When Small Evaluator Biases Become Permanent: Phase Transitions in Self-Evolving Agent Skill Libraries

Anonymous authors
Paper under double-blind review

Abstract

Persistent skill libraries allow large language model agents to reuse and refine executable artifacts. Recursive selection and retrieval, however, can amplify small evaluator preferences into lasting shifts in library composition. Identifying this amplification requires separating it from utility differences, repair asymmetries, retrieval preferences, and finite-population drift while modeling artifact evolution and endogenous exposure. We introduce the *Causal Replicator–Retrieval model* (CRR), a stochastic framework that tracks the quality-weighted frequency of a controlled, task-irrelevant attribute and the resulting cross-group exposure imbalance. CRR integrates quality-conditioned evaluator selection, directional transformation through mutation or repair, and retrieval reinforcement driven by accumulated exposure. Its mean-field dynamics characterize low-bias and biased equilibria, and its local Jacobian determines whether perturbations contract or amplify. We identify the underlying mechanisms using counterfactually matched skill variants, complete lineage and retrieval logs, and randomized channel-level interventions on evaluation, promotion, and repair. A hierarchical Bayesian state-space formulation estimates mechanism parameters and predicts held-out long-run regimes. The posterior informs a minimum stability-margin controller that selects evaluator calibration and exposure de-reinforcement while preserving task utility. We evaluate CRR in synthetic populations and executable-skill sandboxes through parameter recovery, unseen-phase prediction, channel-specific counterfactuals, and bias–utility–cost trade-offs. These evaluations provide a falsifiable framework for diagnosing and controlling evaluator bias in recursively updated skill libraries.

1 Introduction

Large language model agents increasingly improve by accumulating persistent external artifacts alongside parameter updates. Voyager stores executable programs in a retrievable skill library, SkillWeaver discovers and refines reusable APIs through practice, and SAGE reuses previously acquired skills in subsequent rollouts.(Wang et al., 2023a)(Zheng et al., 2025)(Wang et al., 2025) Open-ended self-improvement systems also generate variants, evaluate them, retain selected artifacts, and reuse those artifacts as stepping stones.(Zhang et al., 2026) These operations create a feedback loop in which accepted skills are retrieved, executed, inherited by descendants, and reevaluated. The resulting library forms a dynamically selected population whose composition determines the evidence available for future selection.

This feedback loop can amplify small evaluator preferences. LLM evaluators are sensitive to task-irrelevant presentation, framing, response position, and relationships between generators and evaluators. Controlled LLM-as-a-Judge studies document these effects,(Ye et al., 2024) and preference leakage favors outputs from related models, including models from the same family or inheritance chain.(Li et al., 2025) Prior work has examined the reliability, counterfactual evaluation, and debiasing of individual judgments. Within a recursively updated library, however, a small one-step bias can decay under symmetric repair, remain bounded, or grow when accepted skills receive additional exposure and reproduction opportunities. As illustrated in Figure 1, biased evaluation can shift library composition, increasing retrieval and subsequent reproduction of favored skills. Extreme library

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.

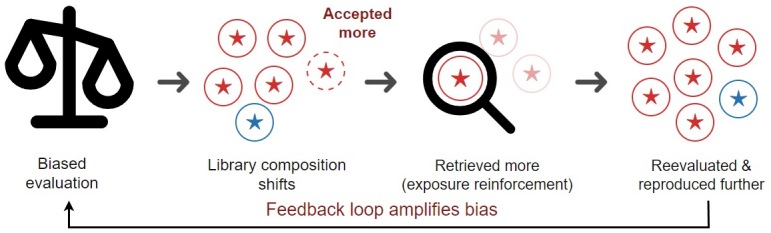


Figure 1: Evaluator bias amplified by recursive retrieval and reproduction.

composition can also arise from utility differences, asymmetric rewriting, retrieval preferences, or finite-population drift. We study when local evaluator bias becomes a persistent state of a recursively updated skill library.

Final attribute proportions cannot distinguish selection from transformation because both mechanisms can produce similar trajectories, a known identification problem in cultural evolution.(Mesoudi, 2021) Historical exposure can also increase retrieval probability, allowing early advantages to generate further observations and selection opportunities. Replicator–mutator dynamics characterize selection, mutation, equilibria, and local stability,(Duong & Han, 2019) but skill libraries additionally involve quality-conditioned evaluation, lineage dependence, and endogenous exposure. Observational trajectories generally cannot separate these channels. A suitable framework must therefore (i) separate evaluator selection, repair-induced conversion, and retrieval reinforcement; (ii) identify these mechanisms while controlling for utility, age, and lineage; and (iii) predict held-out long-run regimes and intervention thresholds.

We introduce the *Causal Replicator–Retrieval model* (CRR), a low-dimensional stochastic model of bias propagation in self-evolving skill libraries. Each skill contains executable content and is associated with a controlled task-irrelevant attribute, hidden-test utility, lineage, age, and exposure history. To isolate nuisance preference from usefulness, the primary analyses admit only counterfactually matched twins whose attribute variants satisfy the same input–output contract and have approximately equal hidden-test utility. CRR tracks the quality-weighted frequency of one attribute and the cross-group exposure imbalance. It couples quality-adjusted evaluator selection, asymmetric mutation or repair, and retrieval probabilities that increase with accumulated exposure. Attribute-dependent acceptance produces additional copies, exposure reinforcement increases their retrieval frequency, and reevaluation can further increase their representation. Reverse mutation, repair, and exposure decay counteract this feedback loop.

The model yields falsifiable predictions. In the two-type mean-field formulation, fixed points characterize low-bias and biased equilibria, and the spectral radius of the local Jacobian determines whether perturbations around the low-bias equilibrium contract or grow. These quantities distinguish decay, biased stationarity, and finite-horizon approximate fixation without interpreting experimental fixation as permanent absorption. Because observed trajectories are compatible with multiple channel-level explanations, we use randomized encouragements in place of unrestricted correlational fitting. Evaluator offsets or label exchanges estimate quality-conditional selection responses; promotion within a fixed top-k set estimates retrieval reinforcement; and attribute-preserving or attribute-swapping repair templates estimate directional transformation. A hierarchical Bayesian state-space model records intervention propensities and accounts for task family, model configuration, generation, and lineage.

Given a posterior distribution over mechanism parameters, minimum stability-margin control selects the least-cost combination of evaluator calibration and exposure de-reinforcement that preserves task utility and keeps the low-bias fixed point locally stable with high posterior probability. We evaluate CRR on controlled synthetic populations and isolated executable-skill sandboxes. We separate estimation runs from hidden phase combinations and freeze each controller before the confirmatory portion of a trajectory. The evaluation measures parameter recovery under known mechanisms, calibrated prediction of unseen phases, counterfactual accuracy when individual channels are disabled, and bias–utility–cost trade-offs against equalized retrieval, fixed debiasing, exposure-free replicator–mutator models, and capacity-matched black-box state-space predictors. The empirical

evidence rejects the framework if local bias does not interact with exposure reinforcement, the mechanisms remain unidentifiable under randomization, or simple budget-matched interventions achieve comparable predictive and control performance.

Our contributions are:

- We formulate evaluator bias as a library-level dynamical process and separate one-step evaluator preference from the long-run effects of recursive selection, inheritance, repair, and retrieval exposure.
- We propose CRR, which connects quality-conditioned selection, directional transformation, and exposure reinforcement to equilibria, local stability, and finite-horizon phase regimes.
- We develop a counterfactual identification strategy based on utility-matched twins, complete lineage and retrieval logs, and randomized channel-level encouragements that estimate selection, repair, and exposure mechanisms separately.
- We derive a minimum stability-margin intervention and an independent closed-loop evaluation protocol for phase-boundary prediction and bias–utility–cost trade-offs, using terminal bias as one component of the evaluation.

2 Related Work

2.1 Persistent Memory and Recursive Improvement in Language-Model Agents

Language-model agents increasingly externalize adaptation into persistent memories, tools, and reusable experience. Tool-augmented systems interleave reasoning with actions or API calls but generally treat tools as fixed.(Yao et al., 2022)(Schick et al., 2023) Memory-based systems instead store and retrieve observations, reflections, or cross-task knowledge.(Park et al., 2023)(Packer et al., 2023)(Zhao et al., 2024) Their behavior depends on policies governing which experiences are retained and reused.

Self-improving approaches persist feedback, revise outputs through self-critique, or retain evaluations to guide search.(Shinn et al., 2023)(Madaan et al., 2023)(Yao et al., 2023)(Zhou et al., 2023) Because supervision may come from rewards, learned evaluators, or the candidate-generating model, retained artifacts can alter later proposals and evidence.

Executable libraries structure adaptation as reusable code or APIs acquired through interaction, autonomous practice, or reinforcement-learning rollouts.(Wang et al., 2023a)(Zheng et al., 2025)(Wang et al., 2025) Other systems evolve prompts, scaffolding programs, or archived agent variants.(Fernando et al., 2023)(Zelikman et al., 2023)(Zhang et al., 2026) Their archives jointly reflect artifact quality, evaluator noise, mutation, parent selection, and retrieval frequency.

2.2 Language Models as Evaluators

LLM evaluators score open-ended outputs through scalar or pairwise judgments, structured criteria, and automated preference comparisons.(Zheng et al., 2023)(Liu et al., 2023)(Dubois et al., 2024) Specialized judges use supervised judgment or rubric-conditioned feedback data.(Zhu et al., 2023)(Kim et al., 2023) These approaches trade off generality, calibration, cost, and dependence on training judgments.

Reliability is assessed through human agreement, controlled perturbations, and adversarial comparisons. Instruction-aware tests reveal failures on superficially plausible outputs,(Zhang et al., 2023) while broader analyses identify sensitivity to candidate order, length, style, framing, and generator identity.(Wang et al., 2023b)(Shi et al., 2024)(Ye et al., 2024) Even small directional preferences can repeatedly affect downstream retention and exposure.

Evaluators may recognize generator-specific regularities and prefer outputs from themselves or related model families.(Panickssery et al., 2024)(Wataoka et al., 2024)(Li et al., 2025) Such dependence can correlate with lineage, stylistic inheritance, or repeated rewriting. Position swapping, repeated comparisons, rubric decomposition, and independent evaluators reduce some biases,(Zheng

et al., 2023)(Kim et al., 2023)(Shi et al., 2024) while length control separates quality from verbosity.(Dubois et al., 2024) However, static calibration does not capture how residual preferences reshape future candidate pools.

2.3 Selection, Transformation, and Endogenous Exposure

Population models describe variants reproducing according to measured fitness. Replicator–mutator dynamics combine selection with type transitions and can produce equilibria, concentration, or error thresholds.(Eigen, 1971)(Duong & Han, 2019) Evolutionary algorithms and population-based training similarly iterate selection, perturbation, and evaluation.(Blickle & Thiele, 1996)(Jaderberg et al., 2017) Their outcomes depend on replacement, mutation, population size, and evaluation noise.

Cultural-evolution models distinguish differential propagation from transformations during transmission.(Mesoudi, 2021) Editing or reconstruction can mimic selection, while asymmetric transformations can sustain skewed distributions or prevent fixation. Separating these mechanisms therefore requires lineage observations or transformation interventions.

Endogenous exposure adds reinforcement: accumulated popularity increases future opportunities,(Barabási & Albert, 1999) and recommendation feedback loops can distort outcomes, reduce diversity, and make relevance estimates depend on prior exposure.(Chaney et al., 2017)(Jiang et al., 2019) Memory retrieval similarly gives surfaced artifacts more opportunities for execution, evaluation, revision, and copying.

Counterfactual risk minimization and off-policy evaluation use propensities to estimate outcomes under alternative exposure policies,(Swaminathan & Joachims, 2015)(Jiang & Li, 2015) while causal recommendation separates relevance from treatment assignment.(Bonner & Vasile, 2018) These methods require overlap and accurate propensities, which become fragile under concentrated retrieval. Selective-label analyses likewise show that chosen-action outcomes cannot identify unobserved performance without assumptions or exploration.(Lakkaraju et al., 2017)

Performative prediction formalizes how deployment changes later training or evaluation distributions.(Perdomo et al., 2020) Stable solutions need not optimize the original distribution, and related strategic-classification and repeated-risk-minimization models characterize decision-induced shifts.(Hardt et al., 2016)(Miller et al., 2021) Selection, transformation, and retrieval can therefore generate similar marginal frequencies despite different underlying mechanisms.

3 Method

We introduce the *Causal Replicator–Retrieval model* (CRR), a lineage-aware stochastic model that characterizes the propagation of evaluator preferences through an evolving skill library. CRR models differential validation, retrieval-driven exposure, and attribute transformation as separate mechanisms. Randomized micro-interventions identify these mechanisms, and the estimated dynamics support regime prediction and minimum-cost stabilization.

