

SKILLFLOW: FLOW-DRIVEN RECURSIVE SKILL EVOLUTION FOR AGENTIC ORCHESTRATION

Anonymous authors

Paper under double-blind review

ABSTRACT

In recent years, a variety of powerful LLM-based agentic systems have been applied to automate complex tasks through task orchestration. However, existing orchestration methods still face key challenges, including strategy collapse under reward maximization, high gradient variance with opaque credit assignment, and unguided skill evolution whose decisions are typically made by directly prompting an LLM to judge rather than derived from principled training signals. To address these challenges, we propose SkillFlow, a flow-based framework that takes a trainable Supervisor as the agent and a structured environment with dynamic skill library and frozen executor, automating task orchestration through multi-turn interaction. SkillFlow employs Tempered Trajectory Balance (TTB), a regression-based objective for normalized action and backward policies whose full-support zero-residual solution samples terminal orchestration strategies in proportion to tempered reward. The same flow objective yields a jointly learned backward policy that exposes per-step flow diagnostics without additional policy evaluations beyond those required by TTB. Building on these flow diagnostics, a recursive skill evolution mechanism determines *when* to evolve, *what* skills to create or prune, and *where* decision gaps lie—closing the loop from training signal to autonomous skill curation. Experimental results on 14 datasets show that SkillFlow outperforms baselines across question answering, mathematical reasoning, code generation, and real-world interactive decision making tasks. Our code is available at <https://anonymous.4open.science/r/SkillFlow-E850>.

1 INTRODUCTION

In recent years, a variety of powerful LLM-based agentic systems have been applied to solve a wide range of complex tasks (Yao et al., 2022b; Hong et al., 2023; Wang et al., 2024b; Dang et al., 2025), gradually moving beyond single-turn question answering toward executable end-to-end task completion.

In this process, task orchestration has become a key bridge from task goals to reproducible execution: by organizing primitive actions and reusable skills into structured trajectories on an action-level orchestration DAG (Fig. 1, left), where each node is an orchestration state and each edge is one orchestration action, agents can complete complex tasks with improved controllability, compositionality, and reusability (Zeng et al., 2024; Qian et al., 2024). However, in practice, orchestration still heavily relies on manual skill design and rigid action definitions (Hu et al., 2024; Xu & Yan, 2026), making it costly to transfer across new tasks, new skill configurations, or different agent and executor capabilities, each redone by hand.

To address these issues, early approaches rely on heuristic orchestration (Fig. 2a), retrieving skills from prebuilt libraries or searching over workflow graphs at test time, e.g., matching a corresponding tool to each subtask by description, or using tree search to explore multi-step action compositions (Wang

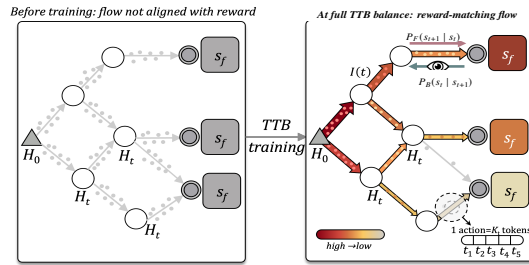


Figure 1: SkillFlow at a glance. **Left:** schematic unaligned initialization. **Right:** reward-matching flow at TTB balance (colormap = flow magnitude). H_t renders canonical s_t , **inset** K_t tokens/action.

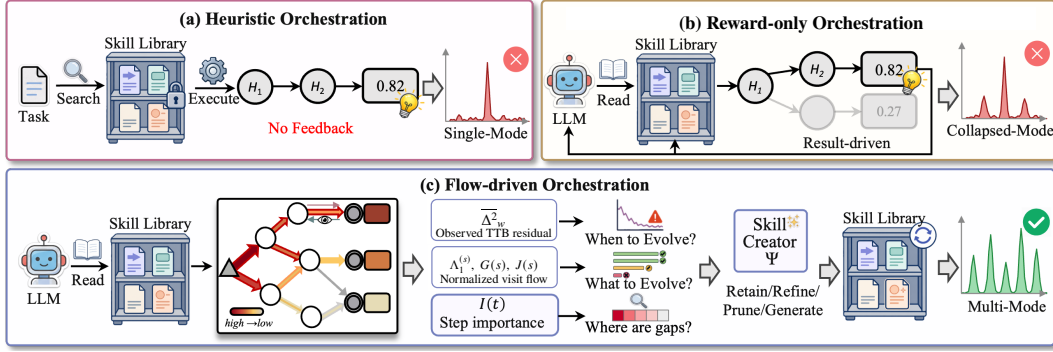


Figure 2: Three orchestration paradigms. (a) heuristic dispatch over a locked library; (b) a schematic reward-only learner with concentrated sampling; (c) SkillFlow co-trains normalized forward and backward policies and curates skills from observed residuals and invocation-flow diagnostics.

et al., 2023; 2024a; Zhang et al., 2024; Zhuge et al., 2024), but their orchestration policies remain bound to pre-designed heuristics and fixed skill repertoires, with no mechanism to adapt from execution outcomes. Recent work has shifted to *learning-based orchestration* (Fig. 2b) (Zeng et al., 2024; Chen et al., 2023; Shao et al., 2024; Guo et al., 2025; Li et al., 2025b; Zhang et al., 2026b; Kong et al., 2026; Wang et al., 2025b; Xia et al., 2026), which trains orchestration policies directly from task-completion signals—sampling orchestration trajectories, collecting terminal rewards such as answer correctness or task success rate, and updating the orchestration model accordingly.

However, these methods still face three challenges. **(i) Strategy collapse.** Terminal-reward maximization does not itself specify how probability should be shared among equally effective strategies; without an explicit distributional target, learning can concentrate on a narrow set of high-reward routes (Yu et al., 2025; Guo et al., 2025; Zhang et al., 2026b; Kong et al., 2026; Wang et al., 2025b). This brittleness is amplified when executor capabilities evolve—once a tool is upgraded, the agent has no suitable fallback strategy (Zhang et al., 2023; Li et al., 2025a). **(ii) High gradient variance and opaque credit assignment.** Terminal-only rewards create long credit-assignment horizons, where policy-gradient variance and intra-group advantage collapse leave it unclear which step of a succeeding or failing trajectory is responsible (Schulman et al., 2015; Tan et al., 2026; Wang et al., 2026c; Li et al., 2025b; Wang et al., 2025b). **(iii) Unguided skill evolution.** Existing frameworks with dynamic skill libraries rely on heuristic triggers, fixed schedules, or direct LLM-as-judge prompting, lacking a principled signal for *when* to update the library, *what* skills to prune or create, or *where* the critical decision points lie—so the library evolves blindly (Wang et al., 2023; Xia et al., 2026; Fang et al., 2025; Xu & Yan, 2026; Alzubi et al., 2026; Zhang et al., 2026a; Wang et al., 2025a; 2026a).

To address these challenges, we propose **SkillFlow** (Fig. 2c), a flow-based framework (Bengio et al., 2021; 2023; Malkin et al., 2022) for general task orchestration with recursive skill evolution, in which a trainable Supervisor interacts with a structured environment containing a dynamic skill library and a frozen executor. At the core of SkillFlow (Fig. 1) is **Tempered Trajectory Balance (TTB)**, a regression-style objective whose full-support zero-residual solution matches *terminal-strategy probability to tempered reward*. On a sufficient quotient DAG, normalized action policies select complete decisions and a normalized backward policy allocates each strategy’s mass among its admissible orderings. This target gives positive mass to supported alternatives rather than prescribing one best path, under the conditions stated in §4.2. The same loss jointly trains a *backward policy* giving flow-based per-step credit from quantities already evaluated for TTB. A recursive skill evolution mechanism then answers three questions from that signal alone: *when* to evolve (an observed TTB plateau), *what* to create or prune (normalized invocation-flow statistics), and *where* candidate decision gaps lie (the step importance $I(t)$)—closing the loop from training signal to autonomous skill curation. We label these **C1** (normalized reward-proportional strategy modeling), **C2** (per-step flow diagnostics from the hindsight backward) and **C3** (flow-driven library curation), and ablate each in §5.

We evaluate on 14 benchmarks over QA, math, code and interactive decision making. SkillFlow outperforms direct LLMs and REINFORCE-style RL baselines (Li et al., 2025b; Zhang et al., 2026b; Xia et al., 2026) in task accuracy and in-framework RL objectives in strategy diversity, at the lowest per-call cost among skill-evolution methods (Jiang et al., 2026a; Yang et al., 2026b; Ma et al., 2026; Alzubi et al., 2026)—a foundation for self-evolving, flow-based agentic orchestration.

2 RELATED WORK

Agent Task Orchestration. LLM-era orchestration has moved from rule-based automation to feedback-driven plan-act-feedback loops (Yao et al., 2022b; Wang et al., 2024a;b), spanning single-agent sequential decision making (Qin et al., 2023; Yao et al., 2022b), LLM-controller routing and constrained API planning (Ong et al., 2024; Wang et al., 2024a; Su et al., 2026), and multi-agent SOP/role collaboration (Hong et al., 2023; Wu et al., 2024; Dang et al., 2025). Reusable skill memory (Wang et al., 2023; Xu & Yan, 2026; Jiang et al., 2026b; Wang et al., 2026a) and self-evolving architectures (Fang et al., 2025; Li, 2026; Alzubi et al., 2026; Zhang et al., 2026a; Wang et al., 2025a; 2026b) further improve efficiency. SkillFlow couples orchestration learning from execution feedback with flow-guided evolution of its skill repertoire.

Reinforcement Learning for Agents. Agent RL treats multi-turn interaction as a long-horizon MDP, with credit-assignment advances in implicit step rewards, hindsight advantage estimation, plan-execute decomposition, progressive reward shaping and process reward models (Zhou et al., 2023; Chen et al., 2025; Liu et al., 2025; Tan et al., 2026; Peng et al., 2026; Su et al., 2025; Zhang et al., 2025; She et al., 2025). Policy optimization has converged on GRPO-style objectives and extensions such as DAPO, VAPO and multi-agent variants (Guo et al., 2025; Shao et al., 2024; Yu et al., 2025; Yue et al., 2025; Liu et al., 2026; Cang et al., 2026), also applied to workflow orchestration and search-augmented reasoning (Zhang et al., 2026b; Li et al., 2025b; Jin et al., 2025; Yang et al., 2026a); whether probability concentrates or remains distributed follows from the reward objective and regularizer, not from REINFORCE-family optimization itself (Li et al., 2025a). Reward-matching flow training has been explored in LLM reasoning (Yu et al., 2024) and robust scheduling (Zhang et al., 2023); SkillFlow targets a different setting, multi-turn agentic orchestration with a co-evolving skill library, and couples a terminal reward-matching distribution with backward-derived per-step diagnostics that are reused, without extra policy evaluations, for library evolution.

3 PRELIMINARIES

Definition 1: Orchestration State Graph. We model task orchestration as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ whose vertices are sufficient orchestration states $s_t = \sigma(H_t)$ and whose edges advance one action. Under the action-DAG convention below, $s_t = [a_1 \cdots a_t]_{\equiv \mathcal{T}}$, identifying action sequences only by swaps that preserve the canonical state, subsequent dynamics and terminal reward (Appendix E.1). Distinct, independently enabled `skill` actions can then give a state several parents, while every edge increases the action count, so $\mathcal{G}$ is acyclic. The logged history $H_t = H_{t-1} \oplus (r_t, a_t, o_t^{\text{exec}})$ reaches the models only through its canonical rendering.

Definition 2: Orchestration Trajectory. A complete path from $s_0 = \sigma(H_0)$ to a terminal state is an orchestration trajectory $\tau = \{(r_t, a_t, o_t^{\text{exec}})\}_{t=1}^T \Rightarrow y_q$, where r_t is a reasoning reflection, $a_t = (\alpha_t, o_t)$ has type $\alpha_t \in \{\text{skill}, \text{act}, \text{accept}\}$ and parameters o_t , and o_t^{exec} is execution feedback. The episode terminates upon `accept` or at $T_{\max}$; a horizon termination is also terminal.

Support and Action-DAG Convention. Fix a query q and training phase k . At every non-terminal state, a nonempty finite set $\Sigma(s)$ contains the legal, uniquely serialized complete actions, including an `accept` action, and both this candidate rule and the library stay fixed within the phase. The action-only model adds a deterministic reasoning rule $r = f^{(k)}(s)$ and transition $s' = \delta^{(k)}(s, a)$, each defined on the sufficient canonical state before the action is chosen (Appendix B). Section 4.1 and Appendix B.4 give the complete probability model when reasoning or execution is stochastic; query and phase indices are suppressed when unambiguous.

