and reasoning supplied to the router and their length features. The question, availability mask, target Anchor used by repairs, and cached action outcomes remain those of the original example. We score the chosen action using that original action matrix. Each donor is used at most twice.

Across three runs, the mean ID score is .691549 with the true Anchor and .689682 with both answer and reasoning shuffled. The corresponding Hard Mixed scores are .611805 and .608063. For seed 42, the paired source-group-weighted score differences between true and shuffled Anchors are +.0020 $[-.0048, .0092]$ on ID and +.0010 $[-.0102, .0122]$ on Hard Mixed.