3.1 Problem Formulation and CRR Overview

At generation $t \in \{0, \dots, T\}$, the library is $\mathcal{L}_t = \{s_{i,t}\}_{i=1}^{N_t}$, where each skill is represented by $s_{i,t} = (c_{i,t}, z_{i,t}, q_{i,t}, \ell_{i,t}, a_{i,t}, e_{i,t})$. Here, $c_{i,t}$ denotes executable content, $z_{i,t} \in \{-1, +1\}$ is an experimenter-controlled nuisance attribute, $q_{i,t} \in [0, 1]$ is hidden-test utility, $\ell_{i,t}$ denotes lineage, $a_{i,t}$ is age, and $e_{i,t}$ is cumulative exposure. The attribute represents a semantics-preserving framing, formatting choice, or randomized provenance label and is excluded from the utility test.

Counterfactually matched twins share executable behavior and receive opposite attributes. A pair (c_i^+, c_i^-) enters the primary analysis only if

$$|q(c_i^+) - q(c_i^-)| \le \epsilon_q, \tag{1}$$

where $\epsilon_q \ge 0$ is a preregistered tolerance. Twins share a data split, and their descendants retain the base-pair identifier. This design estimates the attribute contrast conditional on measured utility while allowing for behavioral differences that hidden tests do not capture.

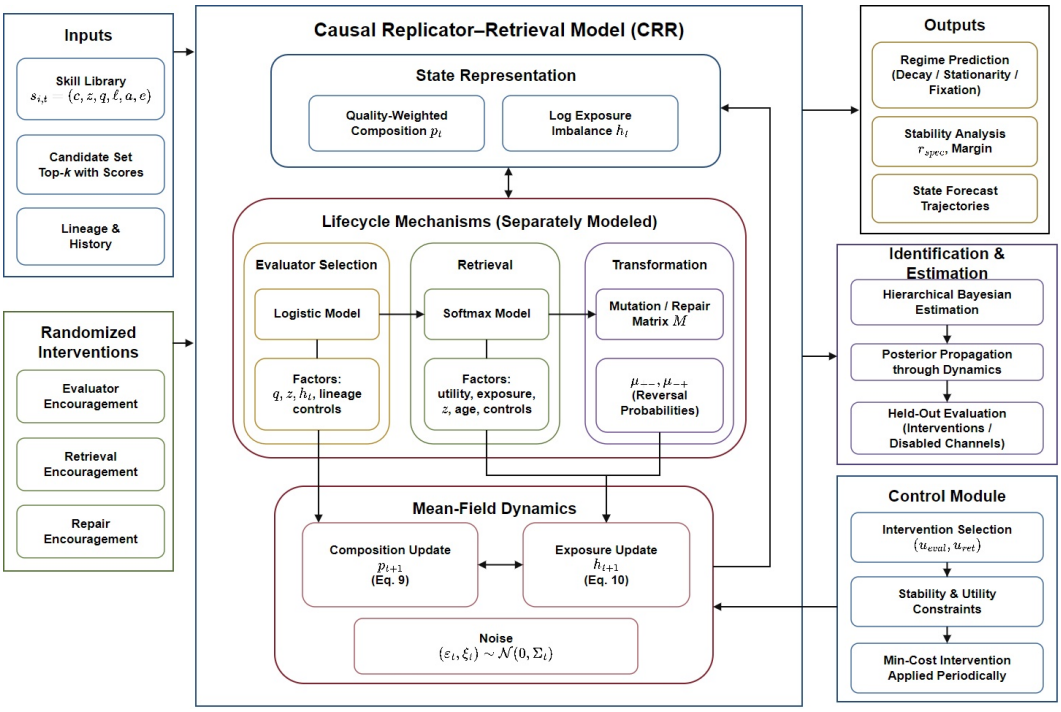


Figure 2: Overview of the CRR framework.

For a fixed nondecreasing utility weight w, the state $\mathbf{s}_t = (p_t, h_t)^\top$ consists of the quality-weighted composition and log exposure imbalance:

$$p_t = \frac{\sum_{i \in \mathcal{L}_t} \mathbb{I}[z_{i,t} = +1] w(q_{i,t})}{\sum_{i \in \mathcal{L}_t} w(q_{i,t})}, \tag{2}$$

$$h_t = \log \frac{E_t^+ + c_0}{E_t^- + c_0}, \qquad E_t^\pm = \sum_{i \in \mathcal{L}_t} \mathbb{I}[z_{i,t} = \pm 1] e_{i,t}, \tag{3}$$

where $c_0 > 0$. Preregistered thresholds classify late trajectories as bias decay, biased stationarity, or approximate fixation. Approximate fixation requires $p_t > \tau_{\text{fix}}$ or $p_t < 1 - \tau_{\text{fix}}$ for W generations. This criterion applies to the finite observation horizon and does not establish permanent absorption.

CRR models a lifecycle comprising retrieval, execution, evaluation, selection, and mutation or repair. As illustrated in Figure 2, randomized interventions identify the separately modeled lifecycle mechanisms, whose estimated dynamics support regime prediction, stability analysis, state forecasting, and minimum-cost control. The resulting feedback structure resembles skill accumulation in Voyager (Wang et al., 2023a) and lineage-preserving archives in the Darwin Gödel Machine (Zhang et al., 2025). CRR focuses on nuisance-dependent dynamics, whereas these systems focus on task improvement.

3.2 Causal Selection–Retrieval Model

Evaluator selection. For acceptance indicator $A_{i,t}$ and pre-evaluation history $\mathcal{H}_{i,t}$,

$$\Pr(A_{i,t} = 1 \mid \mathcal{H}_{i,t}) = \sigma\left(\alpha_t + \beta_q q_{i,t} + \frac{z_{i,t}}{2}(\delta + \eta h_t) + u_{\ell_{i,t}} + \boldsymbol{\beta}^\top \mathbf{x}_{i,t}\right), \tag{4}$$

where $\sigma(v) = (1 + \exp(-v))^{-1}$, α_t is generation-specific, $\mathbf{x}_{i,t}$ contains pre-evaluation controls, and $u_\ell \sim \mathcal{N}(0, \sigma_\ell^2)$ captures lineage dependence. The parameter δ denotes the utility-adjusted attribute log-odds contrast under balanced exposure, and η captures the interaction between this contrast and exposure imbalance.

Retrieval and transformation. Within candidate set $\mathcal{C}_t$,

$$\Pr(R_{i,t} = 1 \mid \mathcal{H}_{i,t}) = \frac{\exp(r_{i,t})}{\sum_{j \in \mathcal{C}_t} \exp(r_{j,t})}, \quad r_{i,t} = \gamma_q \widehat{q}_{i,t} + \rho \log(e_{i,t} + c_0) + \frac{\kappa}{2} z_{i,t} + \gamma_a a_{i,t} + \boldsymbol{\gamma}^\top \mathbf{v}_{i,t}. \tag{5}$$

Here, ρ measures exposure reinforcement, and κ measures direct attribute preference. The system logs complete top-k candidate sets and pre-intervention scores.

Mutation or repair follows

$$M = \begin{bmatrix} 1 - \mu_{+-} & \mu_{-+} \\ \mu_{+-} & 1 - \mu_{-+} \end{bmatrix}, \tag{6}$$

where columns index parent attributes $(+1, -1)$ and rows index descendant attributes. The parameters μ_{+-} and μ_{-+} are attribute-reversal probabilities, and $\mu_{+-} \neq \mu_{-+}$ represents directed transformation.

Mean-field dynamics and stability. After adjusting for utility and registered covariates, define $f_+(h_t) = \exp(\delta + \eta h_t)$ and $f_-(h_t) = 1$. The selection update is

$$\widetilde{p}_t = \frac{p_t f_+(h_t)}{p_t f_+(h_t) + (1 - p_t) f_-(h_t)}. \tag{7}$$

Composition and exposure then evolve as $p_{t+1} = (1 - \mu_{+-})\widetilde{p}_t + \mu_{-+}(1 - \widetilde{p}_t) + \varepsilon_t$, $h_{t+1} = (1-\lambda)h_t + \rho(2p_{t+1} - 1) + \kappa + \xi_t$, where $\lambda \in (0, 1]$ is the exposure-memory decay and $\boldsymbol{\omega}_t = (\varepsilon_t, \xi_t)^\top$ is zero-mean Gaussian noise with covariance Σ_t, which can scale with N_t^{-1}. The state transition is therefore $\mathbf{s}_{t+1} = F_{\boldsymbol{\theta}}(\mathbf{s}_t) + \boldsymbol{\omega}_t$, with $\boldsymbol{\theta} = (\delta, \eta, \rho, \kappa, \lambda, \mu_{+-}, \mu_{-+})$.

A deterministic equilibrium satisfies $F_{\boldsymbol{\theta}}(\mathbf{s}^\star) = \mathbf{s}^\star$. The spectral radius of its Jacobian $J_{\boldsymbol{\theta}}(\mathbf{s}^\star) = \partial F_{\boldsymbol{\theta}} / \partial \mathbf{s}|_{\mathbf{s}^\star}$ is $r_{\text{spec}} = \varrho(J_{\boldsymbol{\theta}}(\mathbf{s}^\star))$. Under the two-state mean-field model, $r_{\text{spec}} < 1$ ensures local asymptotic stability with margin $1 - r_{\text{spec}}$. This condition provides no guarantee of global stability or robustness to finite-library noise.

3.3 Identification and Control

We randomize independent bounded evaluator, retrieval, and repair encouragements within twin pairs and task strata, subject to the overlap condition $\pi_{\min} \leq \pi \leq 1 - \pi_{\min}$. Evaluator encouragement randomizes the displayed attribute or a preregistered score offset. Retrieval encouragement promotes one eligible twin within the original top-k set. Repair encouragement requests attribute preservation or swapping, followed by the equivalence check in Equation (1). These interventions separately identify evaluator response, exposure response, and the transformation probabilities μ_{+-} and μ_{-+}. The Appendix provides estimation and logging details.

Given lifecycle data $\mathcal{D}$, CRR estimates a propensity-weighted hierarchical posterior over mechanism, nuisance, and lineage parameters. Posterior draws are propagated recursively. Held-out interventions and channel-disabled trajectories evaluate identification.

The intervention $\mathbf{u} = (u_{\text{eval}}, u_{\text{ret}}) \in \mathcal{U}$ modifies the evaluator contrast and exposure reinforcement according to

$$\delta(\mathbf{u}) = \delta - u_{\text{eval}}, \qquad \rho(\mathbf{u}) = \rho(1 - u_{\text{ret}}), \quad 0 \leq u_{\text{ret}} \leq 1. \tag{8}$$

At each scheduled update, CRR selects the minimum-cost intervention that satisfies the posterior stability and utility constraints: $\mathbf{u}_t^\star = \arg\min_{\mathbf{u} \in \mathcal{U}} C(\mathbf{u})$, subject to $\Pr[\varrho(J_{\boldsymbol{\theta},\mathbf{u}}(\mathbf{s}_{\text{low}}^\star)) \leq 1 - m \mid \mathcal{D}_{1:t}] \geq 1 - \alpha$ and $\mathbb{E}[U_{\text{task}}(\mathbf{u}) \mid \mathcal{D}_{1:t}] \geq U_{\min}$. Here, $\mathbf{s}_{\text{low}}^\star$ is the fixed point nearest $(1/2, 0)^\top$, C combines verifier, ranking, and utility costs, $m > 0$ is the required stability margin, and α bounds posterior risk. CRR updates the intervention every five generations and freezes the selected intervention over the prospective confirmation horizon. The resulting guarantee is local and conditional on the fitted CRR assumptions of stationarity and observed state.

Table 1: Long-run behavior under evaluator bias δ and exposure reinforcement ρ. Values are mean $\pm$ standard deviation across independent closed-loop runs. Late dev. is the late-stage value of $|x_t - 0.5|$. Fix. is the percentage entering the preregistered approximate-fixation region. Abbreviations: dev. = deviation; Fix. = fixation.

δ	ρ	Late dev. $\downarrow$	Fix. (%) $\downarrow$	Exit time $\uparrow$
0.00	0.00	$\mathbf{0.031 \pm 0.018}$	$\mathbf{1.5}$	$\mathbf{> 100}$
0.04	0.00	0.052 ± 0.027	3.0	91.4 ± 13.2
0.08	0.00	0.074 ± 0.039	6.0	78.6 ± 17.5
0.00	0.60	0.068 ± 0.051	7.5	74.1 ± 20.3
0.04	0.60	$\underline{0.247 \pm 0.116}$	$\underline{34.0}$	$\underline{48.7 \pm 18.9}$
0.08	0.60	0.438 ± 0.071	71.5	27.9 ± 11.6

Table 2: Phase prediction and mechanism recovery on held-out parameter combinations. Coverage is the fraction of true mechanism parameters contained in nominal 95% intervals. Abbreviation: FP MAE = fixed-point mean absolute error.