Problem Statement. Given task $q \in \mathcal{D}$, environment $\mathcal{E}$ and executor $\mathcal{M}_{\text{exec}}$, we seek a conservative flow with terminal values $F_q(x) = \tilde{R}(x)^\beta$, distinguishing a terminal strategy from a particular path reaching it:

$$\pi^*(x | q) = \frac{\tilde{R}(x)^\beta}{Z(q)}, \quad \pi^*(\tau | q) = \frac{\tilde{R}(x)^\beta P_B(\tau | x)}{Z(q)}, \quad x = s_T. \quad (1)$$

Here $P_B(\tau | x)$ is the normalized reverse-path probability, $Z(q) = \sum_{x \in \mathcal{X}_q} \tilde{R}(x)^\beta$, $R(x) \in [0, 1]$ is constant on each terminal equivalence class, $\tilde{R}(x) = R(x) + \varepsilon_{\min} > 0$, and $\beta > 0$ sets the

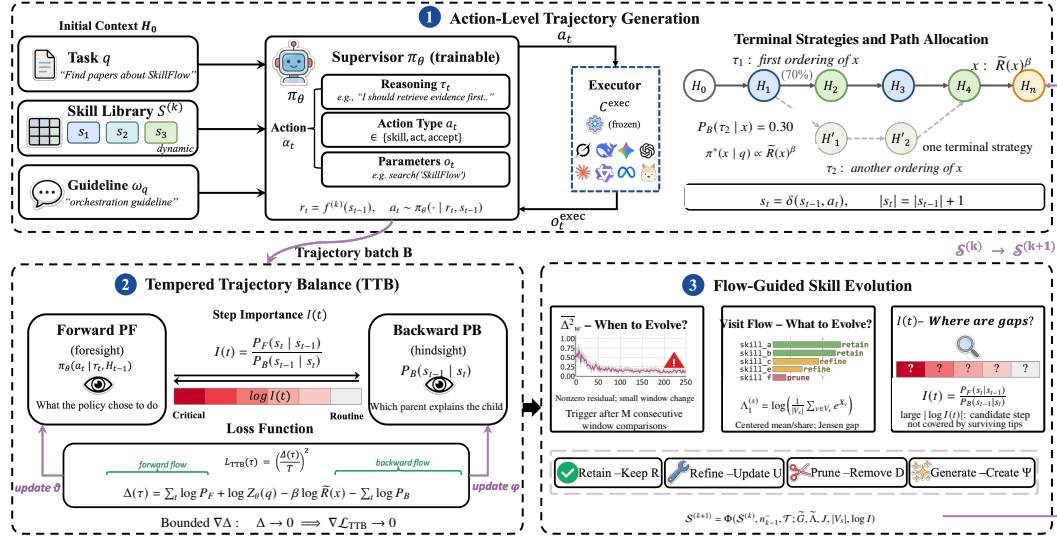


Figure 3: SkillFlow architecture. The Supervisor acts on sufficient canonical states against a fixed Executor (§3), and TTB trains normalized forward and fixed-child backward policies. At zero residual it matches terminal-strategy rewards, the upper-right paths showing two orderings of one terminal with conditional weights. Residuals, invocation-flow summaries and step ratios drive curation.

diversity–quality tradeoff. On a quotient DAG $P_B(\tau | x)$ allocates a strategy’s mass among orderings, the target of entropy-regularized RL with P_B -induced intermediate rewards (Appendix C).

4 METHODOLOGY: SKILLFLOW

Figure 3 illustrates the framework: environment design and task modeling (§4.1), flow-based end-to-end training (§4.2), and flow-driven recursive skill evolution (§4.3).

4.1 ENVIRONMENT DESIGN AND TASK MODELING

SkillFlow follows a Supervisor–Executor paradigm (Yao et al., 2022b): a trainable Supervisor π_θ builds trajectories in the structured environment $\mathcal{E}$. Unlike static frameworks over a fixed skill set, its skill library is dynamic and co-evolves with the policy through the curation operator Φ (§4.3).

Structured Environment $\mathcal{E}$. The environment is $\mathcal{E} = (\mathcal{S}, \Psi, \mathcal{M}_{\text{exec}})$, where $\mathcal{S}$ is the dynamic skill library, Ψ creates or refines tips using trajectory evidence, and $\mathcal{M}_{\text{exec}}$ is the pluggable executor. Ψ updates the library only at phase boundaries, between completed episodes.

State Space and Orchestration Target. The initial context is $H_0 = [q \oplus \mathcal{S}_{\text{ret}} \oplus \omega_q]$, where $\mathcal{S}_{\text{ret}}$ contains retrieved skills and ω_q is a task-category guideline. The Supervisor, reasoning rule and Executor read deterministic renderings of the same sufficient state $s = \sigma(H)$. Activating a skill increments its count in a canonically ordered multiset of versioned tips, so repeated invocations stay distinct occurrences. A swap is admitted only when availability, execution feedback, retained context and reward agree in both orders (Appendix E.1); otherwise the histories remain separate states.

Step-wise Orchestration Policy. For the general stochastic model, let ρ_θ be the normalized reasoning distribution and $\pi_\theta(a | r, s)$ the normalized complete-action distribution of §4.2, with joint policy

$$\pi_\theta^{\text{joint}}(r, a | s) = \rho_\theta(r | s) \pi_\theta(a | r, s), \quad \pi_\theta(a | r, s) = \pi_\theta(\alpha | r, s) \pi_\theta(o | \alpha, r, s). \quad (2)$$

Its type and parameter factors are the induced marginals and conditionals; under the action-DAG convention of §3, $\rho_\theta(\cdot | s)$ collapses to the point mass at the state rule $f^{(k)}(s)$.

Multi-Turn Interaction and Trajectory Distribution. The Executor returns $o_t^{\text{exec}} \sim \mathcal{C}_{\text{exec}}(\cdot | s_{t-1}, a_t)$ and $s_t = U(s_{t-1}, r_t, a_t, o_t^{\text{exec}})$, so a recorded transcript has joint probability

$$P_\theta^{\text{joint}}(\tau | q) = \prod_{t=1}^T \rho_\theta(r_t | s_{t-1}) \pi_\theta(a_t | r_t, s_{t-1}) \mathcal{C}_{\text{exec}}(o_t^{\text{exec}} | s_{t-1}, a_t). \quad (3)$$

A frozen Executor has fixed parameters but need not be deterministic, so on a sufficient state space the graph kernel is the marginal $P_F(s' \mid s) = \sum_{r,a,o} \rho_\theta(r \mid s) \pi_\theta(a \mid r, s) \mathcal{C}_{\text{exec}}(o \mid s, a) \mathbf{1}\{U(s, r, a, o) = s'\}$, normalized whenever outcomes and reasoning that distinguish future behaviour are retained in s . The action-only convention reduces it to $P_F(\delta(s, a) \mid s) = \pi_\theta(a \mid f^{(k)}(s), s)$; Appendix B.4 gives the expanded-graph form and the chance-realizability condition.

Proposition 1. *Under the conditions above, the action quotient is a finite pointed DAG admitting conservative reward-matching flows, with non-trivial parent sets whenever a legal commuting pair yields distinct predecessors. Realizing a specified flow under stochastic execution additionally requires the forward kernel to respect the fixed chance transitions.*

Formal result. Appendix E. Empirical support. §5, RQ1 and RQ3. $\square$

4.2 FLOW-BASED END-TO-END TRAINING

Building on Proposition 1, we train normalized policies toward the terminal-strategy target of Eq. (1).

Forward Policy P_F and Backward Policy P_ϕ . Let $\ell_\theta(a; r, s)$ be the mean token log-likelihood of the complete action a , including its end marker. This length-averaged score is a positive weight, normalized over *all* actions in $\Sigma(s)$ before sampling. For a fixed child x , $b_\phi(x, p)$ scores each legal parent p from canonical child and candidate-parent information:

$$\pi_\theta(a \mid r, s) = \frac{e^{\ell_\theta(a; r, s)}}{C_F(s, r)}, \quad C_F(s, r) = \sum_{a' \in \Sigma(s)} e^{\ell_\theta(a'; r, s)}, \quad P_B(p \mid x) = \frac{e^{b_\phi(x, p)}}{\sum_{p' \in \text{Pa}(x)} e^{b_\phi(x, p')}}. \quad (4)$$

In the action-only model, P_F is this categorical action sampler with $r = f^{(k)}(s)$. $P_\phi(a_t \mid s_t)$ abbreviates $P_B(s_{t-1} \mid s_t)$ when deleting the last maximal occurrence identifies the parent. The backward denominator is evaluated for the same child over its entire legal parent set (Proposition 28). In the stochastic model, P_F is the marginal kernel of §4.1, with P_B normalized on that graph.

Tempered Trajectory Balance (TTB). The tempered **Trajectory Balance** condition (Bengio et al., 2023) requires the following identity for every supported complete trajectory:

$$\log Z_\theta(q) + \sum_{t=1}^T \log P_F(s_t \mid s_{t-1}) = \beta \log \tilde{R}(s_T) + \sum_{t=1}^T \log P_B(s_{t-1} \mid s_t). \quad (5)$$

The **TTB loss** is the squared, length-normalized residual (Dall’Antonia et al., 2026):

$$\Delta(\tau) := \log Z_\theta(q) + \sum_{t=1}^T \log P_F(s_t \mid s_{t-1}) - \beta \log \tilde{R}(s_T) - \sum_{t=1}^T \log P_B(s_{t-1} \mid s_t),$$

$$\mathcal{L}_{\text{TTB}}(\tau) = (\Delta(\tau)/T)^2. \quad (6)$$

Both local log-normalizers in Eq. (4) remain inside Δ (Appendix B.3). If $\Delta = 0$ on every supported complete path, summing over paths ending at x gives $P_F(x \mid q) = \tilde{R}(x)^\beta / Z_\theta(q)$. Zero expected loss under a full-support trajectory measure is sufficient for this condition on the finite graph, and its attainability depends on the policy class and on the chance constraints under stochastic execution. The factor $T^{-2} > 0$ changes loss weighting but not the zero-residual solution set.

Step Importance and Skill Marginal Flow. The normalized forward/backward ratio at an edge defines the **step importance** of that edge at any training iterate:

$$I(t) := \frac{P_F(s_t \mid s_{t-1})}{P_B(s_{t-1} \mid s_t)}; \quad I(t) = \frac{F_q(s_t)}{F_q(s_{t-1})} \quad \text{at exact balance.} \quad (7)$$

At balance, $\log I(t)$ is the increment of state log-flow, which aggregates terminal rewards by backward hitting probabilities (Lemma 18), so a local ratio reflects downstream outcomes. Away from balance, they give a path-indexed prefix estimate and an invocation-conditioned **skill-flow summary**:

$$\hat{F}(s) = \frac{1}{|\mathcal{B}_s|} \sum_{\tau \in \mathcal{B}_s} \sum_{t: a_t \text{ invokes } s} \hat{f}_{\tau, t}, \quad \log \hat{f}_{\tau, t} = \log Z_\theta(q_\tau) + \sum_{t'=1}^t \log I(t'). \quad (8)$$

Here $\mathcal{B}_s$ contains the trajectories invoking s , repeats included. At exact balance $\hat{f}_{\tau, t} = F_{q_\tau}(s_t)$; otherwise it stays path-indexed. $\hat{F}(s)$ measures flow at invocation sites rather than invocation frequency. Large $|\log I(t)|$ flags candidate decision gaps from quantities TTB already evaluates.

Proposition 2. *On the supported sufficient-state DAG, normalized kernels with zero TTB residual on every complete path induce reward-proportional terminal sampling and a state-flow decomposition by $I(t)$. Prefix and skill-flow diagnostics reuse the forward and backward quantities that TTB has already evaluated, at any iterate. Under a bounded residual gradient, the per-trajectory TTB regression gradients have variance tending to zero whenever the mean squared residual tends to zero.*

Formal result. Appendix I. Empirical support. §5, RQ1–RQ2, RQ4 and RQ5. $\square$

4.3 FLOW-DRIVEN RECURSIVE SKILL EVOLUTION

Prior work on skill distillation and refinement (Wang et al., 2023; Alzubi et al., 2026; Zhang et al., 2026a; Wang et al., 2026b; Xia et al., 2026) relies on heuristic schedules or LLM-as-judge prompting. SkillFlow instead couples *when*, *what* and *where* to the flow quantities of Section 4.2: observed TTB residuals set the trigger, and invocation-flow summaries and step ratios guide the edits.

When: TTB Residual Plateau. For a fixed full-support trajectory measure $\mu^{(k)}$ on the phase- k support, the representational residual floor over the trainable parameters (θ, ϕ, Z_θ) is

$$\bar{\Delta}^{*(k)} := \inf_{\theta, \phi, Z_\theta} \mathbb{E}_{\tau \sim \mu^{(k)}} [\Delta(\tau \mid \mathcal{S}^{(k)}, \theta, \phi, Z_\theta)^2] \geq 0. \quad (9)$$

An attained zero is equivalent to a representable exact TB solution. The practical trigger compares consecutive running-residual windows (Appendix F.1) rather than estimating this infimum: a sustained, nonzero plateau initiates the next curation phase at the following episode boundary.

What and Where: Flow-Guided Skill Curation via a CGF. Each skill $s \in \mathcal{S}$ is an *atomic tip*, a bounded, self-contained guidance fragment with a read-only interface. With visits $V_s = \{(\tau, t) : a_t \text{ invokes } s\} \neq \emptyset$ and $X_v = \log \hat{f}_{\tau, t} - \log Z_\theta(q_\tau)$, its **per-skill cumulant generating function** is

$$\Lambda_\lambda^{(s)} := \log \left(\frac{1}{|V_s|} \sum_{v \in V_s} e^{\lambda X_v} \right), \quad \lambda \in \mathbb{R}.$$

Normalizing over visits makes $\Lambda_0^{(s)} = 0$ even with repeated calls, and query-specific normalizers stay inside the visit statistics. Curation reads the **mean log-flow share** $G(s) = \partial_\lambda \Lambda_\lambda^{(s)}|_0$, the **centered log-flow share** $\tilde{\Lambda}(s) = \Lambda_1^{(s)} - \mathbb{E}_{s'}[\Lambda_1^{(s')}]$, the centered mean $\tilde{G}(s) = G(s) - \mathbb{E}_{s'}[G(s')]$ and the Jensen gap $J(s) = \Lambda_1^{(s)} - G(s) \geq 0$, with library averages uniform over sufficiently supported skills:

$$\mathcal{S}^{(k+1)} = \Phi(\mathcal{S}^{(k)}, n_{k-1}^-, \mathcal{T}; \{\tilde{G}(s), \tilde{\Lambda}(s), J(s)\}_{s \in \mathcal{S}_{\text{sup}}^{(k)}}, \{|V_s|\}_{s \in \mathcal{S}^{(k)}}, \{\log I_\tau(t)\}_{\tau, t}). \quad (10)$$

Here n_{k-1}^- stores the previous negative-share counters, $\mathcal{T}$ is the development trajectory buffer, and $\mathcal{S}_{\text{sup}}^{(k)}$ collects the sufficiently supported tips, the only ones for which centered statistics are formed; unsupported tips need only their visit counts and are held over. It holds over under-supported tips, retains supported tips with high *relative* mean and small gap, refines the other survivors, prunes persistently negative relative-share tips, and asks Ψ for tips at high- $|\log I(t)|$ positions in same-query success/failure pairs. $J(s)$ is a dispersion statistic, with its exact tilted-variance identity and small-spread form $J(s) = \frac{1}{2} \text{Var}(X) + O(\text{range}(X)^3)$ proved in Appendix D.5.

Proposition 3. *The phase-boundary curation operator preserves atomic composability when retained and accepted generated tips satisfy the atomic interface, and under the state-sufficiency and support rules each resulting phase again has a pointed DAG. Reward-matching flow conservation in a phase follows from its own exact TB condition, not from a plateau or a library edit.*

Formal result. Appendix J. Empirical support. §5, RQ4 and RQ5. $\square$

5 EXPERIMENTS

Datasets. 14 benchmarks over QA, math, code and interactive decision making—IID (7, *training*): HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), MedQA (Jin et al., 2021), AIME (2026 held out), WebShop (Yao et al., 2022a), ALFWorld (Shridhar et al., 2020), SWE-bench (Jimenez et al., 2023). OOD (7, *held out*): MuSiQue (Trivedi et al., 2022), NQ-Open, MATH-Hard, GPQA Diamond, HumanEval (Chen et al., 2021), ScienceWorld, Mind2Web (Appendix M).

		Baseline		SFT	GRPO [†]	AFlow	Agent+RL			Ours
Dataset	Metric	Qwen3.5	v4-flash	Qwen3.5	Qwen3.5	Qwen3.5	AgentFlow	FlowSteer	SkillRL	SkillFlow (Δ ↑)
(a) In-Distribution (IID) benchmarks										
HotpotQA	Ans EM	60.78±0.65	72.34±1.05	64.69±1.02	69.38±0.35	88.59±0.70	88.44±0.86	89.84±0.78	87.66±0.65	92.03±1.16 (+31.3)
	Ans F1	75.54±0.84	86.11±0.61	79.08±0.75	81.83±0.48	91.08±0.38	89.80±0.79	91.36±0.60	89.32±0.32	93.79±0.66 (+18.3)
TriviaQA	Ans EM	44.84±0.70	74.38±0.35	47.03±0.35	46.72±0.86	92.81±0.86	90.16±0.70	91.25±1.28	89.06±0.00	96.25±0.86 (+51.4)
	Ans F1	54.46±1.27	81.83±0.69	55.89±0.84	56.77±1.06	93.09±1.19	90.31±0.61	92.22±0.83	90.47±0.66	96.63±0.53 (+42.2)
AIME 2026	Acc.	47.33±1.49	48.67±1.83	37.33±2.79	32.67±1.49	53.33±2.36	60.00±0.00	62.00±2.98	56.00±2.79	71.33±1.83 (+24.0)
MedQA	Acc.	69.53±1.24	83.44±0.86	73.59±0.65	77.34±0.78	89.53±0.43	87.34±0.86	90.94±0.43	85.78±0.86	92.03±0.86 (+22.5)
WebShop	Avg Score	56.62±0.70	85.17±0.78	50.96±1.28	53.94±0.35	70.93±0.86	83.77±0.65	85.27±1.02	88.12±0.65	94.57±0.86 (+38.0)
	SR	32.34±0.70	67.97±0.96	35.00±0.86	36.88±0.86	59.38±0.55	74.06±1.28	78.91±0.00	82.97±0.35	93.44±0.70 (+61.1)
ALFWorld	SR	48.75±1.31	60.78±0.65	40.94±0.70	51.09±0.70	75.16±0.65	80.78±0.70	82.66±0.65	85.47±0.70	96.25±0.86 (+47.5)
SWE-bench	Resolved	17.34±0.86	38.12±0.86	16.09±0.43	18.44±0.89	30.16±0.70	41.56±0.65	43.59±1.28	39.38±0.70	52.03±0.70 (+34.7)
Avg. (IID)	Ans EM	52.81±0.48	73.36±0.55	55.86±0.54	58.05±0.46	90.70±0.55	89.30±0.55	90.55±0.75	88.36±0.33	94.14±0.72 (+41.3)
	Ans F1	65.00±0.76	83.97±0.46	67.48±0.56	69.30±0.58	92.09±0.62	90.06±0.50	91.79±0.51	89.89±0.37	95.21±0.42 (+30.2)
	Acc./Pass	45.32±0.45	64.02±0.43	42.32±0.56	45.06±0.37	63.08±0.46	71.25±0.32	73.89±0.58	72.95±0.53	83.28±0.43 (+38.0)
(b) Out-of-Distribution (OOD) benchmarks										
MuSiQue	Ans EM	39.22±0.86	51.09±1.05	39.38±0.70	42.19±0.78	77.81±1.18	80.00±0.43	79.84±1.28	75.47±0.43	84.84±1.05 (+45.6)
	Ans F1	47.48±0.43	59.32±0.96	48.21±0.71	50.14±0.76	84.22±0.89	84.59±0.71	85.53±0.77	81.38±0.92	88.98±0.74 (+41.5)
NQ-Open	Ans EM	21.72±0.35	37.34±0.65	23.75±0.70	22.03±0.86	75.78±0.00	78.91±0.78	80.31±0.86	79.22±1.18	82.50±0.70 (+60.8)
	Ans F1	30.03±0.58	48.21±0.42	30.66±0.70	29.15±0.49	80.39±0.82	83.28±0.88	84.88±0.59	82.78±0.78	86.20±0.30 (+56.2)
MATH-Hard	Acc.	89.38±0.70	91.72±0.70	88.59±0.43	87.19±0.70	90.78±0.65	92.03±0.35	93.59±0.65	91.09±0.43	96.09±0.55 (+6.7)
GPQA Diamond	Acc.	61.88±0.86	73.59±0.86	71.56±0.43	74.69±0.70	75.62±1.28	81.25±0.78	84.38±0.00	78.12±1.10	90.16±0.43 (+28.3)
HumanEval	pass@1	88.91±0.86	97.19±1.18	80.62±0.86	85.78±0.35	92.81±0.86	93.59±0.86	93.91±0.65	92.34±0.65	98.75±1.05 (+9.8)
ScienceWorld	Success	25.94±0.65	43.91±0.65	15.31±1.31	24.53±1.18	29.84±0.86	44.84±0.70	47.81±0.86	38.91±0.65	57.66±1.28 (+31.7)
	Avg Score	41.86±0.78	58.95±0.43	34.85±0.65	40.16±0.86	43.49±0.65	56.69±0.86	59.07±0.86	50.42±0.70	67.87±0.78 (+26.0)
Mind2Web	Step Acc	26.80±0.86	27.41±0.86	19.05±1.05	23.51±0.78	30.78±0.70	40.15±0.70	41.10±0.70	35.16±0.78	51.25±0.43 (+24.5)
	Action F1	63.69±0.89	65.90±1.14	50.33±0.33	67.29±0.71	62.13±0.33	71.93±1.04	75.53±0.70	69.54±0.72	84.58±0.43 (+20.9)
Avg. (OOD)	EM	30.47±0.46	44.22±0.62	31.56±0.49	32.11±0.58	76.80±0.59	79.45±0.45	80.08±0.77	77.34±0.63	83.67±0.63 (+53.2)
	F1	47.07±0.38	57.81±0.52	43.07±0.35	48.80±0.38	75.58±0.42	79.93±0.51	81.98±0.40	77.90±0.47	86.58±0.30 (+39.5)
	Acc./Pass	55.79±0.32	65.46±0.33	51.67±0.35	55.98±0.33	60.56±0.35	68.09±0.30	69.98±0.28	64.34±0.31	76.96±0.33 (+21.2)

Table 1: Main results on 14 IID and OOD benchmarks. Every cell is the mean over five runs; subscripts give the standard deviation, propagated to the Avg. rows. $\Delta \uparrow$ is the absolute point gain over Qwen3.5-9B. GRPO[†] fine-tunes the backbone directly on terminal rewards; GRPO[‡] (Fig. 5a) keeps the full SkillFlow architecture and swaps only the objective. Metric definitions: Appendix O.

Setup and baselines. The Supervisor is Qwen3.5-9B trained with LoRA for 250 steps (7 questions $\times$ 4 trajectories per update, $T_{\max} = 12$) against a frozen Qwen3.5-9B Executor, with GPT-5.5 as Skill Creator Ψ ; every benchmark is scored on 128 instances (30 for AIME) by its official harness; Table 1 averages five runs (Appendices O, P, L). Baselines (Appendix N): the direct LLMs Qwen3.5-9B and v4-flash; SFT and GRPO[†] (Shao et al., 2024) on the same backbone; the search-based AFlow (Zhang et al., 2024); and the Agent+RL methods AgentFlow (Li et al., 2025b), FlowSteer (Zhang et al., 2026b) and SkillRL (Xia et al., 2026). All trained methods share backbone, Executor, an empty initial library, tools, rollout budget and optimiser (Appendix L).

Main results and OOD generalization (RQ1, RQ2). SkillFlow leads on all 14 benchmarks (Table 1) and beats the far stronger direct LLM v4-flash: the gain comes from how the orchestration policy is trained, not from model capacity—nor from the skill generator, which Appendix F.9 swaps for an open-weight model while keeping the average lead. Margins are largest on the interactive tasks WebShop, ALFWorld, ScienceWorld and Mind2Web, where REINFORCE-style objectives collapse to a single strategy (Fig. 5d) and AFlow’s static workflow runs out of moves. The OOD results support cross-benchmark transfer of the evolved skills: with the library frozen, SkillFlow keeps its lead on the held-out benchmarks (Avg.(OOD) F1 4.60 points above FlowSteer).

Executor transferability (RQ3). Figure 4 plugs the trained Supervisor, unchanged, into six third-party executors (a: IID; c: OOD by task category). Every executor improves, so the Supervisor learned orchestration rather than one executor’s habits; the weakest gain most, because per-step credit and diversity-preserving sampling compensate for noisier feedback, and the strongest still gain on agent-style tasks, where orchestration, not execution, is the bottleneck.

Variant	IID							OOD						
	Hotpot	Trivia	AIME	MedQA	WShop	ALFW	SWE	MuSQ	NQ	MATH	GPQA	HEval	SciW	M2W
	EM	EM	Acc	Acc	SR	SR	Res	EM	EM	Acc	Acc	pass@1	Succ	Step Acc
– TTB	83.59	87.50	53.33	85.16	82.03	82.81	40.63	73.44	79.69	87.50	79.69	89.06	39.06	33.59
– Backward policy	85.94	90.63	53.33	86.72	80.47	88.28	43.75	75.00	78.91	89.06	82.81	90.63	42.19	36.72
– $\hat{L}_{TTB}$ for <i>when</i>	82.81	84.38	60.00	89.06	82.81	85.94	46.88	75.78	71.88	90.63	85.16	92.19	48.44	40.63
– $I(t)$ for <i>where</i>	85.16	88.28	63.33	89.06	87.50	90.63	44.53	78.13	75.00	88.28	82.81	93.75	47.66	42.97
– $\hat{F}(s)$ for <i>what</i>	84.38	82.03	50.00	83.59	79.69	80.47	42.19	76.56	75.78	87.50	80.47	90.63	45.31	42.19
– Skills	81.25	78.91	43.33	78.13	77.34	77.34	32.81	71.09	71.88	89.06	80.47	87.50	42.97	40.63
+ Random skills	89.84	94.53	60.00	89.84	85.94	86.72	46.09	82.81	82.81	91.41	85.16	92.97	49.22	44.53
SkillFlow (Full)	92.19	96.09	70.00	92.19	93.75	96.09	52.34	85.16	82.81	96.09	89.84	98.44	57.81	51.56

Table 2: Component ablation across IID and OOD benchmarks (RQ4, run 1). –TTB replaces TTB with the in-framework GRPO[‡] (C1); –Backward policy removes the hindsight P_ϕ (C2); the remaining rows ablate skill evolution (C3)—the *when/where/what* flow signals one at a time, –Skills removing the library altogether, and +Random skills a same-size library of skills sampled at random from Qwen3.8-27B in place of the curated ones.

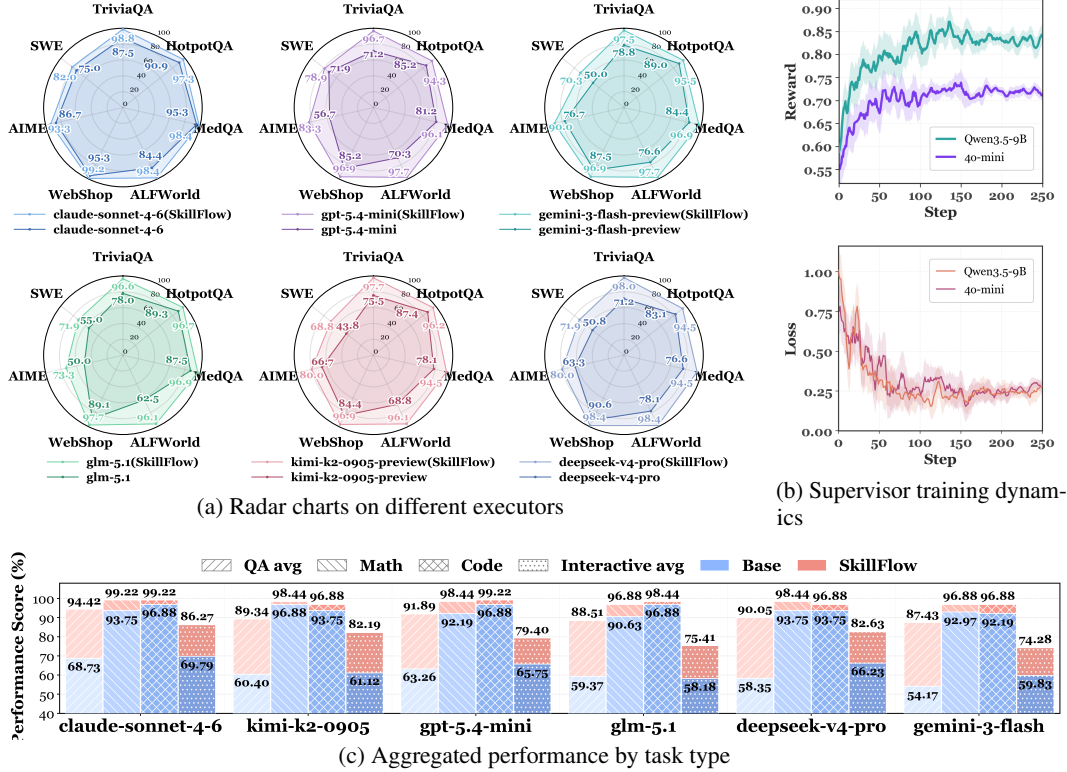


Figure 4: Executor transferability. The Supervisor is always Qwen3.5-9B and is the only trained component; every Executor is frozen. (a) Per-executor radar over the seven IID benchmarks for six third-party LLMs, with vs. without SkillFlow. (b) Supervisor training dynamics with Qwen3.5-9B vs. 40-mini as the frozen Executor. (c) Aggregated gain by task category on the OOD benchmarks.