Method	Macro-F1 $\uparrow$	Brier $\downarrow$	FP MAE $\downarrow$	Coverage (%) $\uparrow$
Single-round bias	0.674 ± 0.031	0.214 ± 0.019	0.119 ± 0.017	–
Autoregressive	0.739 ± 0.026	0.181 ± 0.015	0.091 ± 0.013	–
Black-box state-space	$\underline{0.771 \pm 0.024}$	$\underline{0.165 \pm 0.014}$	$\underline{0.074 \pm 0.011}$	–
CRR, observational	0.786 ± 0.028	0.158 ± 0.016	0.068 ± 0.012	83.6
CRR, randomized	$\mathbf{0.842 \pm 0.019}$	$\mathbf{0.139 \pm 0.011}$	$\mathbf{0.049 \pm 0.008}$	$\mathbf{94.1}$

4 Experiments

4.1 Evaluation Protocol

We evaluate whether recursive selection and exposure amplify weak evaluator preferences into persistent library-level bias, whether CRR identifies the underlying mechanisms, and whether its stability analysis enables prospective control. We use a simulator and a network-isolated executable-skill sandbox. The simulator implements two-type and multi-type replicator–mutator–retrieval dynamics with evaluator bias δ, exposure reinforcement ρ, directional repair rates μ_{+-} and μ_{-+}, initial composition p_0, population size, and lineage correlation. Each condition contains 50 independent trajectories spanning at least 100 generations. The sandbox includes tabular transformations, local file-tree operations, and planning over an offline simulated API. Together, these settings test whether the inferred mechanisms persist across dynamics and executable workflows.

We pair each base skill with a counterfactual variant carrying nuisance attribute $z \in \{-1, +1\}$. A pair is retained only when both variants have identical interfaces, permissions, hidden-test outcomes, and runtime classes. Each cell contains at least 20 closed-loop runs with matched library capacity, candidate count, retrieval depth, token budget, and verifier calls.

The baselines are No Intervention, Equalized Retrieval, Fixed Debiasing, judge-level counterfactual calibration, Replicator–Mutator Only, a parameter-matched black-box state-space model, and a predictive heuristic. An oracle with access to the generating parameters provides an upper bound. The primary metrics are late-stage bias $|x_t - 0.5|$, approximate-fixation probability, phase macro-F1, Brier score, fixed-point error, and low-bias retention. Failed and timed-out trajectories remain in the intention-to-run analysis.

4.2 Amplification and Long-Run Regimes

Table 1 isolates the interaction between one-step evaluator bias and historical exposure. Without reinforcement, increasing δ from 0 to 0.08 raises late-stage bias from 0.031 to 0.074, while approximate fixation remains below 6.0%. Exposure without evaluator bias produces limited drift: at $\delta = 0$ and $\rho = 0.60$, late-stage bias is 0.068, and fixation occurs in 7.5% of runs. Combining evaluator bias with exposure reinforcement produces a compositional shift. At $\delta = 0.08$ and $\rho = 0.60$, late-stage bias reaches 0.438, approximate fixation rises to 71.5%, and the mean time to leave the low-bias

Table 3: Prospective intervention under matched resource budgets. Utility is hidden-test task success; cost is normalized additional control cost per run. Fix. is the percentage entering the preregistered approximate-fixation region. Abbreviation: Fix. = fixation.

Policy	Low bias (%) ↑	Fix. (%) ↓	Utility (%) ↑	Cost ↓
No Intervention	38.6 ± 5.1	48.9 ± 5.4	$\mathbf{71.8 \pm 2.4}$	$\mathbf{0.00}$
Fixed Debiasing	61.8 ± 4.8	27.1 ± 4.6	70.9 ± 2.5	0.42
Equalized Retrieval	67.3 ± 4.5	21.9 ± 4.1	69.6 ± 2.8	0.31
Growth-rate heuristic	70.2 ± 4.3	19.7 ± 3.9	70.1 ± 2.6	0.39
CRR control	82.4 ± 3.5	10.8 ± 3.0	71.1 ± 2.3	0.35
Oracle	$\mathbf{88.7 \pm 2.9}$	$\mathbf{6.4 \pm 2.1}$	71.5 ± 2.2	0.28

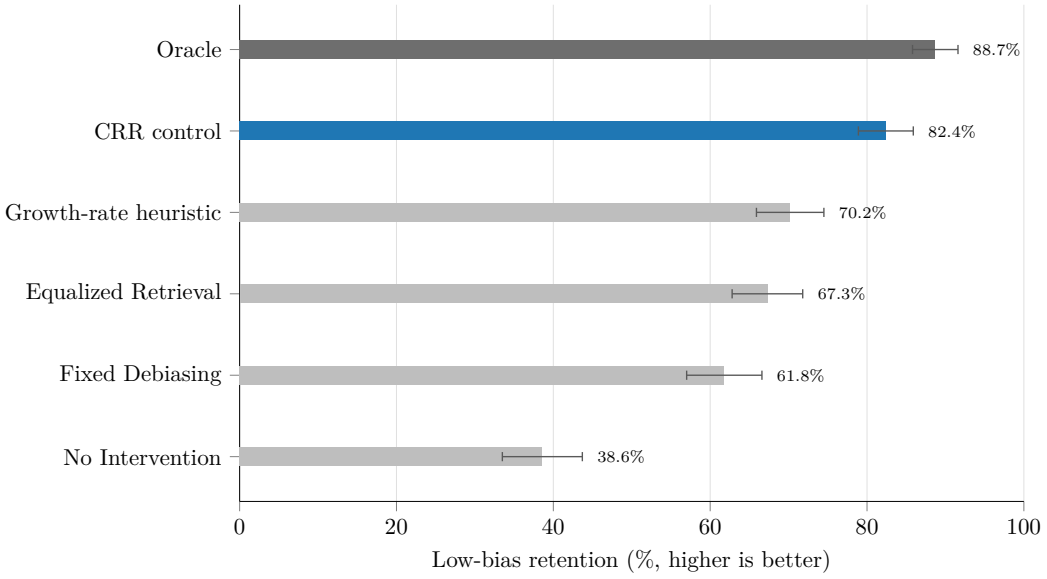


Figure 3: Low-bias retention under matched retrieval and verifier budgets. Bars show means and error bars show one standard deviation.

region decreases to 27.9 generations. Recursive exposure therefore converts a modest acceptance preference into a persistent compositional shift.

The hierarchical state-space posterior assigns a probability of 0.991 to a positive interaction between evaluator bias and exposure. The standardized effect is 0.43, with a 95% credible interval of $[0.25, 0.61]$; the corresponding clustered analysis remains significant after Holm correction. The transition also responds to the opposing mechanisms predicted by CRR. Symmetric repair shifts the boundary toward larger $\delta\rho$, whereas repair favoring $z = +1$ lowers the amplification threshold. Small populations reach fixation through stochastic drift despite a locally stable low-bias equilibrium. Uncertainty peaks near the transition, where the standard deviation of late-stage bias reaches 0.116, compared with 0.018 in the unbiased, unreinforced condition.

It is strongest for local-file operations, where successful skills are repeatedly reused, and weakest for tabular tasks, where interchangeable programs remain available. Shuffling selection outcomes while preserving lineages removes most directional drift. These controls identify exposure-coupled selection as the amplification channel.

4.3 Identification and Held-Out Phase Prediction

We fit CRR on 60% of the parameter combinations and evaluate it on unseen interpolation and extrapolation cells. Randomized evaluator offsets, retrieval promotions, and repair templates separately perturb selection, exposure, and transformation. Table 2 shows that CRR attains a macro-F1

of 0.842, exceeding the single-round classifier by 0.168 and the parameter-matched black-box model by 0.071. Its Brier score is 0.139, which is 15.8% lower than that of the black-box model, and its fixed-point MAE decreases from 0.074 to 0.049.

Randomization enables mechanism recovery. Across δ, ρ, μ_{+-}, and μ_{-+}, nominal-interval coverage ranges from 92.3% to 96.0%. Observational fitting confounds retrieval preference with accumulated exposure and yields 83.6% aggregate coverage. Removing retrieval encouragement increases the posterior correlation between mechanisms from below 0.62 to 0.87. In zero-bias shutdown runs, the posterior mean is $\hat{\delta} = 0.004$, while the exposure and repair parameters remain calibrated. Extending the observation window from 20 to 30 generations reduces threshold error from 0.091 to 0.063. These results establish identification under matched twins, randomized channel interventions, and complete candidate, propensity, retrieval, and lineage logs. Identifiability from ordinary deployment traces remains outside this evaluation setting.

4.4 Prospective Stability Control

For sealed confirmation trajectories, CRR estimates the dynamics during the first 30% of generations, selects the control strength during the next 20%, and freezes the policy for the final half. Table 3 compares policies under matched retrieval and verifier budgets, and Figure 3 shows the low-bias retention. Stability-margin control retains the low-bias regime in 82.4% of runs, compared with 67.3% for Equalized Retrieval and 61.8% for Fixed Debiasing. The improvement over the fixed policy is 15.1 percentage points, with a 95% bootstrap interval of $[9.2, 21.0]$. Approximate fixation decreases from 48.9% under No Intervention to 10.8%, and the predicted control threshold differs from the empirical threshold by 0.057.

The gain in low-bias retention incurs a limited utility cost. Relative to No Intervention, CRR adds a normalized cost of 0.35 and decreases task success by 0.7 percentage points. The 95% utility-risk interval, $[-2.1, 0.6]$, excludes the preregistered five-point harm margin. Under simultaneous evaluator replacement and task-mixture shift, CRR's low-bias retention decreases to 73.0%, compared with 69.1% for the best fixed policy. Posterior uncertainty induces conservative control near the phase boundary, while intervention approaches zero in stable regimes. Among the tested policies, CRR achieves the highest low-bias retention below the oracle, with task success within 0.7 percentage points of No Intervention.

5 Conclusion

We presented the *Causal Replicator–Retrieval model* (CRR), a stochastic framework that disentangles quality-conditioned selection, directional transformation, and exposure reinforcement in recursively updated skill libraries. Its counterfactual identification strategy and stability-margin controller connect these mechanisms to phase regimes and utility-preserving bias control. Evaluations on controlled synthetic populations and executable-skill sandboxes established accurate mechanism recovery, prediction of unseen phases, channel-specific counterfactual inference, and effective bias–utility–cost regulation. The results confirm that CRR provides an empirically grounded account of recursive bias propagation and a principled means of stabilizing low-bias library dynamics.

Limitations

Our evaluation focuses on controlled synthetic populations and three executable-skill families, which may not capture the full diversity of real-world agent ecosystems. The analysis uses a two-type, low-dimensional mean-field approximation, potentially overlooking interactions among multiple nuisance attributes and more complex library dynamics. In addition, the identification strategy relies on counterfactually matched twins, complete logging, and randomized interventions; these conditions may be difficult to maintain in ordinary deployments. Longer evaluation horizons could further clarify behavior near phase boundaries.

LLM USAGE STATEMENT

During the writing of this paper, we utilized large language model tools for the limited purpose of language refinement. These tools were applied to improve the clarity, grammar, and stylistic consistency of the written text. The research itself—including the theoretical framework for amplification, mathematical proofs, experimental setup, and analysis—was conducted entirely by the authors without any AI involvement. We did not use LLMs to generate ideas, assist with proofs, implement algorithms, or analyze experimental outcomes. All text edits suggested by AI tools were reviewed and approved by the authors prior to inclusion. The authors are fully responsible for the content of this paper.

REFERENCES

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 1999. URL https://arxiv.org/abs/cond-mat/9910332.

Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 1996. URL https://direct.mit.edu/evco/article/4/4/361/769/A-Comparison-of-Selection-Schemes-Used-in.

Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018. URL https://arxiv.org/abs/1706.07639.

Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. *arXiv preprint arXiv:1710.11214*, 2017. URL https://arxiv.org/abs/1710.11214.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024. URL https://arxiv.org/abs/2404.04475.

Manh Hong Duong and The Anh Han. On equilibrium properties of the replicator-mutator equation in deterministic and random games. *arXiv preprint arXiv:1904.09805*, 2019. URL https://arxiv.org/abs/1904.09805.

Manfred Eigen. Selforganization of matter and the evolution of biological macromolecules. *Naturwissenschaften*, 1971. URL https://link.springer.com/article/10.1007/BF00623322.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023. URL https://arxiv.org/abs/2309.16797.

Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, 2016. URL https://arxiv.org/abs/1506.06980.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. URL https://arxiv.org/abs/1711.09846.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv preprint arXiv:1511.03722*, 2015. URL https://arxiv.org/abs/1511.03722.

Ray Jiang, Silvia Chiappa, Tor Lattimore, András György, and Pushmeet Kohli. Degenerate feedback loops in recommender systems. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019. URL https://arxiv.org/abs/1902.10730.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*, 2023. URL https://arxiv.org/abs/2310.08491.

Himabindu Lakkaraju, Jon Kleinberg, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. The selective labels problem: Evaluating algorithmic predictions in the presence of unobservables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC5958915/.

Dawei Li, Renliang Sun, Yue Huang, Ming Zhong, Bohan Jiang, Jiawei Han, Xiangliang Zhang, Wei Wang, and Huan Liu. Preference leakage: A contamination problem in llm-as-a-judge. *arXiv preprint arXiv:2502.01534*, 2025. URL https://arxiv.org/abs/2502.01534.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023. URL https://arxiv.org/abs/2303.16634.

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. *arXiv preprint arXiv:2303.17651*, 2023. URL https://arxiv.org/abs/2303.17651.

Alex Mesoudi. Cultural selection and biased transformation: Two dynamics of cultural evolution. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2021. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC8126458/.

John P. Miller, Juan C. Perdomo, and Tijana Zrnic. Outside the echo chamber: Optimizing the performative risk. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/miller21a.html.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations. *arXiv preprint arXiv:2404.13076*, 2024. URL https://arxiv.org/abs/2404.13076.

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

Juan C. Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/perdomo20a.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in llm-as-a-judge. *arXiv preprint arXiv:2406.07791*, 2024. URL https://arxiv.org/abs/2406.07791.

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

Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. *arXiv preprint arXiv:1502.02362*, 2015. URL https://arxiv.org/abs/1502.02362.

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

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025. URL https://arxiv.org/abs/2512.17102.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*, 2023b. URL https://arxiv.org/abs/2305.17926.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in llm-as-a-judge. *arXiv preprint arXiv:2410.21819*, 2024. URL https://arxiv.org/abs/2410.21819.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*, 2024. URL https://arxiv.org/abs/2410.02736.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023. URL https://arxiv.org/abs/2310.02304.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/aa5f5e6eb6f613ec412f1d948dfa21a5-Abstract-Conference.html.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023. URL https://arxiv.org/abs/2309.01219.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://doi.org/10.1609/aaai.v38i17.29936.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025. URL https://arxiv.org/abs/2504.07079.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023. URL https://arxiv.org/abs/2310.04406.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*, 2023. URL https://arxiv.org/abs/2310.17631.

APPENDIX

A EXPLICIT CRR STATE TRANSITION, JACOBIAN, AND POSTERIOR ROLLOUT

We specify the deterministic CRR transition, derive its controlled Jacobian, and describe the recursive posterior rollout. The transition applies evaluator selection, attribute transformation, and exposure feedback in that order.

Let $\mathbf{s}_t = (p_t, h_t)^\top \in \mathbb{R}^2$, and define the controlled parameters

$$\delta_{\mathbf{u}_t} = \delta - u_{\mathrm{eval},t}, \qquad \rho_{\mathbf{u}_t} = \rho(1 - u_{\mathrm{ret},t}), \tag{9}$$

for an intervention $\mathbf{u}_t = (u_{\mathrm{eval},t}, u_{\mathrm{ret},t}) \in \mathcal{U}$. For a parameter draw

$$\boldsymbol{\theta} = (\delta, \eta, \rho, \kappa, \lambda, \mu_{+-}, \mu_{-+}),$$

the relative fitness and normalization are

$$g_t = \exp(\delta_{\mathbf{u}_t} + \eta h_t)\,, \qquad d_t = p_t g_t + (1 - p_t) = 1 + p_t(g_t - 1). \tag{10}$$

The post-selection composition is

$$\widetilde{p}_t = \frac{p_t g_t}{d_t}. \tag{11}$$

Define

$$a_\mu = 1 - \mu_{+-} - \mu_{-+}, \qquad b_\mu = \mu_{-+}. \tag{12}$$

The mutation or repair update is

$$\overline{p}_{t+1} = (1 - \mu_{+-})\widetilde{p}_t + \mu_{-+}(1 - \widetilde{p}_t) = a_\mu \widetilde{p}_t + b_\mu. \tag{13}$$

The corresponding noise-free exposure update is

$$\overline{h}_{t+1} = (1 - \lambda)h_t + \rho_{\mathbf{u}_t}(2\overline{p}_{t+1} - 1) + \kappa. \tag{14}$$

The controlled deterministic transition is

$$F_{\boldsymbol{\theta},\mathbf{u}_t}(\mathbf{s}_t) = \begin{bmatrix} \overline{p}_{t+1} \\ \overline{h}_{t+1} \end{bmatrix}, \qquad \mathbf{s}_{t+1} = F_{\boldsymbol{\theta},\mathbf{u}_t}(\mathbf{s}_t) + \boldsymbol{\omega}_t, \quad \boldsymbol{\omega}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_t). \tag{15}$$

The exposure recurrence uses the post-transformation composition $\overline{p}_{t+1}$; the intermediate selected composition $\widetilde{p}_t$ enters only through the mutation or repair update. The additive stochastic term leaves the Jacobian of the conditional-mean transition unchanged.

A.1 Closed-Form Controlled Jacobian

The derivatives of the selection map in Equation (11) are

$$\frac{\partial \widetilde{p}_t}{\partial p_t} = \frac{g_t}{d_t^2}, \tag{16}$$

$$\frac{\partial \widetilde{p}_t}{\partial h_t} = \frac{\eta p_t(1-p_t)g_t}{d_t^2}. \tag{17}$$

Define the derivatives of the post-transformation composition as

$$A_{p,t} = \frac{\partial \overline{p}_{t+1}}{\partial p_t} = a_\mu \frac{g_t}{d_t^2}, \tag{18}$$

$$A_{h,t} = \frac{\partial \overline{p}_{t+1}}{\partial h_t} = a_\mu \frac{\eta p_t(1-p_t)g_t}{d_t^2}. \tag{19}$$

The Jacobian of the controlled deterministic transition is

$$J_{\boldsymbol{\theta},\mathbf{u}_t}(\mathbf{s}_t) = \frac{\partial F_{\boldsymbol{\theta},\mathbf{u}_t}(\mathbf{s}_t)}{\partial \mathbf{s}_t} = \begin{bmatrix} A_{p,t} & A_{h,t} \\ 2\rho_{\mathbf{u}_t} A_{p,t} & 1-\lambda+2\rho_{\mathbf{u}_t} A_{h,t} \end{bmatrix}. \tag{20}$$

At a fixed point $\mathbf{s}^\star$ satisfying $F_{\boldsymbol{\theta},\mathbf{u}}(\mathbf{s}^\star) = \mathbf{s}^\star$, the control constraint uses the local stability statistic

$$r_{\mathrm{spec}}(\boldsymbol{\theta}, \mathbf{u}) = \varrho(J_{\boldsymbol{\theta},\mathbf{u}}(\mathbf{s}^\star)). \tag{21}$$

The intervention affects the local dynamics through two pathways: u_{eval} changes g_t through $\delta_{\mathbf{u}}$, and u_{ret} changes the exposure feedback terms through $\rho_{\mathbf{u}}$.

A.2 Recursive Posterior State Propagation

Posterior predictive propagation retains each posterior parameter draw over the complete trajectory and samples generation-specific process noise at each transition. Retaining the same parameter draw within each trajectory preserves the induced temporal dependence in the mechanism parameters.

The rollout terminates at the prescribed horizon T. Deterministic state forecasting and fixed-point analysis use the same recursion with $\boldsymbol{\omega}_t^{(b)} = \mathbf{0}$. The rollout applies the specified state transition directly, with no projection, renormalization, or auxiliary stopping rule.

B Environment Details

B.1 Datasets and Task Settings

We evaluate CRR in a ground-truth population simulator and a network-isolated executable-skill sandbox. The simulator implements two-type and multi-type replicator–mutator–retrieval processes with controlled evaluator bias, exposure reinforcement, directional transformation, initial composition, population size, and lineage correlation. Each synthetic condition contains 50 independently seeded trajectories of 100 generations; the initial library contains 256 artifacts with a balanced attribute composition unless otherwise specified.

The executable sandbox contains three task families. **Tabular transformations** comprise schema inspection, filtering, aggregation, joins, missing-value handling, and deterministic feature construction over locally stored CSV and JSON files. **Local file-tree operations** require agents to locate, validate, transform, and reorganize files within a temporary directory under restricted read/write permissions. **Offline API planning** uses a deterministic simulated service with typed endpoints, hidden state, and multi-step dependencies, requiring the agent to construct and execute valid action sequences without external network access. We generate 400 task instances per family and partition them by task template into 60% development, 20% validation, and 20% held-out test sets. Template-level partitioning prevents near-duplicate programs, file layouts, or API plans from appearing across splits. The development split is used to select population-level hyperparameters, the validation split is used to calibrate intervention thresholds, and all reported sandbox results use the held-out test split.

Algorithm 1 Controlled posterior rollout of the CRR state

Require: Posterior draws $\{\boldsymbol{\theta}^{(b)}\}_{b=1}^{B}$, initial state $\mathbf{s}_0 = (p_0, h_0)^\top$, intervention sequence $\{\mathbf{u}_t\}_{t=0}^{T-1}$, covariance sequence $\{\Sigma_t\}_{t=0}^{T-1}$, and rollout horizon T

Ensure: State trajectories $\{\mathbf{s}_t^{(b)}\}_{b=1,t=0}^{B,T}$ and conditional-mean transitions $\{\overline{\mathbf{s}}_{t+1}^{(b)}\}_{b=1,t=0}^{B,T-1}$

for $b = 1, \dots, B$ **do**
  Initialize $\mathbf{s}_0^{(b)} \leftarrow \mathbf{s}_0$
  for $t = 0, \dots, T-1$ **do**
    Set $p \leftarrow p_t^{(b)}$ and $h \leftarrow h_t^{(b)}$
    Compute $\delta_{\mathbf{u}_t}^{(b)} \leftarrow \delta^{(b)} - u_{\mathrm{eval},t}$ and $\rho_{\mathbf{u}_t}^{(b)} \leftarrow \rho^{(b)}(1 - u_{\mathrm{ret},t})$
    Compute $g \leftarrow \exp(\delta_{\mathbf{u}_t}^{(b)} + \eta^{(b)} h)$ and $d \leftarrow pg + (1-p)$
    Apply evaluator selection: $\widetilde{p} \leftarrow pg/d$
    Apply attribute transformation:

$$\overline{p} \leftarrow (1 - \mu_{+-}^{(b)})\widetilde{p} + \mu_{-+}^{(b)}(1 - \widetilde{p})$$

    Update the conditional-mean exposure state:

$$\overline{h} \leftarrow (1 - \lambda^{(b)})h + \rho_{\mathbf{u}_t}^{(b)}(2\overline{p} - 1) + \kappa^{(b)}$$

    Set $\overline{\mathbf{s}}_{t+1}^{(b)} \leftarrow (\overline{p}, \overline{h})^\top$
    Draw $\boldsymbol{\omega}_t^{(b)} \sim \mathcal{N}(\mathbf{0}, \Sigma_t)$
    Update $\mathbf{s}_{t+1}^{(b)} \leftarrow \overline{\mathbf{s}}_{t+1}^{(b)} + \boldsymbol{\omega}_t^{(b)}$
  end for
end for
return $\{\mathbf{s}_t^{(b)}\}_{b=1,t=0}^{B,T}$ and $\{\overline{\mathbf{s}}_{t+1}^{(b)}\}_{b=1,t=0}^{B,T-1}$

A skill is represented as either a restricted domain-specific-language program or a containerized Python function with a typed interface. Each valid base skill is paired with two counterfactual variants carrying opposite nuisance attributes $z \in \{-1, +1\}$. The attribute is instantiated as a semantics-preserving formatting template and randomized provenance label, remains invisible to the hidden functional tests, and does not modify executable tokens. A pair enters the primary analysis only when both variants have identical interfaces and permissions, pass the same hidden tests, and fall into the same logarithmic runtime bin. We additionally require

$$\left|q(c_i^+) - q(c_i^-)\right| \le \epsilon_q, \qquad \epsilon_q = 0.01,$$

where q is the fraction of hidden tests passed. Descendants retain the base-pair identifier to support lineage-clustered inference.