Component ablation (RQ4). Table 2 removes the components behind C1–C3 one at a time. C1: swapping TTB for the in-framework GRPO[‡] hurts most where several solution routes exist (ScienceWorld, Mind2Web, AIME: –17 to –19): TTB’s value is keeping multiple high-reward paths alive. C2: removing the backward policy P_ϕ costs multi-step tasks about three times what it costs factoid QA (AIME –16.7 vs. TriviaQA –5.5): the more decisions, the more per-step credit matters. C3: dropping any one of *when/where/what* lowers every benchmark; removing the library costs 7–27 points, and refilling it with as many random skills recovers only about 60%—what matters is not how many skills the library holds but which steps the flow signals turned into skills, and when.

(i) IID benchmarks

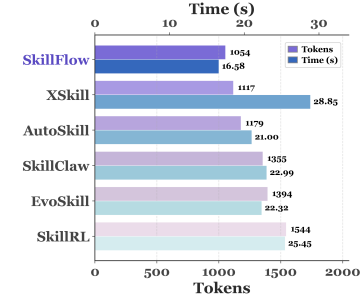
Method	Hotpot EM	Trivia EM	AIME Acc	MedQA Acc	WShop SR	ALFW SR	SWE Res
Qwen3.5	60.94	44.53	46.67	69.53	32.03	48.44	17.19
GRPO [‡]	83.59	87.50	53.33	85.16	82.03	82.81	40.63
Tree-GRPO	87.50	93.75	60.00	88.28	85.94	90.63	48.44
HCAPO	85.94	93.75	63.33	90.63	87.50	92.19	46.88
SkillFlow (Ours)	92.19	96.09	70.00	92.19	93.75	96.09	52.34

(ii) OOD benchmarks

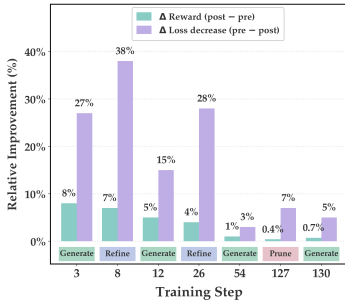
Method	MuSQ Ans EM	NQ Ans EM	MATH Acc	GPQA Acc	HEval pass@1	SciW Succ	M2W Step Acc
Qwen3.5	39.06	21.88	89.06	61.72	89.06	25.78	26.64
GRPO [‡]	73.44	79.69	87.50	79.69	89.06	39.06	33.59
Tree-GRPO	77.34	80.47	92.19	80.47	92.19	42.19	36.00
HCAPO	79.69	81.25	93.75	84.38	93.75	43.75	39.10
SkillFlow (Ours)	85.16	82.81	96.09	89.84	98.44	57.81	51.56

(a) Algorithm comparison on IID and OOD benchmarks (RQ5)

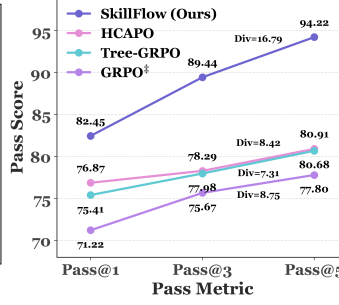
Method	GPU-h	s/turn	s/traj
GRPO [‡]	267	1.79	35.6
HCAPO	537	1.72	48.1
Tree-GRPO	424	1.99	45.1
Ours	292	1.13	33.9



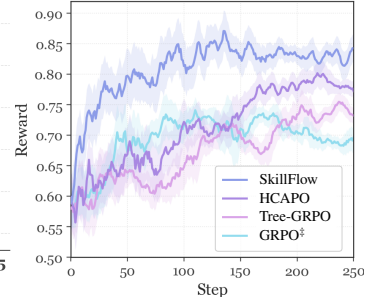
(b) Training & per-call cost



(c) Evolution events



(d) Pass@K



(e) Reward curves

Figure 5: Mechanism analysis and algorithm comparison. (a) Objective-level comparison (GRPO[‡]: in-framework GRPO). (b) Top: training cost of the objectives in (a), as total GPU-hours (GPU-h; Appendix P.4) and mean rollout seconds per model call (s/turn) and per trajectory (s/traj); bottom: per-call cost vs five skill-evolution baselines (Jiang et al., 2026a; Yang et al., 2026b; Ma et al., 2026; Alzubi et al., 2026; Xia et al., 2026). (c) Skill-evolution events by curation class (§4.3): *relative* change in reward and TTB loss around each event (*Retain* produces none). (d) Pass@K and strategy diversity *Div*, the number of *structurally distinct* successful workflows (distinct step arrangements) per query; both pooled over all 14 benchmarks (Appendix O). (e) Reward curves.

Algorithm comparison and cost (RQ5). With backbone, data and architecture fixed and only the objective (plus Tree-GRPO’s branching) swapped, SkillFlow still beats GRPO[‡], Tree-GRPO (Ji et al., 2025) and HCAPO (Tan et al., 2026) on all 14 benchmarks (Fig. 5a). Fig. 5(d,e) shows why: the REINFORCE-style objectives concentrate probability on one mode, so extra samples repeat one path and Pass@K rises only 4–7 points from K=1 to 5; SkillFlow yields 1.9× as many distinct successful workflows (Div 16.79 vs. 7.3–8.8) and gains 12 points—diversity turns into success under repeated attempts. Evolution events fire at TTB plateaus and each precedes a reward gain (Fig. 5c). On cost (Fig. 5b), training takes 9% more GPU-hours than GRPO[‡] but fewer than Tree-GRPO and HCAPO; per call, SkillFlow uses the fewest tokens and least time of six skill-evolution methods, ≈32%/35% below SkillRL: flow signals, not extra LLM judgments, trigger evolution.

6 CONCLUSION

SkillFlow unifies orchestration training and recursive skill evolution under TTB and beats direct-LLM and RL baselines in accuracy, RL objectives in diversity, and skill-evolution methods in per-call cost. Its normalized flow formulation links a conditional terminal-sampling guarantee to reusable prefix diagnostics and a phase-boundary curation rule, all read off quantities TTB already evaluates. It answers one question: can flow-based training drive recursive skill evolution from end-to-end task feedback alone, under the state-sufficiency and realizability conditions of §4.2? Orchestration amplifies what a backbone knows rather than adding new reasoning (Appendix Q).

AI USE STATEMENT

We have not used generative AI tools for generating synthetic data, developing theoretical models or conceptual frameworks, formulating mathematical claims, contributing to or writing proofs, proposing or refining hypotheses, designing the methodology or experiments, implementing methods, or interpreting results; translation, dataset cleaning and reformatting, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to aid and polish the writing, and for retrieval and discovery of related work. Large language models also appear as components of the studied system, namely the frozen Executor and the Skill Creator Ψ (§4); this role is part of the method rather than of the research workflow. We have reviewed all AI-assisted work: the authors checked every polished passage against its original meaning and read every retrieved paper before citing it. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work follows the principles of the ICLR Code of Ethics. We aim to conduct and report our research with scientific integrity, transparency, and reproducibility. Experimental results are reported without fabrication, falsification, or intentional misrepresentation, and sufficient implementation and evaluation details are provided to facilitate verification and reproduction. We appropriately acknowledge prior work and use existing benchmarks, datasets, and associated resources in accordance with their intended research purposes and applicable licenses. This study does not involve human participants or the collection of personal information.

REPRODUCIBILITY STATEMENT

Every trained method in the paper runs from the same pinned artefacts and budget, except that Tree-GRPO keeps its published branching; the shared configuration (backbone, frozen Executor, empty initial library, rollout and update budgets, LoRA and optimizer settings, hardware) and the per-benchmark evaluation harnesses are tabulated in Appendix L, the evolution hyperparameters in Appendix F.7, and the datasets, splits and OOD hygiene in Appendices M and L. The TTB objective, the backward-policy parameterisation and every proposition used in the main text are derived and proved in Appendices A–K. Compute and wall-clock costs, including the per-step time breakdown, are reported in Appendix P. Source code, configuration files and the evaluation harnesses are available anonymously at <https://anonymous.4open.science/r/SkillFlow-E850>.

REFERENCES

- Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.
- Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.
- Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.
- Yueyang Cang, Xiaoteng Zhang, Erlu Zhao, Zehua Ji, Yuhang Liu, Yuchen He, Zhiyuan Ning, Chen Yijun, Wenge Que, and Li Shi. Graph-grpo: Stabilizing multi-agent topology learning via group relative policy optimization. *arXiv preprint arXiv:2603.02701*, 2026.
- Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.
- Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025.

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Pedro Dall’Antonia, Tiago da Silva, Daniel Csillag, Salem Lahlou, and Diego Mesquita. Avoid what you know: Divergent trajectory balance for gflownets. *arXiv preprint arXiv:2602.17827*, 2026.
- Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *arXiv preprint arXiv:2505.19591*, 2025.
- Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.
- Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.
- Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.
- Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R Fung. Xskill: Continual learning from experience and skills in multimodal agents. *arXiv preprint arXiv:2603.12056*, 2026a.
- Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills—beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The twelfth international conference on learning representations*, 2023.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Serkan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.
- Mingze Kong, Zikun Qu, Zhongquan Zhou, Pengyu Liang, Xiang Li, Zhiwei Shang, Zhi Hong, Kaiyu Huang, Zhiyong Wang, and Zhongxiang Dai. Workflow-r1: Group sub-sequence policy optimization for multi-turn workflow construction. *arXiv preprint arXiv:2602.01202*, 2026.
- Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, et al. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *arXiv preprint arXiv:2509.07430*, 2025a.
- Xiaoxiao Li. When single-agent with skills replace multi-agent systems and when they fail. *arXiv preprint arXiv:2601.04748*, 2026.

- Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025b.
- Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32150–32158, 2026.
- Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Junge Zhang, and Jianbin Jiao. Agentic reinforcement learning with implicit step rewards. *arXiv preprint arXiv:2509.19199*, 2025.
- Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.
- Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.
- Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *arXiv preprint arXiv:2602.16165*, 2026.
- Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*, 2024.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The twelfth international conference on learning representations*, 2023.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. R-prm: Reasoning-driven process reward modeling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13449–13462, 2025.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.
- Jianghao Su, Xia Zeng, Luhui Liu, Chao Luo, Ye Chen, and Zhuoran Zhuang. Enhancing agentic rl with progressive reward shaping and value-based sampling policy optimization. *arXiv preprint arXiv:2512.07478*, 2025.
- Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, and Lewei He. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026*, pp. 2060–2070, 2026.
- Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.
- Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, et al. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*, 2026a.
- 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*, 2023.
- Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026b.
- Jiong Xiao 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*, 2025a.
- Xiangwei Wang, Wei Wang, Ken Chen, Nanduni Nimalsiri, and Saman Halgamuge. Discovering process-outcome credit in multi-step llm reasoning. *arXiv preprint arXiv:2602.01034*, 2026c.
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024a.
- Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024b.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.
- Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.
- Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.
- Matthew YR Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in llm reasoning. *arXiv preprint arXiv:2601.14209*, 2026a.
- Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, et al. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

- 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*, 2022b.
- Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. Flow of reasoning: Training llms for divergent reasoning with minimal examples. *arXiv preprint arXiv:2406.05673*, 2024.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Yu Yue, Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024.
- David W Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with gflownets. *arXiv preprint arXiv:2302.05446*, 2023.
- Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Evoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026a.
- Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2024.
- Mingda Zhang, Haoran Luo, Tiesunlong Shen, Qika Lin, Xiaoying Tang, Rui Mao, and Erik Cambria. Flowsteer: Interactive agentic workflow orchestration via end-to-end reinforcement learning. *arXiv preprint arXiv:2602.01664*, 2026b.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.
- 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.
- Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

APPENDIX

Appendices A–K give the theoretical treatment of SkillFlow’s flow-based learning framework, following the progressive development style of GFlowNet Foundations (Bengio et al., JMLR 2023): the foundations of flow networks on DAGs, the derivation of the Tempered Trajectory Balance (TTB) loss, its zero-residual sampling and gradient properties, and all supporting results, each section building on the previous ones with complete proofs. Appendices L–Q give the experimental material: the comparison protocol, datasets, baselines, evaluation metrics, computational cost, and limitations.

A FLOW-THEORETIC FOUNDATIONS

This section introduces the fundamental concepts of flow networks on directed acyclic graphs (DAGs) and establishes the connection to reward-proportional sampling.

A.1 FLOW NETWORKS ON DAGS

We begin with the basic definitions of flow networks.

Definition 1 (Flow Network). *A flow network is a tuple $(\mathcal{G}, F)$ where $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a finite DAG with a unique source s_0 and terminal set $\mathcal{X}$ consisting of its sinks, and $F : \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$ assigns an edge flow. The source is non-terminal, every vertex is reachable from it and can reach a terminal, and internal vertices satisfy the conservation law below. These conditions hold for the finite-horizon supported orchestration graph.*

For each state $s \in \mathcal{V}$, we denote by $\text{Pa}(s)$ the set of predecessors (parents) and $\text{Ch}(s)$ the set of successors (children). We now introduce the crucial concept of state flow.

Definition 2 (State Flow and Flow Conservation). *For every internal state $s \in \mathcal{V} \setminus (\mathcal{X} \cup \{s_0\})$, the state flow satisfies*

$$F(s) := \sum_{s' \in \text{Ch}(s)} F(s \rightarrow s') = \sum_{p \in \text{Pa}(s)} F(p \rightarrow s). \quad (11)$$

At the source, define $F(s_0)$ by the outgoing sum only; at a terminal x , define $F(x)$ by the incoming sum only. Thus conservation equates incoming and outgoing flow at internal vertices, not at the source or sinks.