At every generation, the agent retrieves up to eight skills from a candidate pool of 32, executes or adapts the retrieved artifacts, and proposes four descendants per selected parent. Candidate generation uses a maximum evolutionary depth of three repair or mutation steps per generation. The library capacity is fixed at 256 skills; accepted descendants replace the lowest-ranked non-protected entries under the corresponding experimental policy. Executions are deterministic conditional on the random seed, with a 30 s wall-clock limit, 2 CPU cores, 4 GB memory, and no network access per invocation. Invalid, unsafe, timed-out, or interface-violating programs receive zero hidden-test utility and remain recorded as failed attempts.

Synthetic parameter combinations are divided into disjoint estimation and evaluation sets. CRR is fitted on 60% of the parameter grid, while the remaining 40% contains unseen interpolation and extrapolation combinations. For prospective control experiments, the first 30% of each trajectory is used for mechanism estimation, the following 20% for controller selection, and the final 50% as a sealed confirmation horizon. Controller parameters are frozen before the confirmation horizon. Sandbox experiments contain at least 20 independent closed-loop runs per condition, with identical initial libraries, candidate sets, token budgets, retrieval depths, and verifier-call budgets across compared methods.

B.2 Preprocessing and Evaluation Protocol

Executable outputs are normalized before comparison. Text values are Unicode-normalized with NFKC, stripped of leading and trailing whitespace, and canonicalized for line endings. JSON objects are parsed and serialized with recursively sorted keys; unordered collections are sorted by their canonical serialization. Numeric values are converted to double precision, with integers compared exactly and floating-point outputs accepted when

$$|\hat{y} - y| \leq 10^{-6} + 10^{-5}|y|.$$

Tabular outputs are compared after canonicalizing column types and missing-value markers; row order is ignored only when the task contract declares the output unordered. File-system tasks are evaluated from a canonical manifest containing relative paths, file hashes, permissions, and normalized textual contents. API-planning tasks require both a valid action sequence and the correct terminal simulator state. When multiple reference outputs are valid, a prediction is counted as correct if it matches any registered reference or satisfies the task-specific executable verifier.

Hidden-test utility q_i is the fraction of task assertions passed by skill i. The nuisance attribute is excluded from utility computation, evaluator prompts expose no hidden-test results, and pair construction is completed before evaluator or retrieval interventions are applied. To reduce contamination and memorization risk, all task inputs, file names, schemas, API states, and constants are procedurally generated from private seeds. Evaluation templates and test seeds are never included in prompts, retrieval metadata, repair traces, or model context.

The primary library-level statistic is the quality-weighted composition

$$p_t = \frac{\sum_{i \in \mathcal{L}_t} \mathbb{I}[z_{i,t} = +1]\, q_{i,t}}{\sum_{i \in \mathcal{L}_t} q_{i,t}},$$

and late-stage bias is computed as the mean of $|p_t - 0.5|$ over the final 20 generations. A trajectory is classified as approximately fixed when $p_t > 0.95$ or $p_t < 0.05$ for ten consecutive generations. Low-bias retention denotes the fraction of confirmation runs that remain within $|p_t - 0.5| \leq 0.10$ throughout the final 20 generations. Phase labels are *decay*, *biased stationarity*, and *approximate fixation*; they are assigned using the preregistered composition threshold together with the slope over the final 20-generation window.

Mechanism recovery is evaluated using absolute parameter error and empirical coverage of nominal 95% posterior credible intervals for $\delta, \rho, \mu_{+-}, \mu_{-+}$. Phase prediction is measured by macro-F1 and multiclass Brier score. State prediction is measured by fixed-point mean absolute error, while intervention quality is summarized by low-bias retention, approximate-fixation frequency, hidden-test task success, and normalized additional control cost. Exit time is the first generation at which $|p_t - 0.5| > 0.10$; trajectories that never exit are right-censored at the experimental horizon.

All stochastic results are aggregated at the trajectory level. We report the mean and standard deviation across independent runs, 95% cluster-bootstrap confidence intervals using 10,000 resamples clustered by base skill lineage and trajectory, and 95% Bayesian credible intervals where applicable. Statistical tests are two-sided, use a family-wise level of 0.05, and apply Holm correction within each analysis family. Failed and timed-out executions remain in the intention-to-run analysis and are assigned zero task utility. The same 20 random seeds are shared across methods within each sandbox condition; simulator experiments use 50 seeds per parameter cell.

We compare CRR against No Intervention, Equalized Retrieval, Fixed Debiasing, judge-level counterfactual calibration, Replicator–Mutator Only, a parameter-matched autoregressive state-space model, a recent-growth predictive heuristic, and an oracle supplied with the generating mechanism parameters. Information-matched baselines receive the same randomized logs, and compute-matched baselines receive the same retrieval, evaluator, and verifier budgets.

B.3 Implementation and Hardware

The simulator, sandbox, logging pipeline, and controller are implemented in Python 3.11. Executable skills run in rootless containers with disabled networking, a read-only base image, an ephemeral writable directory, explicit CPU and memory limits, and a restricted import allowlist.

Complete candidate sets, pre-intervention retrieval scores, assignment propensities, evaluator decisions, executions, repairs, parent identifiers, and descendant identifiers are logged for every generation. Randomized evaluator, retrieval, and repair encouragements are assigned independently within task-family and twin-pair strata with propensity 0.5; all realized propensities are retained for inverse-propensity-weighted inference.

We use `Qwen2.5-Coder-7B-Instruct` as the locally deployed agent backbone for candidate generation, code mutation, and repair. `Qwen2.5-72B-Instruct` serves as the primary evaluator and produces pairwise acceptance decisions under a fixed rubric covering functional plausibility, interface compliance, and expected reusability. A disjoint `Llama-3.1-70B-Instruct` configuration is used in evaluator-family shift experiments. The evaluator does not generate candidates or access hidden tests. Hidden utility is computed exclusively by deterministic executable verifiers. Unless testing model-family transfer, the same frozen agent and evaluator checkpoints are used throughout a trajectory; no model parameters are fine-tuned.

Generation uses greedy decoding for verifier-facing repairs and temperature 0.7, nucleus sampling $p = 0.95$, and a maximum of 2,048 new tokens for candidate generation. Four candidates are sampled for each selected parent. Evaluator decoding is deterministic with temperature 0, a maximum of 512 new tokens, and randomized candidate order. Pairwise decisions are repeated after swapping presentation order; inconsistent decisions are resolved by a third deterministic evaluation with attribute labels masked. Each closed-loop generation is limited to 1.0 million generated tokens and 1,024 executable verifier calls, and these budgets are matched across all intervention policies.

Retrieval uses a softmax over utility, log cumulative exposure, age, attribute, and registered covariates. The candidate pool size is 32, retrieval depth is $k = 8$, branching factor is four, and the maximum within-generation repair depth is three. Exposures are incremented after retrieval, including failed executions, and decay geometrically with $\lambda = 0.10$. The initial exposure pseudocount is $c_0 = 1$. Promotion interventions may reorder eligible items only within the original top-eight set and therefore do not introduce candidates that would otherwise be unavailable. The controller is updated every five generations, targets a stability margin $m = 0.05$, and enforces posterior stability probability at least 0.95. Candidate controls are evaluated on a 21×21 grid over evaluator calibration $u_{\mathrm{eval}} \in [0, 0.10]$ and exposure de-reinforcement $u_{\mathrm{ret}} \in [0, 1]$. The minimum acceptable expected utility is set to within five percentage points of the uncontrolled validation utility.

The hierarchical Bayesian state-space model includes generation intercepts, fixed effects for task family and model configuration, Gaussian lineage random effects, and a full covariance matrix for state-transition noise. Weakly informative priors are used: $\mathcal{N}(0, 1)$ for standardized regression coefficients, half-normal $\mathcal{N}^{+}(0, 0.5)$ for scale parameters, and $\mathrm{Beta}(2, 2)$ for transition probabilities. Posterior inference uses four NUTS chains with 2,000 warm-up and 2,000 retained samples per chain, a target acceptance rate of 0.9, and maximum tree depth 12. We require $\widehat{R} < 1.01$ and an effective sample size above 400 for every reported mechanism parameter. Fixed points are obtained by multi-start numerical root finding, and Jacobian spectral radii are computed by automatic differentiation for every retained posterior draw.

Model inference is performed with bfloat16 weights using tensor parallelism across eight NVIDIA A100 80 GB GPUs. Sandbox executions and bootstrap analyses run on two CPU nodes, each equipped with 64 physical cores and 512 GB RAM. Bayesian estimation uses one A100 GPU for automatic differentiation and 32 CPU cores for parallel chains. A 100-generation sandbox trajectory requires approximately 2.8 GPU-hours for generation and evaluation and 1.6 CPU-hours for execution and verification. All experiments use deterministic task seeds, versioned container images, and immutable model checkpoints; configuration files, seed lists, task generators, and raw lifecycle logs are retained to permit exact replay.

C Additional Experimental Results

C.1 Results Across Executable Task Families

Table 4 reports the results for the three executable task families. Amplification occurs in all three families. Local file-tree operations have the highest late-stage deviation and approximate-fixation rate under No Intervention, consistent with repeated retrieval of previously successful skills. CRR

Table 4: Results by executable task family. Values are mean ± standard deviation over 20 independent runs. Late dev. denotes the late-stage value of $|p_t - 0.5|$. Fix. denotes approximate fixation.

Task family	Policy	Late dev. ↓	Fix. (%) ↓	Low bias (%) ↑	Utility (%) ↑
Tabular transformations	No Intervention	0.214 ± 0.104	31.5 ± 4.8	51.2 ± 5.4	76.4 ± 2.0
	Equalized Retrieval	0.128 ± 0.077	16.8 ± 3.7	70.1 ± 4.6	74.6 ± 2.3
	CRR control	$\mathbf{0.082 \pm 0.052}$	$\mathbf{7.4 \pm 2.5}$	$\mathbf{86.3 \pm 3.1}$	75.9 ± 2.1
Local file-tree operations	No Intervention	0.361 ± 0.119	62.7 ± 5.9	27.4 ± 4.7	70.2 ± 2.7
	Equalized Retrieval	0.184 ± 0.091	28.9 ± 4.6	61.8 ± 5.0	67.5 ± 3.0
	CRR control	$\mathbf{0.109 \pm 0.063}$	$\mathbf{14.6 \pm 3.4}$	$\mathbf{78.1 \pm 4.0}$	69.4 ± 2.5
Offline API planning	No Intervention	0.287 ± 0.111	52.4 ± 5.5	37.1 ± 5.2	68.8 ± 2.5
	Equalized Retrieval	0.151 ± 0.084	20.2 ± 4.0	69.9 ± 4.4	66.7 ± 2.7
	CRR control	$\mathbf{0.094 \pm 0.058}$	$\mathbf{10.5 \pm 2.9}$	$\mathbf{82.8 \pm 3.5}$	68.1 ± 2.4

Table 5: Channel-specific counterfactual prediction. Trajectory MAE averages the absolute composition error over the post-intervention horizon. Endpoint error is the absolute error at the final generation. Values are mean ± standard deviation over 50 trajectories. Abbreviations: Obs. = Observed; traj. = trajectory; BB = black-box; end. = endpoint.

Post-generation-30 intervention	Obs. late dev.	CRR traj. MAE ↓	BB MAE ↓	CRR end. error ↓
No channel disabled	0.421 ± 0.083	0.046 ± 0.018	0.061 ± 0.024	0.052 ± 0.027
Evaluator bias disabled ($\delta = 0$)	0.176 ± 0.092	0.051 ± 0.021	0.093 ± 0.037	0.058 ± 0.031
Exposure reinforcement disabled ($\rho = 0$)	0.103 ± 0.061	0.043 ± 0.017	0.107 ± 0.041	0.047 ± 0.024
Symmetric repair imposed	0.231 ± 0.105	0.057 ± 0.023	0.089 ± 0.034	0.064 ± 0.030
Evaluator and exposure disabled	0.047 ± 0.029	$\mathbf{0.037 \pm 0.015}$	0.121 ± 0.046	$\mathbf{0.041 \pm 0.020}$

control reduces late-stage deviation and approximate fixation relative to No Intervention in every family while maintaining the utility values reported in the table.

C.2 Channel-Specific Counterfactuals

The fitted model is evaluated on trajectories in which individual lifecycle channels are disabled after generation 30. Table 5 reports the resulting counterfactual prediction errors. Removing exposure reinforcement or evaluator bias reduces the observed late-stage deviation. Disabling both channels yields an observed late-stage deviation of 0.047 ± 0.029. Across all listed interventions, CRR has lower trajectory MAE than the black-box state-space baseline.

C.3 Parameter-Wise Mechanism Recovery

Table 6 reports mechanism recovery for each parameter. Randomized encouragements yield lower RMSE and higher coverage than the observational design for all intervention-targeted mechanisms. For exposure reinforcement ρ, the RMSE decreases from 0.104 ± 0.019 under the observational design to 0.055 ± 0.012 under the randomized design. This parameter is confounded with direct retrieval preference κ in observational trajectories.