Remark 1. *Flow conservation is the defining property that relates edge flows to state flows. By Definition 2, either sum computes $F(s)$ at an internal vertex; source and terminal flows use their respective boundary sums.*

The partition function represents the total flow circulating through the network.

Definition 3 (Partition Function and Total Flow). *The partition function Z is the total flow in the network, equal to the source flow:*

$$Z := F(s_0) = \sum_{s' \in \text{Ch}(s_0)} F(s_0 \rightarrow s') = \sum_{x \in \mathcal{X}} F(x). \quad (12)$$

The last equality (terminal flow equals source flow) follows from recursive application of flow conservation at each state: every unit of flow leaving the source must traverse some path and arrive at some terminal.

We now introduce policies derived from flow normalization.

Definition 4 (Forward Policy and Backward Policy). *On positive-flow states, we define:*

Forward policy: $P_F(s' | s) = \frac{F(s \rightarrow s')}{F(s)}$ for each edge $s \rightarrow s'$ where s is non-terminal.

Backward policy: $P_B(s | s') = \frac{F(s \rightarrow s')}{F(s')}$ for each edge $s \rightarrow s'$ where $s' \in \mathcal{V} \setminus \{\text{source}\}$.

By the appropriate internal or boundary sums, both kernels are normalized on their domains: $\sum_{s'} P_F(s' | s) = 1$ and $\sum_s P_B(s | s') = 1$. At a zero-flow state a kernel can be assigned arbitrarily on its legal support without affecting positive-flow sampling; logarithmic TB below uses strictly positive kernels.

We now define trajectory flow, which is essential for understanding how the forward policy generates trajectories.

Definition 5 (Trajectory and Trajectory Flow). *A complete trajectory is a path $\tau = (s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_T = x)$ from the source s_0 to some terminal $x \in \mathcal{X}$. The trajectory flow is:*

$$F(\tau) := Z \cdot \prod_{t=1}^T P_F(s_t | s_{t-1}). \quad (13)$$

Lemma 4 (Trajectory Flow Identity). *For kernels induced by a common flow, on paths whose visited states have strictly positive flow,*

$$F(\tau) = Z \prod_{t=1}^T \frac{F(s_{t-1} \rightarrow s_t)}{F(s_{t-1})} = F(s_T) \prod_{t=1}^T P_B(s_{t-1} | s_t).$$

In contrast, the ratio-product telescope on state flows is

$$Z \prod_{t=1}^T \frac{F(s_t)}{F(s_{t-1})} = F(s_T). \quad (14)$$

Thus the state-flow telescope is not generally the trajectory flow or the product of unnormalized edge flows. It equals $F(\tau)$ exactly when the reverse probability of that path is one, in particular on a positive-flow tree.

Proof. Substitute Definition 4 into the trajectory product. Multiplying and dividing by consecutive state flows telescopes to $F(s_T)/F(s_0)$, and $F(s_0) = Z$. The remaining reverse edge product is $P_B(\tau | s_T)$, proving both identities. $\square$

We now establish the key decomposition of state flow.

Proposition 5 (Flow Decomposition). *For any state $s \in \mathcal{V}$, the state flow equals the sum of flows of all trajectories passing through s :*

$$F(s) = \sum_{\tau \ni s} F(\tau), \quad (15)$$

where $\{\tau \ni s\}$ denotes all complete trajectories passing through state s . In particular, for terminal states:

$$F(x) = \sum_{\tau: s_T=x} F(\tau), \quad (16)$$

i.e., the flow into a terminal equals the sum of flows of trajectories terminating at that state.

Proof. If $Z = 0$, nonnegativity and conservation imply that all edge and state flows vanish, so the claim is immediate. Hence assume $Z > 0$. Let $h_F(s) = P_F(\text{visit } s)$ for the forward walk. Its finite acyclic support guarantees termination, and no state can be revisited. We first prove $h_F(s) = F(s)/Z$ by induction in topological order.

Base case: $h_F(s_0) = 1 = F(s_0)/Z$.

Inductive step: every visit to $s \neq s_0$ has one final incoming edge. The events of traversing its different incoming edges are disjoint, and the Markov kernel gives

$$h_F(s) = \sum_{p \in \text{Pa}(s)} h_F(p) P_F(s | p) \quad (17)$$

$$= \frac{1}{Z} \sum_{p \in \text{Pa}(s)} F(p) P_F(s | p) = \frac{1}{Z} \sum_{p \in \text{Pa}(s)} F(p \rightarrow s) = \frac{F(s)}{Z}. \quad (18)$$

For each prefix reaching s , the probabilities of all legal suffixes from s sum to one. Consequently $h_F(s) = \sum_{\tau \ni s} P_F(\tau)$, where the sum is over complete paths, not prefixes extended as if they were complete. Multiplying by Z and using Definition 5 yields

$$F(s) = \sum_{\tau \ni s} Z P_F(\tau) = \sum_{\tau \ni s} F(\tau).$$

For a terminal x , visiting it means terminating there, which gives the terminal decomposition. $\square$

A.2 REWARD-MATCHING AND GFLowNET SAMPLING

We now establish when a flow induces sampling proportional to a reward function.

Definition 6 (Reward-Matching Flow). *Given a strictly positive reward function $R : \mathcal{X} \rightarrow \mathbb{R}_{>0}$ on terminal states, a flow F is **reward-matching** if:*

$$F(x) = R(x) \quad \text{for all } x \in \mathcal{X}. \quad (19)$$

That is, the flow into each terminal state equals the reward of that terminal.

The central theorem of GFlowNets states that reward-matching flows induce a distribution over terminals proportional to rewards.

Theorem 6 (GFlowNet Sampling Property). *If F is reward-matching with respect to $R : \mathcal{X} \rightarrow \mathbb{R}_{>0}$, then sampling a complete trajectory by following the forward policy P_F produces a distribution over terminals given by:*

$$P_F(x) := \Pr_{\tau \sim P_F} [s_T = x] = \frac{R(x)}{Z}, \quad (20)$$

where $Z = \sum_{x' \in \mathcal{X}} R(x')$ is the partition function (total reward).

Proof. Fix a terminal x . The probability of sampling a trajectory ending at x is:

$$P_F(x) = \sum_{\tau: s_T=x} \prod_{t=1}^{|\tau|} P_F(s_t \mid s_{t-1}). \quad (21)$$

Substituting Equation (13):

$$P_F(x) = \sum_{\tau: s_T=x} \frac{F(\tau)}{Z}. \quad (22)$$

By Proposition 5, $F(x) = \sum_{\tau: s_T=x} F(\tau)$, so:

$$P_F(x) = \frac{F(x)}{Z}. \quad (23)$$

Since F is reward-matching, $F(x) = R(x)$, thus:

$$P_F(x) = \frac{R(x)}{Z} = \frac{R(x)}{\sum_{x'} R(x')}. \quad (24)$$

□

Remark 2 (Positive Rewards for Logarithmic TB). *The terminal sampling identity also permits nonnegative rewards with positive total reward: a zero-reward terminal then receives zero flow. Strict positivity is used here to retain all supported terminals and make logarithmic TB and positive edge kernels well defined. SkillFlow uses $R \in [0, 1]$ and the additive smoothing in Appendix G for this purpose.*

A.3 APPLICATION TO TASK ORCHESTRATION

We now instantiate the abstract flow network framework to SkillFlow’s task orchestration setting.

Proposition 7 (Flow Network Instantiation for Task Orchestration). *Under the action-DAG convention of Section 3, orchestration has the following flow-network structure:*

- **Vertices:** *sufficient canonical states $s_t = \sigma(H_t) = [a_1 \cdots a_t]_{\equiv_{\mathcal{T}}}$ on the supported legal language.*
- **Edges:** *a legal complete action $a_t = (\alpha_t, o_t)$ advances the state by the deterministic transition δ .*
- **Source:** *the empty trace with initial task, retrieved skills, and orchestration guideline.*
- **Terminals:** *accept states and horizon terminations, with a reward constant on each equivalence class.*

- **Forward kernel:** the normalized complete-action sampler $P_F(\delta(s, a) \mid s) = \pi_\theta(a \mid f^{(k)}(s), s)$.
- **Edge flow:** for any normalized forward kernel and source value $Z > 0$, $F(s \rightarrow s') = Z P_F(\text{visit } s) P_F(s' \mid s)$ is conservative at internal states.
- **Reward matching:** $F(x) = \tilde{R}(x)^\beta$ is the boundary objective, obtained when the actual normalized forward/backward kernels satisfy full TB, not from graph structure alone.

The key property is acyclicity, established by the strictly increasing action count.

Lemma 8 (DAG Acyclicity by Action Growth). *Under $s_t = \sigma(H_t) = [a_1 \cdots a_t]_{\equiv_{\mathcal{T}}}$, each edge appends exactly one action and hence strictly increases the action count: $|s_t| = |s_{t-1}| + 1$. The count is well defined on equivalence classes because $\equiv_{\mathcal{T}}$ permutes letters without deleting them. Therefore the orchestration graph is acyclic: no path can visit the same state twice, as that would require returning to a state of smaller or equal size. Unlike the unquotiented history graph, however, a state may be reached from several parents (Proposition 28), so $\mathcal{G}$ is a DAG that may have non-trivial merges.*

This acyclicity is essential for applying flow matching algorithms, which require the DAG structure.

B TRAJECTORY BALANCE AND TTb DERIVATION

This section derives the Trajectory Balance (TB) condition and the Tempered Trajectory Balance (TTB) loss, the core objective for SkillFlow training.

Convention (DAG edge = action; node = sufficient state). We first use the finite action-DAG convention of Section 3. An edge appends one complete action, although its score is computed autoregressively over its tokens. Length averaging occurs *before* normalization over complete actions; it does not change the graph to token-level granularity. Each serialization carries an end marker and at most $K_{\max}$ tokens, and legal traces are closed under the admitted swaps. Reasoning logs are not reintroduced in an order-dependent form: the state rules are causal, not a conditioning on a future realized transcript. All vertices are reachable from the source and can reach a terminal, all sampling targets are over the terminals reachable on the specified support, and forward and backward kernels are positive on every legal edge. All formulas are phase- and query-specific. The stochastic extension in Section B.4 uses the actual marginal kernel or explicitly represented reasoning and chance edges.

B.1 BACKWARD POLICY DEFINITION

The backward policy is the reverse-direction distribution derived from flows.

Definition 7 (Backward Policy). *Given a positive flow F , the backward policy is*

$$P_B(s \mid s') = \frac{F(s \rightarrow s')}{F(s')}, \quad s \in \text{Pa}(s'). \quad (25)$$

For each fixed non-source child s' , incoming-flow normalization gives $\sum_{s \in \text{Pa}(s')} P_B(s \mid s') = F(s')/F(s') = 1$.

For learning, P_ϕ is parameterized independently of P_F . Its relationship to a common flow is a consequence of full TB, not an assumption about a partially trained model.

Remark 3 (SkillFlow’s Hindsight-Conditioned Backward). *For a fixed child x , enumerate every legal parent $p \in \text{Pa}(x)$. A deterministic context $h(x, p)$ combines the canonical child with that candidate parent; the scorer $b_\phi(x, p)$ may use the post-action evidence retained in x :*

$$P_B(p \mid x) = \frac{\exp b_\phi(x, p)}{C_B(x)}, \quad C_B(x) := \sum_{p' \in \text{Pa}(x)} \exp b_\phi(x, p'). \quad (26)$$

Every component of this vector is a function of the same fixed child and its candidate parent. In the trace quotient, deleting a maximal occurrence identifies its parent, so $P_\phi(a \mid x)$ is shorthand for the corresponding component (Proposition 28). A different arrival history does not change the vector. The hindsight signal compares forward choice from the parent with reverse attribution from the child; it is not a probability of task success.

B.2 TRAJECTORY BALANCE CONDITION AND FULL DERIVATION

Trajectory Balance characterizes when normalized kernels describe a common reward-matching flow, including when the kernels have been parameterized separately.

Theorem 9 (Trajectory Balance (Malkin et al., 2022)). *On a finite pointed DAG, let P_F and P_B be strictly positive kernels normalized over legal children and parents, respectively, let $R(x) > 0$ on all terminals, and let $Z > 0$. There exists a positive reward-matching flow inducing these kernels with source value Z if and only if every complete path $\tau = (s_0, \dots, s_T)$ satisfies*

$$Z \prod_{t=1}^T P_F(s_t | s_{t-1}) = R(s_T) \prod_{t=1}^T P_B(s_{t-1} | s_t). \quad (27)$$

Equivalently,

$$\log Z + \sum_{t=1}^T \log P_F(s_t | s_{t-1}) = \log R(s_T) + \sum_{t=1}^T \log P_B(s_{t-1} | s_t). \quad (28)$$

In this case $Z = \sum_x R(x)$ and $P_F(x) = R(x)/Z$.

Proof. We prove both directions explicitly.

Forward direction (F reward-matching $\Rightarrow$ TB holds). Expand the two sides using the policies induced by F :

$$\text{LHS} = Z \prod_{t=1}^T \frac{F(s_{t-1} \rightarrow s_t)}{F(s_{t-1})}, \quad (29)$$

$$\text{RHS} = R(s_T) \prod_{t=1}^T \frac{F(s_{t-1} \rightarrow s_t)}{F(s_t)}. \quad (30)$$

Their ratio is

$$\frac{\text{LHS}}{\text{RHS}} = \frac{Z}{R(s_T)} \prod_{t=1}^T \frac{F(s_t)}{F(s_{t-1})} = \frac{Z}{R(s_T)} \frac{F(s_T)}{F(s_0)} = 1, \quad (31)$$

because $F(s_0) = Z$ and $F(s_T) = R(s_T)$.

Reverse direction (TB holds $\Rightarrow$ a common reward-matching flow). First sum Eq. (27) over paths ending at a fixed terminal x :

$$Z \sum_{\tau \rightarrow x} P_F(\tau) = R(x) \sum_{\tau \rightarrow x} P_B(\tau | x) = R(x). \quad (32)$$

The reverse-path sum is one: a backward walk on a finite DAG strictly decreases a topological rank, and the only state with no parent is the source. Forward walks likewise terminate, so summing over x gives $Z = \sum_x R(x)$ and $P_F(x) = R(x)/Z$.

It remains to establish interior flow consistency rather than assuming it. For a prefix $\gamma : s_0 \rightarrow s$, set

$$F_\gamma(s) := Z \prod_{e=(u,v) \in \gamma} \frac{P_F(v | u)}{P_B(u | v)}. \quad (33)$$

Take any two prefixes γ_1, γ_2 ending at s and append the same legal suffix $\eta : s \rightarrow x$. Applying TB to both complete paths yields

$$F_{\gamma_i}(s) \prod_{e \in \eta} \frac{P_F(e)}{P_B(e)} = R(x), \quad i = 1, 2.$$

Positivity makes the suffix product nonzero, hence $F_{\gamma_1}(s) = F_{\gamma_2}(s)$. Write this well-defined value as $F(s)$ and define $F(s \rightarrow s') = F(s)P_F(s' | s)$. Extending a prefix by one edge gives

$$F(s)P_F(s' | s) = F(s')P_B(s | s'). \quad (34)$$

Summing over children gives outgoing flow $F(s)$; summing over parents gives incoming flow $F(s)$ at every non-source state. Also $F(s_0) = Z$ and $F(x) = R(x)$ by TB. Thus the constructed flow is conservative at internal states, reward-matching, and induces both specified kernels. $\square$

B.3 TTB LOSS DERIVATION WITH TEMPERATURE AND LENGTH NORMALIZATION

The loss uses two distinct length operations: a mean token score parameterizes a complete-action weight, and T^{-2} weights the trajectory residual. Only the normalized complete-action probabilities enter TB.

Definition 8 (Per-Token Scores and Normalized Edge Probabilities). *Let a have $K(a)$ tokens, including its end marker, and let p_θ denote the autoregressive token scorer. Define*

$$\ell_\theta(a; r, s) := \frac{1}{K(a)} \sum_{j=1}^{K(a)} \log p_\theta(\text{tok}_j \mid \text{tok}_{<j}, r, \text{can}(s)), \quad (35)$$

$$b_\phi(x, p) := \frac{1}{K(a_{p,x})} \sum_{j=1}^{K(a_{p,x})} \log p_\phi(\text{tok}_j(a_{p,x}) \mid \text{tok}_{<j}(a_{p,x}), h(x, p)). \quad (36)$$

Here $a_{p,x}$ is the action occurrence whose deletion from the trace x yields p . For a richer stochastic state graph, a scorer of a uniquely serialized legal predecessor may be used instead. With $C_F(s, r) = \sum_{a \in \Sigma(s)} e^{\ell_\theta(a; r, s)}$ and $C_B(x) = \sum_{p \in \text{Pa}(x)} e^{b_\phi(x, p)}$, the edge log-probabilities in the action-only model are

$$\begin{aligned} \log P_F(\delta(s, a) \mid s) &= \ell_\theta(a; f^{(k)}(s), s) - \log C_F(s, f^{(k)}(s)), \\ \log P_B(p \mid x) &= b_\phi(x, p) - \log C_B(x). \end{aligned} \quad (37)$$

Finite candidate supports and finite token scores make both denominators positive and finite. Sampling selects a complete action according to this categorical distribution. If several latent events lead to the same state edge, their probabilities are summed as in Eq. (42).

Definition 9 (TTB Residual). *For $r_t = f^{(k)}(s_{t-1})$ and $x = s_T$, substituting Eq. (37) into TB gives*

$$\begin{aligned} \Delta(\tau) &= \log Z_\theta(q) - \beta \log \tilde{R}(x) \\ &+ \sum_{t=1}^T [\ell_\theta(a_t; r_t, s_{t-1}) - \log C_F(s_{t-1}, r_t) - b_\phi(s_t, s_{t-1}) + \log C_B(s_t)]. \end{aligned} \quad (38)$$

$Z_\theta(q) > 0$ is a query-conditioned partition estimate for the phase. For the stochastic model, replace each forward term by the log of Eq. (42), or use all edges of the expanded graph of Lemma 13. No reasoning or chance probability is removed merely because its parameters are frozen.

At zero residual on every complete path, Theorem 9 applies with terminal reward $\tilde{R}(x)^\beta$. In particular, the path law is $P_F(\tau \mid q) = \tilde{R}(x)^\beta P_B(\tau \mid x) / Z_\theta(q)$, while the terminal law has no multiplicity factor.

Definition 10 (Tempered Trajectory Balance Loss). *The TTB loss is*

$$\mathcal{L}_{TTB}(\tau) = \left(\frac{\Delta(\tau)}{T} \right)^2, \quad 1 \leq T \leq T_{\max}. \quad (39)$$

Expectations used to characterize the zero set are taken under a measure μ assigning positive mass to every supported complete path. For the expanded graph, the residual sums over every subedge, while the positive outer weight may still use the number of action turns T .

Lemma 10 (Length Normalization Property). *On the finite support, $\mathbb{E}_\mu \mathcal{L}_{TTB} = 0$ if and only if $\Delta(\tau) = 0$ for every complete path. If the absolute edge log-ratio is bounded by B and $|\log Z_\theta(q) - \beta \log \tilde{R}(x)| \leq A$, then $|\Delta(\tau)/T| \leq B + A/T$. Per-token averaging alone does not supply such uniform bounds.*

Proof. Each summand $\mu(\tau) \Delta(\tau)^2 / T^2$ is nonnegative with strictly positive coefficient. Their sum vanishes exactly when every residual vanishes. The bound follows from the triangle inequality on the T edge log-ratios. Thus the weighting controls cumulative length effects under the stated bounds without changing the target. $\square$

Proposition 11 (TTB Self-Annealing Property). *For a recorded trajectory, with its discrete states, support, state-rule outputs, and T held fixed during differentiation,*

$$\nabla_\theta \mathcal{L}_{TTB}(\tau) = \frac{2\Delta(\tau)}{T^2} \nabla_\theta \Delta(\tau). \quad (40)$$

If the residual gradient stays bounded, the per-trajectory gradient tends to zero as the residual tends to zero. This identity is not a convergence theorem for the optimizer.

Proof. Differentiate Δ^2/T^2 by the chain rule; T is fixed for the recorded sample. The convergence implication follows by multiplying the bounded gradient by $2|\Delta|/T^2$. Appendix H.1 expands the derivatives of both local normalizers. $\square$

Lemma 12 (Temperature Effect on Policy). *For a fixed finite terminal support, the exact terminal target is $\pi_\beta^*(x) = \tilde{R}(x)^\beta / \sum_y \tilde{R}(y)^\beta$. As $\beta \rightarrow 0^+$ it becomes uniform over terminals; as $\beta \rightarrow \infty$ it becomes uniform over the reward-maximizing terminals, possibly more than one. Conditional on x , paths have law $P_B(\tau | x)$.*

Proof. For the first limit, each positive $\tilde{R}(x)^\beta$ tends to one. For the second, divide numerator and denominator by $\max_y \tilde{R}(y)^\beta$: submaximal terms vanish and each maximizer contributes one. The conditional path law follows by dividing TB by its terminal marginal. No limit on raw path uniformity follows unless the backward path distributions have the required form. $\square$

B.4 CONDITIONAL TB, HINDSIGHT BACKWARD, AND PER-TOKEN TEMPERING

We now make explicit the probability conditions behind the action-only formulation, the fixed-child backward policy, and normalized length-tempered action sampling.

(i) Causal reasoning rules and stochastic transitions.

Lemma 13 (Action-Only TB and the Complete Stochastic Kernel). *Under the deterministic state rules of Section 3, each legal action path has probability*

$$P_\theta(\tau | q) = \prod_{t=1}^T \pi_\theta(a_t | f^{(k)}(s_{t-1}, s_{t-1})). \quad (41)$$

This is the actual action-path law, not the conditional law obtained by fixing a realized future reasoning or observation sequence. With stochastic reasoning and execution, the actual state-path law is instead the product of the marginal kernels in Eq. (42), provided σ is sufficient. TB with those kernels has the same terminal guarantee if its full-support zero-residual condition is satisfied.

Proof. In the deterministic case, the reasoning and transition factors in Eq. (3) are point masses equal to one on each legal realization. The chain rule then gives Eq. (41). Each alternative parent state uses its own value of $f^{(k)}$, so no future text is fixed across counterfactual action choices.

For the stochastic case, condition only on the *current* sufficient state. Summing the normalized joint distribution of (r, a, o) over events with next state s' gives the marginal forward kernel

$$P_F(s' | s) = \sum_{r,a,o} \rho_\theta(r | s) \pi_\theta(a | r, s) \mathcal{C}_{\text{exec}}(o | s, a) \mathbf{1}\{U(s, r, a, o) = s'\}. \quad (42)$$

Summing again over all s' gives one because U assigns each event a unique successor. State sufficiency means that any histories represented by s induce the same candidate supports and the same pushed-forward transition probabilities, and agree on terminal rewards. The state process is therefore Markov and its path law factors into these marginal kernels. Theorem 9 applies to that law directly.

Equivalently, one may retain intermediate states for reasoning selection, action selection, and execution feedback. On this expanded DAG the forward edge factors are $\rho_\theta(r | s)$, $\pi_\theta(a | r, s)$, and $\mathcal{C}_{\text{exec}}(o | s, a)$; the TB residual contains the log of *each* factor and the corresponding reverse-edge probabilities. All three supports must be discrete and finite for the finite-DAG result used here. Coarsening an expanded state is permitted only when sufficiency and reward invariance remain valid. $\square$

Conditioning on a realized future sequence is different: for example, if a_1 is a fair bit and $r_2 = a_1$, then $P(a_1 = 1 | r_2 = 1) = 1$, not the original action probability $1/2$. Stop-gradient changes a derivative, not this conditional law. A genuinely exogenous context c can instead be fixed before

an episode, in which case TB uses $Z(q, c)$ and terminal target $\tilde{R}(x; c)^\beta / Z(q, c)$. Marginalizing c gives the mixture of these normalized targets; it gives a common query-only target when those targets coincide, not merely because every conditional theorem holds.

Realizability with a fixed Executor. Fix P_B and positive terminal rewards, and let F_B be the unique backward-induced flow of Lemma 18. Its desired forward kernel is

$$P_F^*(s' | s) = \frac{F_B(s')P_B(s | s')}{F_B(s)} = \sum_{r,a} u_\theta(r, a | s) K_{r,a}(s' | s), \quad (43)$$

where $K_{r,a}(s' | s) = \sum_o \mathcal{C}_{\text{exec}}(o | s, a) \mathbf{1}\{U(s, r, a, o) = s'\}$ and $u_\theta = \rho_\theta \pi_\theta$ must be realizable by the joint policy class. Equation (43) at *all* states is necessary and sufficient for this specified (R, P_B) flow to be realized by that class: necessity follows from the actual marginal kernel, and sufficiency follows by substituting the equality into Detailed Balance and telescoping. With unrestricted statewise controls, it is the condition that $P_F^*(\cdot | s)$ lies in the convex hull of the available $K_{r,a}(\cdot | s)$; shared neural parameters can impose additional restrictions. In the deterministic action quotient, every outgoing edge is selectable, removing this chance constraint but not parametric limitations. For instance, a single uncontrollable transition with outcome probabilities (0.9, 0.1) cannot realize a uniform target over its two outcomes. Freezing that transition does not remove the obstruction.

(ii) Hindsight-asymmetric backward policy.

Lemma 14 (Fixed-Child Hindsight Conditioning Preserves the Backward Policy). *For each child x , let $b_\phi(x, p)$ be finite for all $p \in \text{Pa}(x)$ and depend only on canonical data determined by (x, p) . Then*

$$P_B(p | x) = \frac{\exp b_\phi(x, p)}{\sum_{p' \in \text{Pa}(x)} \exp b_\phi(x, p')} \quad (44)$$

is a single normalized parent distribution, independent of the actual arrival history. In the action quotient it may be indexed by the removable maximal occurrence, and the SkillFlow TB condition is

$$\log Z_\theta(q) + \sum_t \log \pi_\theta(a_t | f^{(k)}(s_{t-1}), s_{t-1}) = \beta \log \tilde{R}(s_T) + \sum_t \log P_\phi(a_t | s_t). \quad (45)$$

Proof. At fixed x the numerator terms are positive, and their sum is the denominator, so the vector sums to one. The context construction specifies every component from (x, p) , hence a different sampled parent cannot change it. Proposition 28 identifies the components with removable maximal occurrences on the trace graph. Substitution of this kernel and Eq. (41) into Theorem 9 gives Eq. (45). $\square$

Remark 4 (Why the Quotient Matters Here). *On a raw-history tree every non-source state has one parent, so $P_B \equiv 1$ and there is no reverse parent choice to learn. Valid quotient merges make that choice non-trivial. They are not required for TB itself: with $\mathcal{I} = \emptyset$, the theory remains valid on a tree, but its terminal objects are ordered histories. A library with no eligible commuting pair need not produce merges.*

(iii) Length-tempered weights and their local normalization.