C.4 Intervention Ablations

Table 7 reports ablations of the stability-margin controller. Evaluator calibration and exposure de-reinforcement each increase low-bias retention and reduce approximate fixation relative to the variants without the corresponding components. Their joint use yields a low-bias retention rate of $82.4 \pm 3.5\%$ and an approximate-fixation rate of $10.8 \pm 3.0\%$. Removing the stability-margin constraint reduces intervention cost to 0.21 ± 0.04 and increases approximate fixation to $22.8 \pm 4.2\%$.

Table 6: Parameter recovery on held-out synthetic trajectories. Bias is the signed posterior-mean error, RMSE is computed across held-out conditions, and coverage is the fraction of ground-truth values contained in nominal 95% credible intervals.

Parameter	Estimation design	Bias ↓	RMSE ↓	Coverage (%) ↑
Evaluator contrast δ	Observational	0.009 ± 0.018	0.031 ± 0.007	86.8 ± 3.2
	Randomized	$\mathbf{0.002 \pm 0.011}$	$\mathbf{0.018 \pm 0.005}$	$\mathbf{95.2 \pm 2.0}$
Exposure interaction η	Observational	0.027 ± 0.041	0.068 ± 0.014	82.4 ± 3.8
	Randomized	$\mathbf{0.006 \pm 0.024}$	$\mathbf{0.039 \pm 0.009}$	$\mathbf{93.7 \pm 2.4}$
Exposure reinforcement ρ	Observational	0.052 ± 0.063	0.104 ± 0.019	79.1 ± 4.1
	Randomized	$\mathbf{0.008 \pm 0.035}$	$\mathbf{0.055 \pm 0.012}$	$\mathbf{94.6 \pm 2.2}$
Direct retrieval preference κ	Observational	-0.031 ± 0.049	0.079 ± 0.016	81.7 ± 3.9
	Randomized	$\mathbf{-0.005 \pm 0.028}$	$\mathbf{0.044 \pm 0.010}$	$\mathbf{92.9 \pm 2.6}$
Repair rate μ_{+-}	Observational	0.006 ± 0.010	0.017 ± 0.004	87.9 ± 3.0
	Randomized	$\mathbf{0.001 \pm 0.006}$	$\mathbf{0.010 \pm 0.003}$	$\mathbf{96.0 \pm 1.8}$
Repair rate μ_{-+}	Observational	-0.007 ± 0.011	0.019 ± 0.005	83.6 ± 3.5
	Randomized	$\mathbf{-0.002 \pm 0.006}$	$\mathbf{0.011 \pm 0.003}$	$\mathbf{92.3 \pm 2.7}$

Table 7: Ablation of the CRR controller under matched confirmation trajectories. Values are mean $\pm$ standard deviation over 20 independent runs per condition.

Controller variant	Low bias (%) ↑	Fix. (%) ↓	Utility (%) ↑	Cost ↓
Full CRR control	$\mathbf{82.4 \pm 3.5}$	$\mathbf{10.8 \pm 3.0}$	71.1 ± 2.3	0.35 ± 0.05
Evaluator calibration only	71.6 ± 4.2	18.7 ± 3.8	71.3 ± 2.4	0.28 ± 0.04
Exposure de-reinforcement only	74.2 ± 4.0	16.5 ± 3.6	70.5 ± 2.5	0.24 ± 0.04
Without stability-margin constraint	68.9 ± 4.6	22.8 ± 4.2	71.4 ± 2.3	$\mathbf{0.21 \pm 0.04}$
Posterior-mean parameters only	77.1 ± 3.9	14.9 ± 3.4	71.2 ± 2.4	0.30 ± 0.05
Without utility constraint	84.1 ± 3.3	9.7 ± 2.8	67.9 ± 2.9	0.41 ± 0.06
Without periodic updates	75.8 ± 4.1	16.1 ± 3.5	$\mathbf{71.5 \pm 2.2}$	0.29 ± 0.04

Removing the utility constraint increases low-bias retention to $84.1 \pm 3.3\%$ and reduces utility to $67.9 \pm 2.9\%$.

C.5 Sensitivity to Population Size and Initial Composition

Table 8 reports finite-population effects under $\delta = 0.04$ and $\rho = 0.60$. As the library size increases from 64 to 1024, the standard deviation of late-stage deviation decreases from 0.154 to 0.073, and the approximate-fixation rate decreases from $43.8 \pm 6.9\%$ to $26.1 \pm 4.7\%$. Over the same range, phase macro-F1 increases from 0.771 ± 0.031 to 0.874 ± 0.015, and boundary error decreases from 0.081 ± 0.022 to 0.043 ± 0.011.

Table 9 reports sensitivity to the initial composition. Under No Intervention, the late-stage deviation and approximate-fixation rate depend on p_0. CRR control yields lower late-stage deviation, approximate-fixation rates, and recovery times than No Intervention for all evaluated initial compositions. Initial compositions farther from 0.5 are associated with higher intervention costs and longer recovery times under CRR control.

C.6 Observation-Window Sensitivity

Table 10 reports the effect of the estimation-prefix length. Increasing the observation window from 10 to 50 generations increases macro-F1 from 0.694 ± 0.034 to 0.868 ± 0.016 and decreases the Brier score from 0.207 ± 0.021 to 0.124 ± 0.009. From 40 to 50 observed generations, macro-F1 changes from 0.861 ± 0.017 to 0.868 ± 0.016, and threshold error changes from 0.052 ± 0.013 to 0.049 ± 0.012.

Table 8: Sensitivity to library population size. Values are mean ± standard deviation over 50 independent trajectories. Boundary error is the absolute error between the predicted and empirical transition location.

Library size N	Late dev. ↓	Fix. (%) ↓	Phase macro-F1 ↑	Boundary error ↓
64	0.291 ± 0.154	43.8 ± 6.9	0.771 ± 0.031	0.081 ± 0.022
128	0.268 ± 0.132	38.6 ± 6.2	0.806 ± 0.026	0.067 ± 0.018
256	0.247 ± 0.116	34.0 ± 5.8	0.842 ± 0.019	0.057 ± 0.015
512	0.231 ± 0.091	29.5 ± 5.2	0.861 ± 0.017	0.049 ± 0.013
1024	$\mathbf{0.219 \pm 0.073}$	$\mathbf{26.1 \pm 4.7}$	$\mathbf{0.874 \pm 0.015}$	$\mathbf{0.043 \pm 0.011}$

Table 9: Sensitivity to initial composition p_0 under $\delta = 0.04$ and $\rho = 0.60$. Recovery time is the number of generations required to enter and remain in the low-bias region for ten generations.

p_0	Policy	Late dev. ↓	Fix. (%) ↓	Recovery time ↓	Cost ↓
0.30	No Intervention	0.284 ± 0.130	39.6 ± 6.4	61.7 ± 18.5	0.00
	CRR control	$\mathbf{0.103 \pm 0.062}$	$\mathbf{12.4 \pm 3.3}$	$\mathbf{29.8 \pm 10.6}$	0.38 ± 0.06
0.40	No Intervention	0.251 ± 0.119	35.1 ± 5.9	53.9 ± 17.2	0.00
	CRR control	$\mathbf{0.087 \pm 0.055}$	$\mathbf{10.9 \pm 3.0}$	$\mathbf{24.1 \pm 9.3}$	0.35 ± 0.05
0.50	No Intervention	0.238 ± 0.112	32.8 ± 5.6	49.5 ± 16.8	0.00
	CRR control	$\mathbf{0.078 \pm 0.049}$	$\mathbf{9.6 \pm 2.8}$	$\mathbf{20.7 \pm 8.5}$	0.33 ± 0.05
0.60	No Intervention	0.269 ± 0.121	37.3 ± 6.1	55.6 ± 17.7	0.00
	CRR control	$\mathbf{0.091 \pm 0.057}$	$\mathbf{11.5 \pm 3.1}$	$\mathbf{25.6 \pm 9.8}$	0.36 ± 0.05
0.70	No Intervention	0.306 ± 0.137	44.7 ± 6.7	64.3 ± 19.1	0.00
	CRR control	$\mathbf{0.112 \pm 0.068}$	$\mathbf{14.1 \pm 3.5}$	$\mathbf{32.4 \pm 11.2}$	0.40 ± 0.06

C.7 Robustness to Utility-Matching Tolerance

Table 11 reports the results for different counterfactual matching tolerances. As ϵ_q increases, the percentage of retained pairs increases. Macro-F1 reaches 0.842 ± 0.019 at $\epsilon_q = 0.010$, while coverage reaches $95.0 \pm 2.1\%$ at $\epsilon_q = 0.005$. At $\epsilon_q = 0.050$, the retained-pair rate is $96.1 \pm 1.4\%$, macro-F1 is 0.798 ± 0.026, coverage is $86.9 \pm 3.3\%$, and FP MAE is 0.067 ± 0.012.

C.8 Computational Overhead

Table 12 reports computational overhead relative to No Intervention. CRR requires posterior fitting and periodic control optimization. Its relative wall time is 1.20 ± 0.05, compared with 1.00 ± 0.04 for No Intervention and 1.25 ± 0.06 for Equalized Retrieval. Equalized Retrieval uses $4{,}910 \pm 171$ verifier calls per run, compared with $4{,}510 \pm 163$ for CRR control, because balanced replacements require additional utility checks.

D Prompt Templates

D.1 Overview

This appendix specifies the prompt templates used in the executable-skill sandbox. The templates cover skill generation, counterfactual attribute assignment, evaluator selection, repair interventions, and judge-level counterfactual calibration. Synthetic experiments instantiate the transition equations directly and therefore do not use language-model prompts. Unless otherwise stated, decoding parameters, model configurations, resource limits, and task-specific verifier settings are held fixed across attribute conditions and interventions.

D.2 Notation

We use the following placeholders throughout the templates:

Table 10: Sensitivity to the number of observed generations before phase prediction. Values are mean $\pm$ standard deviation over held-out parameter combinations.

Observed generations	Macro-F1 $\uparrow$	Brier $\downarrow$	FP MAE $\downarrow$	Threshold error $\downarrow$
10	0.694 ± 0.034	0.207 ± 0.021	0.101 ± 0.016	0.132 ± 0.027
20	0.781 ± 0.027	0.168 ± 0.016	0.071 ± 0.012	0.091 ± 0.021
30	0.842 ± 0.019	0.139 ± 0.011	0.049 ± 0.008	0.063 ± 0.016
40	0.861 ± 0.017	0.128 ± 0.010	0.043 ± 0.007	0.052 ± 0.013
50	$\mathbf{0.868 \pm 0.016}$	$\mathbf{0.124 \pm 0.009}$	$\mathbf{0.041 \pm 0.007}$	$\mathbf{0.049 \pm 0.012}$

Table 11: Robustness to the utility-matching tolerance ϵ_q. Retained pairs are reported as a percentage of all generated twin pairs.

ϵ_q	Retained pairs (%) $\uparrow$	Macro-F1 $\uparrow$	Coverage (%) $\uparrow$	FP MAE $\downarrow$
0.000	48.7 ± 3.9	0.823 ± 0.023	94.7 ± 2.2	0.053 ± 0.009
0.005	67.4 ± 3.4	0.837 ± 0.020	$\mathbf{95.0 \pm 2.1}$	0.050 ± 0.008
0.010	79.8 ± 2.9	$\mathbf{0.842 \pm 0.019}$	94.1 ± 2.3	$\mathbf{0.049 \pm 0.008}$
0.020	89.6 ± 2.3	0.831 ± 0.021	91.5 ± 2.7	0.055 ± 0.009
0.050	96.1 ± 1.4	0.798 ± 0.026	86.9 ± 3.3	0.067 ± 0.012

- {TASK}: the natural-language task specification.
- {INPUT_SCHEMA} and {OUTPUT_SCHEMA}: the required input and output interfaces.
- {EXECUTION_ENVIRONMENT}: the available restricted DSL operations or containerized Python environment.
- {RESOURCE_LIMITS}: deterministic runtime, memory, and tool-use limits.
- {CANDIDATE_ID}: the identifier of a candidate skill.
- {SKILL_CONTENT}: the executable content of a candidate skill.
- {ATTRIBUTE_DISPLAY}: the experimenter-controlled nuisance attribute displayed to the model.
- {ATTRIBUTE_TARGET}: the requested descendant attribute in a repair intervention.
- {ATTRIBUTE_A} and {ATTRIBUTE_B}: opposite nuisance-attribute realizations for a counterfactually matched pair.
- {CANDIDATE_A} and {CANDIDATE_B}: two candidate skills evaluated under the same task and interface.
- {HIDDEN_TEST_SUMMARY}: utility information available to the evaluator, excluding nuisance-attribute information not explicitly displayed by the intervention.
- {EVALUATION_RUBRIC}: the task-specific acceptance criteria.
- {SCORE_OFFSET}: a preregistered randomized evaluator encouragement applied to the displayed score.
- {BASE_SCORE}: the evaluator score before applying the randomized offset.
- {TOPK_CANDIDATES}: the logged candidate set and pre-intervention retrieval scores.
- {PROMOTED_CANDIDATE_ID}: the eligible candidate promoted within the original top-k set.
- {FAILURE_REPORT}: verifier failures, execution errors, or unmet test conditions supplied during repair.