Lemma 15 (Normalized Geometric-Mean Action Weights). *The weight $w_F(a | r, s) = e^{\ell_\theta(a; r, s)}$ is the geometric mean of the action's token probabilities. It is a positive score, not generally a probability or a sub-probability mass. Dividing by $C_F(s, r)$ gives a normalized complete-action policy. Analogously, $e^{b_\phi(x, p)} / C_B(x)$ is a normalized parent policy. TB evaluated with these normalized policies has the exact guarantee of Theorem 9; it need not preserve rankings of the original variable-length sequence likelihoods.*

Proof. The identity $\exp(K^{-1} \sum_j \log p_j) = \prod_j p_j^{1/K}$ gives the geometric mean. Positivity and finiteness of each candidate set give $0 < C_F, C_B < \infty$ and normalization by direct summation. If Δ_{raw} denotes the expression with ℓ_θ, b_ϕ but without their log-normalizers, then

$$\Delta(\tau) = \Delta_{\text{raw}}(\tau) - \sum_t \log C_F(s_{t-1}, r_t) + \sum_t \log C_B(s_t). \quad (46)$$

Those terms are included in Definition 9, so the loss enforces the ordinary TB identity for proper kernels. A single per-query term cannot replace a generally path-dependent correction. Nor is $p^{1/K}$ a common monotone transformation when K varies: $0.6 > 0.4$ but $0.6^{1/1} < 0.4^{1/10}$. Reward ordering at exact balance follows at the terminal level from TB, not from preserving raw likelihood rankings. $\square$

The length-tempered parameterization does not itself preclude arbitrary positive categorical action distributions on a finite support. To see this, fix desired masses $v_a > 0$ with $\sum_a v_a = 1$. There is a unique $c > 0$ with $\sum_a (cv_a)^{K(a)} = 1$, since the left side is continuous, strictly increasing, and ranges from zero to infinity. An unrestricted autoregressive scorer can assign complete-string probabilities $p(a) = (cv_a)^{K(a)}$ on the prefix tree of the end-marked strings. Their geometric means are cv_a , whose normalized values are exactly v_a . This is a representability construction for an unrestricted scorer; the shared neural parameterization still requires the realizability premise above.

Remark 5 (Why Per-Token Tempering, Not Token-Level GFlowNet). *Tokens are internal coordinates of one complete-action score. The action-only graph has one edge per action, and its sampler draws from the normalized distribution over complete candidates. Random candidate proposals would introduce another stochastic choice: either the proposal set and its law must be represented in the state process, or its induced action law must be marginalized. The finite deterministic support rule specified here avoids omitting such a factor.*

Together, these results identify the actual probability object used by TTB: a normalized action policy under causal deterministic state rules, or the complete stochastic kernel with chance constraints retained. Both use fixed-child backward normalization and the same full-support balance condition.

C ENTROPY-REGULARIZED RL EQUIVALENCE

This section identifies the entropy-regularized path objective corresponding to terminal reward matching on a DAG. The backward policy supplies the path weights that remove ordering multiplicity.

Theorem 16 (GFlowNet–RL Equivalence with Backward Shaping). *Fix a finite supported DAG, positive terminal rewards, and a normalized positive backward policy P_B . The reward-matching path distribution is*

$$\pi_{\text{GFN}}^*(\tau \mid q) = \frac{\tilde{R}(s_T)^\beta P_B(\tau \mid s_T)}{Z_\beta(q)}, \quad Z_\beta(q) = \sum_{x \in \mathcal{X}_q} \tilde{R}(x)^\beta. \quad (47)$$

It is the unique maximizer over path distributions of

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \pi} [\log \tilde{R}(s_T)] + \frac{1}{\beta} \mathbb{E}_{\tau \sim \pi} [\log P_B(\tau \mid s_T)] + \frac{1}{\beta} \mathcal{H}[\pi]. \quad (48)$$

If the forward policy class, including any fixed chance transitions, can realize this distribution, it is also the maximizer in that class and its terminal marginal is $\tilde{R}(x)^\beta / Z_\beta(q)$.

Proof. Write the objective with a multiplier λ for $\sum_\tau \pi(\tau) = 1$:

$$\begin{aligned} \mathcal{J}_\lambda(\pi) &= \sum_\tau \pi(\tau) \log \tilde{R}(s_T) + \frac{1}{\beta} \sum_\tau \pi(\tau) \log P_B(\tau \mid s_T) \\ &\quad - \frac{1}{\beta} \sum_\tau \pi(\tau) \log \pi(\tau) + \lambda \left(\sum_\tau \pi(\tau) - 1 \right). \end{aligned} \quad (49)$$

The stationary condition is

$$0 = \log \tilde{R}(s_T) + \frac{1}{\beta} \log P_B(\tau \mid s_T) - \frac{1}{\beta} (\log \pi^*(\tau) + 1) + \lambda. \quad (50)$$

Consequently,

$$\log \pi^*(\tau) = \beta \log \tilde{R}(s_T) + \log P_B(\tau \mid s_T) + \beta \lambda - 1. \quad (51)$$

Normalization uses the reverse-path identity for each terminal:

$$\sum_{\tau} \tilde{R}(s_T)^\beta P_B(\tau | s_T) = \sum_x \tilde{R}(x)^\beta \sum_{\tau \rightarrow x} P_B(\tau | x) = \sum_x \tilde{R}(x)^\beta = Z_\beta(q). \quad (52)$$

This gives Eq. (47). To verify global optimality and uniqueness, write $Q = \pi_{\text{GFN}}^*$. Substituting its log-density into Eq. (48) yields

$$\mathcal{J}(\pi) = \frac{1}{\beta} \log Z_\beta(q) - \frac{1}{\beta} \text{KL}(\pi \| Q). \quad (53)$$

Since KL divergence is nonnegative and vanishes exactly at $\pi = Q$, the stated maximizer is unique. Summing Q over paths ending at x gives the terminal marginal. If a constrained forward class contains Q , the same maximizer is feasible; otherwise Eq. (53) characterizes minimization of KL over that class, without guaranteeing attainment or exact terminal matching. $\square$

Remark 6 (Scope of the Equivalence). *The shaping term is additive along edges, since $\log P_B(\tau | x) = \sum_t \log P_B(s_{t-1} | s_t)$. Thus Eq. (48) has terminal log-reward, intermediate rewards $\beta^{-1} \log P_B(s_{t-1} | s_t)$, and path entropy temperature $1/\beta$. Without backward shaping, the optimizer is proportional to $\tilde{R}(s_T)^\beta$ over paths; its terminal marginal is proportional to $\text{Lin}(x) \tilde{R}(x)^\beta$ on the trace DAG (Proposition 29). On a tree $P_B \equiv 1$, so the shaping term vanishes. On a general DAG the shaped and unshaped terminal targets can also agree when path multiplicity is constant, but their full path laws need not agree. The theorem fixes P_B ; jointly learning it with TTB does not make the optimization updates identical to policy-gradient updates on a fixed RL objective.*

Remark 7 (Practical Implications). *Reward maximization alone does not prescribe how to distribute mass among equally effective strategies, while entropy and backward shaping specify a distribution on the supported quotient. SkillFlow realizes that target through a trajectory-balance regression objective and reuses the learned backward for curation. This is a distinction in the probability object, training construction, and available diagnostics, not a claim that every REINFORCE-family method must collapse or that entropy-regularized RL cannot preserve diversity.*

D DETAILED BALANCE AND FLOW METRICS

This section establishes the Detailed Balance condition and derives the flow-based credit assignment metrics.

D.1 DETAILED BALANCE THEOREM

Theorem 17 (Detailed Balance). *For a reward-matching flow F satisfying the Trajectory Balance condition (Theorem 9), the Detailed Balance (DB) condition holds at every edge:*

$$F(s) \cdot P_F(s' | s) = F(s') \cdot P_B(s | s'), \quad (54)$$

where $P_F(s' | s) = F(s \rightarrow s')/F(s)$ and $P_B(s | s') = F(s \rightarrow s')/F(s')$.

Proof. By definition of P_F and P_B from Definition 4:

$$\text{LHS} = F(s) \cdot \frac{F(s \rightarrow s')}{F(s)} = F(s \rightarrow s'), \quad (55)$$

$$\text{RHS} = F(s') \cdot \frac{F(s \rightarrow s')}{F(s')} = F(s \rightarrow s'). \quad (56)$$

Both sides equal the edge flow $F(s \rightarrow s')$. $\square$

Detailed Balance is an edgewise identity. With positive normalized kernels and the terminal boundary condition, full TB and the existence of such a common flow are equivalent by the constructive proof of Theorem 9. At a finite training residual, neither shared interior flow nor path-independent prefix estimates are assumed. On a general DAG a state may be reached along several trajectories, so reward-matching alone does not pin down the interior flows; what does pin them down is the backward policy. The next lemma makes this dependence explicit.

Lemma 18 (Uniqueness of the Reward-Matching Flow Given P_B). *On a finite pointed DAG, fix a positive normalized backward policy P_B and positive terminal values $\{F(x) = R(x)\}_{x \in \mathcal{X}}$. There exists a unique reward-matching flow inducing this P_B : for every state s ,*

$$F(s) = \sum_{x \in \text{Desc}(s) \cap \mathcal{X}} R(x) P_B(s | x), \quad (57)$$

where $\text{Desc}(s)$ includes s itself and $P_B(s | x)$ denotes the backward hitting probability, not a one-edge probability unless s is a parent of x . For SkillFlow, substitute $R(x) = \tilde{R}(x)^\beta$ in this generic flow statement. Full TB with a learned P_B identifies (i) these terminal values, (ii) the corresponding interior flow, and (iii) Detailed Balance. Realizing the induced forward kernel under fixed chance dynamics additionally requires Eq. (43).

Proof. We prove Eq. (57) by reverse induction on the depth of s (longest path from the source).

Base case (terminal $s = x \in \mathcal{X}$): $F(x) = R(x)$ by reward-matching, $\text{Desc}(x) \cap \mathcal{X} = \{x\}$ and $P_B(x | x) = 1$, so Eq. (57) is the identity $R(x) = R(x)$.

Inductive step: Suppose Eq. (57) holds for all states deeper than s . By flow conservation (Definition 2) together with $F(s \rightarrow s') = F(s') P_B(s | s')$ (Definition 4),

$$F(s) = \sum_{s' \in \text{Ch}(s)} F(s \rightarrow s') = \sum_{s' \in \text{Ch}(s)} F(s') P_B(s | s'). \quad (58)$$

By the inductive hypothesis and the Chapman–Kolmogorov identity $P_B(s | x) = \sum_{s' \in \text{Ch}(s)} P_B(s' | x) P_B(s | s')$ for the backward walk,

$$F(s) = \sum_{s' \in \text{Ch}(s)} \sum_{x \in \text{Desc}(s') \cap \mathcal{X}} R(x) P_B(s' | x) P_B(s | s') = \sum_{x \in \text{Desc}(s) \cap \mathcal{X}} R(x) P_B(s | x). \quad (59)$$

The reverse recursion uniquely specifies every $F(s)$. Conversely, define $F(s \rightarrow s') = F(s') P_B(s | s')$ from that recursion. Its outgoing sum is $F(s)$ at each non-terminal, and its incoming sum is $F(s)$ at every non-source by normalization of P_B . This proves existence as well as uniqueness. The induced forward kernel is normalized and satisfies DB. Every backward walk from a terminal hits the source, so $F(s_0) = \sum_x R(x)$. On a tree the hitting weights are one only for descendant terminals, yielding $F(s) = \sum_{x \in \text{Desc}(s) \cap \mathcal{X}} R(x)$. $\square$

D.2 FLOW RATIO AND STEP IMPORTANCE

Corollary 19 (Flow Ratio). *For normalized kernels at full TB, the constructed positive flow satisfies Detailed Balance. Therefore*

$$\frac{F_q(s_t)}{F_q(s_{t-1})} = \frac{P_F(s_t | s_{t-1})}{P_B(s_{t-1} | s_t)}. \quad (60)$$

Under the action-DAG convention the numerator is $\pi_\theta(a_t | f^{(k)}(s_{t-1}), s_{t-1})$, and the denominator is the fixed-child component $P_\phi(a_t | s_t)$.

Proof. Divide $F_q(s_{t-1}) P_F(s_t | s_{t-1}) = F_q(s_t) P_B(s_{t-1} | s_t)$ by the positive product $F_q(s_{t-1}) P_B(s_{t-1} | s_t)$. $\square$

Definition 11 (Step Importance). *At any iterate, the step importance is*

$$I_\tau(t) := \frac{P_F(s_t | s_{t-1})}{P_B(s_{t-1} | s_t)}. \quad (61)$$

We write $I(t)$ on a specified trajectory. At exact TB it equals the ratio of consecutive state flows; before exact balance it remains a well-defined normalized-policy ratio.

$I(t) > 1$ and $I(t) < 1$ describe amplification and attenuation of the prefix-flow estimate, respectively. Large $|\log I(t)|$ means the forward and reverse kernels place different relative weights on that edge. Because these kernels condition on different endpoints, the sign is neither a success label nor a causal advantage. A merge can have larger state flow than one particular parent, even though every state

flow remains at most the source flow. In the stochastic formulation, the ratio uses the actual marginal kernel; on an expanded graph, the product of subedge ratios provides the corresponding per-turn diagnostic, including reasoning and chance factors.

D.3 SKILL MARGINAL FLOW

Definition 12 (Invocation-Conditioned Skill Flow). *Let $\mathcal{B}_s \subseteq \mathcal{B}$ contain all sampled trajectories invoking skill s , and let V_s be the multiset of visits (τ, t) at which it is invoked. Repeated calls and repeated sampled trajectories are counted with multiplicity. For $|\mathcal{B}_s| > 0$, define*

$$\hat{F}(s) := \frac{1}{|\mathcal{B}_s|} \sum_{(\tau, t) \in V_s} \hat{f}_{\tau, t}, \quad (62)$$

where the prefix estimate is

$$\hat{f}_{\tau, t} := Z_\theta(q_\tau) \prod_{j=1}^t I_\tau(j), \quad \log \hat{f}_{\tau, t} = \log Z_\theta(q_\tau) + \sum_{j=1}^t \log I_\tau(j). \quad (63)$$

At full TB this estimate is $F_{q_\tau}(s_t)$ by Theorem 9; at finite residual, two prefixes reaching the same state may give different values.

This is the skill marginal-flow summary of main-text Eq. (8), with its invocation conditioning made explicit. It averages total post-invocation prefix flow per invoking trajectory. A large value may reflect high-flow invocation states or more invocations per such trajectory; a small value does not by itself establish rarity. The invocation rate $|\mathcal{B}_s|/|\mathcal{B}|$ and mean call count $|V_s|/|\mathcal{B}|$ are separate quantities. Atomicity specifies an invocation interface, not an at-most-once constraint.