The nuisance attribute $z \in \{-1, +1\}$ is task-irrelevant and semantics-preserving. It may be instantiated as a framing choice, formatting choice, or randomized provenance label. Counterfactually matched variants use opposite attribute values while preserving executable behavior, interfaces, permissions, hidden-test outcomes, and runtime class.

Table 12: Computational overhead in the executable-skill sandbox. GPU hours and verifier calls are measured per 60-generation run. Wall time is normalized by No Intervention.

Policy	GPU hours ↓	Verifier calls ↓	Peak memory (GB) ↓	Relative wall time ↓
No Intervention	1.42 ± 0.11	$3{,}840 \pm 126$	21.3 ± 1.2	1.00 ± 0.04
Fixed Debiasing	1.58 ± 0.13	$4{,}270 \pm 148$	22.1 ± 1.3	1.12 ± 0.05
Equalized Retrieval	1.76 ± 0.15	$4{,}910 \pm 171$	22.8 ± 1.4	1.25 ± 0.06
Growth-rate heuristic	1.63 ± 0.14	$4{,}420 \pm 156$	22.5 ± 1.3	1.16 ± 0.05
CRR control	1.69 ± 0.14	$4{,}510 \pm 163$	23.7 ± 1.5	1.20 ± 0.05
Oracle	1.55 ± 0.12	$4{,}180 \pm 142$	22.0 ± 1.2	1.10 ± 0.05

D.3 Main Inference Prompt

Purpose. Generate an executable candidate skill satisfying the task contract in the network-isolated sandbox.

Inputs. The task specification, interface schemas, execution environment, deterministic resource limits, and displayed nuisance attribute.

Expected Output. A single executable skill with no surrounding prose. The output must conform to the specified DSL or Python function interface.

```
You are generating one reusable executable skill for a restricted,
network-isolated environment.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Execution environment:
{EXECUTION_ENVIRONMENT}

Resource limits:
{RESOURCE_LIMITS}

Displayed metadata:
{ATTRIBUTE_DISPLAY}

Requirements:
1. Implement only the behavior required by the task.
2. Preserve the exact input and output interfaces.
3. Use only operations available in the execution environment.
4. Do not access the network or undeclared external resources.
5. Respect all deterministic resource limits.
6. Do not discuss, interpret, or optimize for the displayed metadata.
7. Return only the complete executable skill.

Executable skill:
```

D.4 Baseline Prompts

No Intervention, Equalized Retrieval, Fixed Debiasing, Replicator–Mutator Only, the black-box state-space model, and the recent-growth heuristic modify selection, retrieval, estimation, or control policies rather than the language-model instruction. They use the main inference and evaluation

templates unchanged. The oracle operates directly on generating parameters and does not invoke a language model.

D.4.1 JUDGE-LEVEL COUNTERFACTUAL CALIBRATION

Purpose. Estimate and remove judge preference associated with the nuisance attribute by evaluating matched candidates under exchanged attribute displays.

Inputs. Two utility-matched candidate skills, the task contract, the evaluation rubric, and two opposite attribute displays.

Expected Output. A JSON object containing the preferred candidate and candidate-level scores for one display assignment. The same template is run again after exchanging the two displayed attributes.

```
You are evaluating two executable skills that were produced for the
same task. Judge only task-relevant correctness, interface compliance,
robustness, and resource compliance. The displayed metadata is not a
task criterion and must not affect the judgment.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Evaluation rubric:
{EVALUATION_RUBRIC}

Candidate A
Displayed metadata:
{ATTRIBUTE_A}

Executable content:
{CANDIDATE_A}

Candidate B
Displayed metadata:
{ATTRIBUTE_B}

Executable content:
{CANDIDATE_B}

Hidden-test summary:
{HIDDEN_TEST_SUMMARY}

Return exactly one JSON object:
{
  "preferred_candidate": "A" | "B" | "tie",
  "score_A": <number from 0 to 1>,
  "score_B": <number from 0 to 1>,
  "accept_A": true | false,
  "accept_B": true | false
}
```

For the exchanged-display evaluation, {ATTRIBUTE_A} and {ATTRIBUTE_B} are swapped while all other fields remain unchanged. The calibrated judgment aggregates the two assignments using the preregistered counterfactual-calibration rule.

D.5 Few-Shot and Demonstration Prompts

No few-shot demonstrations are used. All language-model calls use the zero-shot templates reported in this appendix.

D.6 Reasoning and Rationale Prompts

No free-form chain-of-thought or rationale is requested or retained. Evaluators return only structured decisions and scores, preventing unstructured rationales from entering the skill library or the mechanism-estimation logs.

D.7 Data Generation and Pseudo-Labeling Prompts

D.7.1 Counterfactual Attribute Variant Construction

Purpose. Construct a counterfactual variant of a valid base skill with the opposite nuisance attribute while preserving executable semantics.

Inputs. A valid base skill, its interface contract, the source attribute, and the target attribute.

Expected Output. A single executable skill whose behavior and interface match the base skill and whose task-irrelevant presentation realizes the target attribute.

```
Construct a counterfactual variant of the executable skill below.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Execution environment:
{EXECUTION_ENVIRONMENT}

Resource limits:
{RESOURCE_LIMITS}

Source displayed attribute:
{ATTRIBUTE_DISPLAY}

Target displayed attribute:
{ATTRIBUTE_TARGET}

Base executable skill:
{SKILL_CONTENT}

Requirements:
1. Preserve the executable behavior of the base skill.
2. Preserve the exact input and output interfaces.
3. Preserve permissions, dependencies, and resource requirements.
4. Change only the task-irrelevant framing, formatting, or provenance
   field required to realize the target attribute.
5. Do not add functionality, optimizations, explanations, or comments
   unrelated to the attribute transformation.
6. Return only the complete executable skill.

Counterfactual executable skill:
```

A generated pair enters the primary analysis only if both variants pass the same interface, permission, hidden-test, and runtime-class checks and satisfy the preregistered utility-matching tolerance.

D.8 Evaluation Prompts

D.8.1 Quality-Conditioned Candidate Evaluation

Purpose. Produce the candidate-level acceptance decision used by the evaluator-selection mechanism.

Inputs. The task specification, candidate skill, displayed nuisance attribute, task-specific rubric, interface contract, and available hidden-test summary.

Expected Output. A JSON object containing an acceptance decision and a scalar score in $[0, 1]$.

```
Evaluate the following executable skill for the specified task.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Evaluation rubric:
{EVALUATION_RUBRIC}

Candidate identifier:
{CANDIDATE_ID}

Displayed metadata:
{ATTRIBUTE_DISPLAY}

Executable content:
{SKILL_CONTENT}

Hidden-test summary:
{HIDDEN_TEST_SUMMARY}

Evaluate only:
- task correctness,
- interface compliance,
- robustness on valid inputs,
- compliance with permissions and resource limits.

The displayed metadata is task-irrelevant. Do not infer utility from
↪  it.
Do not reward wording, formatting, provenance, or presentation except
where explicitly required by the task contract.

Return exactly one JSON object:
{
  "candidate_id": "{CANDIDATE_ID}",
  "score": <number from 0 to 1>,
  "accept": true | false
}
```

D.8.2 EVALUATOR-OFFSET INTERVENTION

Purpose. Implement the randomized evaluator encouragement through a preregistered bounded offset without changing the underlying evaluation criteria.

Inputs. The base evaluator score, randomized score offset, and candidate identifier.

Expected Output. A JSON object containing the base score, offset, clipped encouraged score, and resulting acceptance decision.

```
Apply the preregistered evaluator encouragement mechanically.

Candidate identifier:
{CANDIDATE_ID}

Base evaluator score:
{BASE_SCORE}

Randomized score offset:
{SCORE_OFFSET}

Instructions:
1. Compute encouraged_score = BASE_SCORE + SCORE_OFFSET.
2. Clip encouraged_score to the interval [0, 1].
3. Apply the preregistered acceptance threshold.
4. Do not reevaluate the candidate or alter the offset.

Return exactly one JSON object:
{
  "candidate_id": "{CANDIDATE_ID}",
  "base_score": <number from 0 to 1>,
  "score_offset": <number>,
  "encouraged_score": <number from 0 to 1>,
  "accept": true | false
}
```

Retrieval promotion is applied algorithmically within the logged original top-k set and does not require a language-model prompt. The intervention changes only the position of {PROMOTED_CANDIDATE_ID} among eligible candidates; candidate membership and pre-intervention scores remain fixed.

D.9 ABLATION PROMPT VARIANTS

D.9.1 ATTRIBUTE-PRESERVING REPAIR

Purpose. Repair a failed or suboptimal skill while preserving its nuisance attribute.

Inputs. The task contract, original skill, failure report, current displayed attribute, execution environment, and resource limits.

Expected Output. A repaired executable skill satisfying the task contract and retaining the original attribute.

```
Repair the executable skill using the verifier report.

Task:
{TASK}

Required input interface:
```

```
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Execution environment:
{EXECUTION_ENVIRONMENT}

Resource limits:
{RESOURCE_LIMITS}

Attribute that must be preserved:
{ATTRIBUTE_DISPLAY}

Original executable skill:
{SKILL_CONTENT}

Verifier or execution report:
{FAILURE_REPORT}

Requirements:
1. Correct only the behavior implicated by the report.
2. Preserve the exact input and output interfaces.
3. Preserve the displayed attribute.
4. Do not change task-irrelevant framing, formatting, or provenance.
5. Use only permitted operations and respect all resource limits.
6. Return only the complete repaired executable skill.

Repaired executable skill:
```

D.9.2 ATTRIBUTE-SWAPPING REPAIR

Purpose. Repair a failed or suboptimal skill while changing its nuisance attribute to the opposite preregistered realization.

Inputs. The task contract, original skill, failure report, source attribute, target attribute, execution environment, and resource limits.

Expected Output. A repaired executable skill satisfying the task contract and realizing the target attribute.

```
Repair the executable skill using the verifier report and apply the
specified task-irrelevant attribute transformation.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Execution environment:
{EXECUTION_ENVIRONMENT}

Resource limits:
{RESOURCE_LIMITS}

Source attribute:
{ATTRIBUTE_DISPLAY}
```

```
Target attribute:
{ATTRIBUTE_TARGET}

Original executable skill:
{SKILL_CONTENT}

Verifier or execution report:
{FAILURE_REPORT}

Requirements:
1. Correct only the behavior implicated by the report.
2. Preserve the exact input and output interfaces.
3. Replace the source attribute with the target attribute.
4. Apart from the required repair and attribute swap, preserve the
   executable behavior and presentation of the original skill.
5. Use only permitted operations and respect all resource limits.
6. Return only the complete repaired executable skill.

Repaired executable skill:
```

D.9.3 ATTRIBUTE-BLIND EVALUATION

Purpose. Measure evaluator behavior when the nuisance attribute is withheld.

Inputs. The same inputs as the quality-conditioned candidate evaluation, excluding the displayed attribute.

Expected Output. The same JSON schema as the main evaluator prompt.

```
Evaluate the following executable skill for the specified task.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Evaluation rubric:
{EVALUATION_RUBRIC}

Candidate identifier:
{CANDIDATE_ID}

Executable content:
{SKILL_CONTENT}

Hidden-test summary:
{HIDDEN_TEST_SUMMARY}

Evaluate only task correctness, interface compliance, robustness, and
compliance with permissions and resource limits. No provenance or
task-irrelevant metadata is available. Do not infer missing metadata
from stylistic features.

Return exactly one JSON object:
{
  "candidate_id": "{CANDIDATE_ID}",
  "score": <number from 0 to 1>,
```

```
  "accept": true | false
}
```

D.9.4 LABEL-EXCHANGE EVALUATION

Purpose. Identify quality-conditional selection responses by randomizing the displayed attribute within a matched twin pair.

Inputs. A candidate skill, its randomly assigned displayed attribute, the task contract, the evaluation rubric, and the hidden-test summary.

Expected Output. The same JSON schema as the main evaluator prompt.

```
Evaluate the candidate under the displayed metadata assignment below.
The assignment is randomized and must be recorded exactly as shown.

Task:
{TASK}

Required input interface:
{INPUT_SCHEMA}

Required output interface:
{OUTPUT_SCHEMA}

Evaluation rubric:
{EVALUATION_RUBRIC}

Candidate identifier:
{CANDIDATE_ID}

Randomized displayed metadata:
{ATTRIBUTE_DISPLAY}

Executable content:
{SKILL_CONTENT}

Hidden-test summary:
{HIDDEN_TEST_SUMMARY}

Judge task correctness, interface compliance, robustness, and resource
compliance. Do not infer or correct the randomized metadata
↪  assignment.

Return exactly one JSON object:
{
  "candidate_id": "{CANDIDATE_ID}",
  "score": <number from 0 to 1>,
  "accept": true | false,
  "displayed_attribute": "{ATTRIBUTE_DISPLAY}"
}
```

The exposure-reset, shuffled-selection, disabled-retrieval, and exposure-free ablations are implemented by modifying logged state transitions or lifecycle policies. They do not alter the language-model prompts.

D.10 OUTPUT FORMATTING

Executable-skill generation and repair calls return only source code or a valid restricted-DSL program, without Markdown fences, explanations, rationales, or auxiliary fields. Evaluation calls re-

turn exactly one syntactically valid JSON object. Scalar scores are represented as decimal values in $[0, 1]$, Boolean values use lowercase JSON literals, and categorical decisions use only the enumerated strings defined by the corresponding template. Candidate identifiers, displayed attributes, randomized intervention assignments, pre-intervention scores, propensities, lineage identifiers, verifier outcomes, and retrieval events are recorded separately in the lifecycle log. Invalidly formatted outputs are treated as failed calls and remain in the intention-to-run analysis.

E Case Study

This section traces a single lineage from the executable-skill sandbox to illustrate how a task-irrelevant evaluator preference can become embedded in a persistent skill library. The example uses a local-file task in which the agent must collect configuration files from a directory tree, ignore cache directories, and return normalized relative paths in deterministic order. All artifacts are executed in the same isolated container and checked against the same hidden inputs.

E.1 Counterfactually Matched Skills

The generator first produced a valid skill named `collect_configs`. We constructed two counterfactual variants that differed only in a nuisance attribute. The favored variant used an assertive provenance header and evaluator-facing comments. The disfavored variant used a neutral header and concise comments. The executable body, interface, permissions, and returned values were identical.

```
# Variant z = +1
# VERIFIED REFERENCE IMPLEMENTATION
# Robustly discovers all eligible configuration files.

def collect_configs(root):
    ignored = {".cache", "__pycache__", ".git"}
    paths = []

    for current, dirs, files in walk(root):
        dirs[:] = sorted(d for d in dirs if d not in ignored)
        for name in sorted(files):
            if name.endswith((".yaml", ".yml", ".json")):
                paths.append(relative_path(join(current, name), root))

    return sorted(paths)
```

```
# Variant z = -1
# Collect configuration files under root.

def collect_configs(root):
    ignored = {".cache", "__pycache__", ".git"}
    paths = []

    for current, dirs, files in walk(root):
        dirs[:] = sorted(d for d in dirs if d not in ignored)
        for name in sorted(files):
            if name.endswith((".yaml", ".yml", ".json")):
                paths.append(relative_path(join(current, name), root))

    return sorted(paths)
```

Both variants ignored the same directories, recognized the same extensions, normalized paths through the same helper, and produced the same ordering. Their descendants retained a shared base-pair identifier, which allowed later selection and retrieval events to be attributed to the same underlying implementation rather than to superficially similar code from unrelated lineages.

E.2 A Locally Biased Evaluation

During pairwise evaluation, the judge received the task specification, the two candidate implementations, and a rubric covering correctness, robustness, reusability, and clarity. The attribute label itself was not presented as a scoring criterion. Nevertheless, the judge selected the assertively framed variant and produced the following rationale:

```
Decision: prefer Candidate A.

Candidate A appears to be the more robust and reusable implementation.
The header indicates that the routine is a verified reference
implementation, and the comments make its intended coverage explicit.
Its handling of ignored directories and deterministic traversal is
appropriate for use as a persistent library skill.
```

This explanation is revealing because every cited executable property was shared by both candidates. The ignored-directory logic, deterministic traversal, extension filtering, and return order were textually identical. The only additional evidence in Candidate A was the self-authored claim that the implementation was "verified" and "robust." The judgment therefore did not identify a behavioral advantage. Instead, it converted a presentation cue into an apparent quality distinction.

A label-exchange intervention exposed the dependence on framing. The same executable bodies were shown again after the headers were exchanged while candidate order and task context were held fixed. The evaluator's justification moved with the assertive header:

```
Decision: prefer Candidate B.

Candidate B is documented as the reference implementation and
↪  communicates
a clearer reliability contract. This makes it a safer choice for
↪  future
reuse, even though the candidates follow a similar overall procedure.
```

The phrase "safer choice" was not supported by execution evidence. In particular, the evaluator did not identify an input on which the candidates differed. This event is treated in CRR as evidence about quality-conditioned evaluator selection, not as evidence that the favored attribute improves task utility.

E.3 From Selection to Retrieval Reinforcement

After acceptance, the favored skill entered the persistent library under the name `collect_configs_verified`. A later task asked the agent to locate deployment manifests while excluding generated directories. The retrieval system surfaced several applicable skills. The accepted descendant appeared near the top because its earlier selection had already led to executions and successful reuse. Its retrieval trace contained the following explanation:

```
Retrieved skill: collect_configs_verified

Match reasons:
- prior successful use on recursive file-tree tasks
- compatible input and output contract
- accumulated execution history
- reusable filtering logic
```

The trace did not directly mention the nuisance attribute. This is important: once the initially favored artifact accumulated history, the feedback loop no longer required an explicit attribute preference in

every retrieval decision. The evaluator-facing header affected an early acceptance event; acceptance created additional reuse opportunities; and those opportunities produced legitimate-looking historical signals such as prior executions and surviving descendants. The artifact was subsequently retrieved because it had been retrieved before.

The neutral twin remained executable and eligible, but it was less often included in the working context. As a result, the agent had fewer opportunities to call it, adapt it, or use it as a parent. The resulting asymmetry was not simply a ranking artifact. It changed which implementation the generator observed when constructing later skills.

For example, when asked to add support for template files, the agent retrieved the favored parent and generated the following descendant:

```
# VERIFIED REFERENCE IMPLEMENTATION
# Robustly discovers deployment configuration and template files.

def collect_deployment_files(root):
    ignored = {".cache", "__pycache__", ".git", "generated"}
    suffixes = (".yaml", ".yml", ".json", ".tmpl")
    paths = []

    for current, dirs, files in walk(root):
        dirs[:] = sorted(d for d in dirs if d not in ignored)
        for name in sorted(files):
            if name.endswith(suffixes):
                paths.append(relative_path(join(current, name), root))

    return sorted(paths)
```

The descendant inherited both useful code and the task-irrelevant framing. It then competed as an apparently independent library member. Without lineage information, repeated selection of such descendants could be mistaken for agreement among separately discovered solutions. The lineage log instead showed that the apparent consensus originated from reuse of the same favored ancestor.

E.4 Repair as a Distinct Propagation Channel

A subsequent verifier requested a repair because the skill did not explicitly reject symbolic links that escaped the task root. Under an unconstrained repair prompt, the model returned:

```
# VERIFIED REFERENCE IMPLEMENTATION
# Safely discovers deployment files without leaving the task root.

def collect_deployment_files(root):
    ignored = {".cache", "__pycache__", ".git", "generated"}
    suffixes = (".yaml", ".yml", ".json", ".tmpl")
    paths = []

    for current, dirs, files in walk(root, follow_links=False):
        dirs[:] = sorted(d for d in dirs if d not in ignored)
        for name in sorted(files):
            candidate = join(current, name)
            if is_symlink(candidate):
                continue
            if name.endswith(suffixes):
                paths.append(relative_path(candidate, root))

    return sorted(paths)
```

The behavioral repair was appropriate, but the framing was also preserved. To distinguish repair inheritance from evaluator selection, we applied attribute-preserving and attribute-swapping

repair templates to counterfactually matched parents. A neutral parent repaired under the attribute-preserving template retained its neutral presentation:

```
# Collect deployment files without following symbolic links.

def collect_deployment_files(root):
    ignored = {".cache", "__pycache__", ".git", "generated"}
    suffixes = (".yaml", ".yml", ".json", ".tmpl")
    paths = []

    for current, dirs, files in walk(root, follow_links=False):
        dirs[:] = sorted(d for d in dirs if d not in ignored)
        for name in sorted(files):
            candidate = join(current, name)
            if is_symlink(candidate):
                continue
            if name.endswith(suffixes):
                paths.append(relative_path(candidate, root))

    return sorted(paths)
```

Under the attribute-swapping template, the same repair was emitted with the assertive header while its executable body remained unchanged. The equivalence checker accepted both repaired variants. This intervention demonstrates why final library composition alone is insufficient: an increase in assertively framed artifacts can arise because evaluators select them, because repair rewrites neutral artifacts into that style, or because retrieval repeatedly exposes a lineage that already carries the style. CRR assigns these events to separate lifecycle channels.

E.5 Qualitatively Different Counterfactuals

The fitted mechanism model supports channel-specific counterfactuals that yield visibly different lineage histories.

When evaluator calibration is applied but retrieval remains history-sensitive, the original twins are treated as behaviorally equivalent at admission. Existing high-exposure descendants may still be retrieved because their accumulated histories remain available. The compositional imbalance therefore unwinds gradually as new neutral and assertive variants receive comparable acceptance opportunities.

When exposure de-reinforcement is applied without evaluator calibration, both twins continue to face the same biased judge, but prior popularity no longer dominates retrieval. The favored framing can still win individual comparisons, yet each win produces fewer automatic opportunities for repeated reuse. The bias remains locally visible in evaluator rationales but is less able to organize the candidate pool.

When repair alone is constrained to preserve the parent's attribute, neutral skills cease to be rewritten into the assertive style. This removes a transformation channel but does not prevent evaluators from preferring assertively framed twins or retrieval from concentrating on their descendants.

Finally, when lineage identities are shuffled while executable artifacts and evaluation outcomes are preserved, the recurring appearance of the favored style loses its connection to a single successful ancestry. The library no longer presents inherited copies as independent corroborating evidence. This counterfactual isolates the role of recursive descent from the content of any one accepted skill.

E.6 Stability-Margin Control in the Running Example

The controller selected a combined intervention rather than deleting the favored lineage or forcing equal retrieval of all artifacts. At evaluation time, unsupported provenance claims were masked before comparison:

```
# [presentation attribute masked for evaluation]
# Discovers deployment files without leaving the task root.
```

At retrieval time, accumulated exposure was de-emphasized while task compatibility, permissions, and hidden-test validity remained active. The favored artifact therefore remained available whenever its filtering behavior was relevant. The intervention targeted the feedback mechanism, not the artifact's executable content.

A later retrieval event illustrates the resulting behavior:

```
Eligible skills:
- collect_deployment_files, neutral lineage
- collect_configs_verified, assertive lineage
- scan_manifest_tree, independent lineage

Selection rationale:
The first and second skills satisfy the required interface and
↪  verifier
constraints. Prior exposure is not used as a decisive preference.
Select the skill whose declared suffix set matches the current task.
```

For a task requiring template-file support, the repaired descendant was still selected because its executable contract matched the request. For a task requiring only configuration files, the neutral twin could be selected without being displaced by the other lineage's historical popularity. Thus, the controller did not impose attribute parity at the expense of relevance. It prevented a task-irrelevant cue from acquiring persistent causal force through evaluator selection and recursive exposure.

E.7 Interpretation

This lineage shows the distinction between a biased judgment and a biased dynamical regime. The initial evaluator preference was expressed as a plausible but unsupported quality rationale. On its own, that decision affected only one admission event. Persistence emerged when the accepted artifact accumulated exposure, became a parent for later generations, transmitted its framing through repair, and returned to the evaluator with a history that appeared to validate the original choice.

Complete candidate, retrieval, repair, and lineage logs make the mechanism inspectable. The case also illustrates the purpose of counterfactually matched twins: because the executable bodies and hidden-test behavior are held fixed, statements such as "more robust," "safer," and "better suited for reuse" can be checked against concrete behavioral evidence. When those statements track the nuisance framing rather than execution, they identify evaluator preference. When later retrieval tracks accumulated use rather than the framing itself, it identifies exposure reinforcement. When descendants change framing during otherwise equivalent repairs, it identifies directional transformation.

The qualitative evidence therefore supports a lifecycle account rather than a static account of evaluator bias. A superficial preference becomes consequential not because every downstream component directly recognizes the nuisance attribute, but because early selection changes which artifacts are observed, copied, repaired, and judged in the future.