For a balanced model, a genuine unconditional expected invocation count obeys

$$\mathbb{E}_{\tau \sim P_F(\cdot|q)} N_s(\tau) = \frac{1}{Z(q)} \sum_{(u, v) \in \mathcal{A}_s} F_q(u \rightarrow v), \quad (64)$$

where $\mathcal{A}_s$ is the set of edges invoking s and N_s counts such edges. This follows by writing N_s as a sum of edge-visit indicators and using $P_F(\text{traverse } u \rightarrow v) = F_q(u \rightarrow v)/Z(q)$. It can exceed one. Equation (62) is not an estimator of this edge-occupancy sum; the two objects answer different questions.

D.4 ZERO-COST COMPUTATION

Proposition 20 (Zero-Cost Flow Metrics). *The step ratios, prefix estimates, and invocation-flow summaries reuse the normalized forward and backward log-probabilities already evaluated for TTB:*

$$\log I_\tau(t) = \log P_F(s_t | s_{t-1}) - \log P_B(s_{t-1} | s_t), \quad (65)$$

$$\log \hat{f}_{\tau, t} = \log Z_\theta(q_\tau) + \sum_{j=1}^t \log I_\tau(j), \quad (66)$$

$$\log \hat{F}(s) = \log \left(\frac{1}{|\mathcal{B}_s|} \sum_{(\tau, t) \in V_s} \exp \left[\log Z_\theta(q_\tau) + \sum_{j=1}^t \log I_\tau(j) \right] \right). \quad (67)$$

The final expression is the log of an arithmetic mean, not the mean log-flow. In a mixed-query batch each $Z_\theta(q_\tau)$ remains inside the sum.

Proof. The first equation is the definition of the ratio, the second accumulates its log along a recorded prefix, and the third substitutes that estimate into Eq. (62). Only arithmetic on the TTB quantities is added. $\square$

Here, “zero-cost” means no additional policy-model evaluations to form the diagnostics after TTB scoring. Computing candidate normalizers, marginal stochastic probabilities when used, and the backward policy itself remains part of TTB’s cost. If a state edge has marginalized out the identity of a skill invocation, skill-specific diagnostics require retaining that invocation label in the graph or computing its conditional expected count.

D.5 CUMULANT GENERATING FUNCTION (CGF) PROPERTIES

For each skill with nonempty visit multiset V_s , define

$$\Lambda_\lambda^{(s)} := \log \left(\frac{1}{|V_s|} \sum_{v \in V_s} e^{\lambda X_v} \right), \quad \lambda \in \mathbb{R}, \quad (68)$$

where a visit $v = (\tau, t)$ has log-flow share

$$X_v := \sum_{j=1}^t \log I_\tau(j) = \log \hat{f}_{\tau,t} - \log Z_\theta(q_\tau). \quad (69)$$

At exact balance, $X_v = \log(F_{q_\tau}(s_t)/Z(q_\tau))$ and $F_q(s)/Z(q) = P_F(\text{visit } s \mid q) \leq 1$, by Proposition 5. Hence $X_v \leq 0$ then. Prefix estimates need not obey this bound before balance.

Visit normalization and repeated calls. The empirical distribution is uniform over the *visits*, so

$$\Lambda_\lambda^{(s)} = \log \mathbb{E}_{V_s}[e^{\lambda X}], \quad \Lambda_0^{(s)} = 0. \quad (70)$$

Let $m_s = |V_s|/|\mathcal{B}_s| \geq 1$ be the mean number of calls per invoking trajectory. Using $|\mathcal{B}_s|$ instead of $|V_s|$ inside the logarithm would add $\log m_s$ to the function and to its value at zero. We therefore use visit normalization for the CGF and retain m_s explicitly when relating it to $\hat{F}(s)$. This choice weights occurrences uniformly; it is not equal query weighting when queries generate different numbers of visits.

Lemma 21 (Convexity of the CGF in λ). $\Lambda_\lambda^{(s)}$ is convex on $\mathbb{R}$.

Proof. Equation (70) is a log-sum-exp of affine functions λX_v minus the constant $\log |V_s|$, hence convex. More explicitly, with $w_\lambda(v) = e^{\lambda X_v} / \sum_u e^{\lambda X_u}$,

$$\partial_\lambda \Lambda_\lambda^{(s)} = \sum_v w_\lambda(v) X_v, \quad \partial_\lambda^2 \Lambda_\lambda^{(s)} = \text{Var}_{w_\lambda}(X) \geq 0.$$

All derivatives exist because the visit multiset is finite. $\square$

Lemma 22 (Mean Log-Flow Identity for $G(s)$).

$$G(s) := \left. \frac{\partial \Lambda_\lambda^{(s)}}{\partial \lambda} \right|_{\lambda=0} = \frac{1}{|V_s|} \sum_{v \in V_s} X_v = \mathbb{E}_{V_s}[\log \hat{f}_v - \log Z_\theta(q_v)]. \quad (71)$$

Proof. Differentiating the finite sum gives

$$\frac{\partial \Lambda_\lambda^{(s)}}{\partial \lambda} = \frac{\sum_v X_v e^{\lambda X_v}}{\sum_v e^{\lambda X_v}}. \quad (72)$$

At $\lambda = 0$, numerator and denominator give the visit mean. Substitution of Eq. (69) establishes the last expression. The query-dependent shift is inside the expectation. $\square$

Lemma 23 (CGF at One and Normalized Invocation Flow). *For arbitrary mixed-query visits,*

$$\Lambda_1^{(s)} = \log \left(\frac{1}{|V_s|} \sum_{v \in V_s} \frac{\hat{f}_v}{Z_\theta(q_v)} \right). \quad (73)$$

If all those visits have the same query q , then $\Lambda_1^{(s)} = \log \hat{F}(s) - \log Z_\theta(q) - \log m_s$.

Proof. Set $\lambda = 1$ in Eq. (68) and use $e^{X_v} = \hat{f}_v / Z_\theta(q_v)$. When $q_v = q$ for every visit, the normalizer is constant and

$$\frac{1}{|V_s|} \sum_v \frac{\hat{f}_v}{Z_\theta(q)} = \frac{|\mathcal{B}_s|}{|V_s| Z_\theta(q)} \hat{F}(s) = \frac{\hat{F}(s)}{m_s Z_\theta(q)}.$$

Taking logs gives the stated special case. For mixed queries the constant cannot be factored out; even library centering does not remove different query mixtures across skills. $\square$

Lemma 24 (Jensen Inequality on the CGF). *For every s with $V_s \neq \emptyset$,*

$$G(s) \leq \Lambda_1^{(s)}, \quad (74)$$

with equality if and only if X_v is constant over all visits.

Proof. By strict convexity of the exponential,

$$\exp\left(\frac{1}{|V_s|} \sum_v X_v\right) \leq \frac{1}{|V_s|} \sum_v e^{X_v}, \quad (75)$$

with equality exactly when all X_v agree. Taking logarithms proves the claim. $\square$

Proposition 25 (Jensen Gap and the Cumulant Expansion). *The Jensen gap $J(s) = \Lambda_1^{(s)} - G(s)$ has the exact representation*

$$J(s) = \int_0^1 (1 - \lambda) \text{Var}_{w_\lambda}(X) d\lambda = \frac{1}{2} \text{Var}_{V_s}(X) + \mathcal{R}_s, \quad |\mathcal{R}_s| \leq \frac{d_s^3}{6}, \quad (76)$$

where $d_s = \max_v X_v - \min_v X_v$. If the CGF’s Taylor series at zero converges at one to the CGF, then also $J(s) = \frac{1}{2} \text{Var}_{V_s}(X) + \sum_{j \geq 3} \kappa_j(s)/j!$, with empirical cumulants $\kappa_j(s)$.

Proof. Let $L(\lambda) = \Lambda_\lambda^{(s)}$. Since $L(0) = 0$ and $L'(0) = G(s)$, Taylor’s formula with integral remainder gives

$$L(1) - L'(0) = \int_0^1 (1 - \lambda) L''(\lambda) d\lambda.$$

The convexity proof identifies $L''(\lambda)$ with the variance under exponential tilt, proving the exact identity. Moreover, $L'''(\lambda) = \mathbb{E}_{w_\lambda}[(X - \mathbb{E}_{w_\lambda} X)^3]$. All tilted means lie between the smallest and largest X_v , so $|L'''(\lambda)| \leq d_s^3$ on $[0, 1]$. The second-order Taylor remainder consequently satisfies

$$|L(1) - L'(0) - \frac{1}{2} L''(0)| \leq \frac{1}{2} \int_0^1 (1 - \lambda)^2 d_s^3 d\lambda = \frac{d_s^3}{6}.$$

Finally, where the cumulant series converges to L , substituting $\lambda = 1$ into $L(\lambda) = G(s)\lambda + \frac{1}{2} \text{Var}_{V_s}(X)\lambda^2 + \sum_{j \geq 3} \kappa_j(s)\lambda^j/j!$ gives the series expression. The integral identity and remainder bound require no such series convergence assumption. $\square$

Remark 8 (Stability Diagnostic via the Jensen Gap). *The gap is nonnegative and vanishes exactly for constant log-flow shares. It provides a descriptive measure of context variation under the empirical visit distribution; for small spread its leading term is one-half the variance. It is not universally ordered by variance alone, and small empirical gap does not establish context-independent causal usefulness. Cross-query variation is computed from $X_v = \log \hat{f}_v - \log Z_\theta(q_v)$, not from raw $\log \hat{f}_v$ with an assumed common shift.*

Remark 9 (Centered Log-Flow Share). *Let $\mathcal{S}_{\text{sup}}$ contain the tips with at least $n_{\min}$ visits at the boundary. If this set is nonempty, take uniform library averages over it and define*

$$\tilde{\Lambda}(s) = \Lambda_1^{(s)} - \frac{1}{|\mathcal{S}_{\text{sup}}|} \sum_{u \in \mathcal{S}_{\text{sup}}} \Lambda_1^{(u)}, \quad \tilde{G}(s) = G(s) - \frac{1}{|\mathcal{S}_{\text{sup}}|} \sum_{u \in \mathcal{S}_{\text{sup}}} G(u). \quad (77)$$

These are relative scores among supported tips. Since balanced state-flow shares are at most one, raw $G(s)$ is nonpositive at exact balance; a zero retention threshold is therefore applied to $\tilde{G}$, not to G . Centering preserves relative comparisons of the defined normalized statistics but does not convert invocation-conditioned summaries into occupancy or cancel heterogeneous query mixtures. If no tip is supported, no library average is formed and all existing tips are held over.

E INDEPENDENCE, ACYCLICITY, AND PARENT STRUCTURE

This appendix formalises $\mathcal{I}$, proves that the quotiented orchestration graph is a pointed DAG, characterises its parent sets, and quantifies the multiplicity bias of balancing on raw action histories.

E.1 THE INDEPENDENCE RELATION

Definition 13 (Independence Relation). *Fix a phase and its legal action language $\mathcal{L} \subseteq \Sigma^{\leq T_{\max}}$. A symmetric, irreflexive relation $\mathcal{I} \subseteq \Sigma \times \Sigma$ is admissible if swapping adjacent independent actions preserves legality and canonical state. In particular, for $(a, a') \in \mathcal{I}$ and every reachable prefix where the pair occurs,*

$$\delta(\delta(s, a), a') = \delta(\delta(s, a'), a), \quad (78)$$

with both sides legal. Equality is equality of the sufficient canonical state: the rendered inputs to the Supervisor and reasoning rule, the Executor/environment state, action availability, stopping status, and terminal reward agree. The equivalence $\equiv_{\mathcal{I}}$ is generated by these swaps; $D_{\mathcal{I}} = (\Sigma \times \Sigma) \setminus \mathcal{I}$ is the dependence relation.

$\mathcal{I}$ is declared as part of the state interface, not inferred from superficial similarity of action names. A read-only operation is not automatically independent of another: its observations, subsequent retrieval, or context ordering can still matter. Operations sharing mutable state, position-coupled navigation, and pairs involving `accept` remain dependent unless the stronger interface conditions are satisfied. Setting $\mathcal{I} = \emptyset$ gives ordered histories. For stochastic dynamics, equality of rendered strings alone is insufficient: histories in a class must have identical pushed-forward kernels in Eq. (42) and reward, or the state is refined until they do.

Lemma 26 (Canonical Skill Activations Commute). *Consider distinct tips s_i, s_j with the atomic activation interface of Definition 19. If both activation orders are legal and no order-dependent truncation, retrieval, or mutation is performed, then $\text{skill}(s_i)$ and $\text{skill}(s_j)$ satisfy Eq. (78). They may be included in $\mathcal{I}$ when this holds throughout the supported legal language.*

Proof. Activation increments the multiplicity of the selected versioned tip in a canonically rendered guidance multiset, leaves the external environment and prior execution records unchanged, and appends an occurrence to the trace. Both orders produce the same multiset with the same counts, the same action count, and traces identified by the allowed adjacent swap. Thus their canonical representations agree. The next reasoning value is recomputed from that canonical state; order-dependent intermediate reasoning logs are not fed back. The common representation and unchanged environment give identical subsequent inputs, transitions, availability, and reward. This is the asserted commutation. $\square$

This lemma is a property of the activation and state interfaces, not of bounded text length alone. It permits repeated invocations: two occurrences of the *same* action label are dependent because $\mathcal{I}$ is irreflexive. An empty or singleton library need not supply an independent pair.

E.2 ACYCLICITY

Theorem 27 (Orchestration Graph is a Pointed DAG). *Under the fixed-phase support and sufficient-state rules of Section 3, the reachable quotient $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with $\mathcal{V} = \mathcal{L}/\equiv_{\mathcal{I}}$, is a finite DAG with unique source $s_0 = [\epsilon]$.*

Proof. Every generating swap permutes occurrences without creating or deleting them, so the action count $|\llbracket u \rrbracket| = |u|$ is well defined. With $\text{depth}(s) = |s|$, each edge satisfies

$$\text{depth}(s') = \text{depth}(s) + 1 > \text{depth}(s). \quad (79)$$

A directed cycle would require a strictly increasing depth to return to its starting value, a contradiction. The empty trace has no incoming edge. Every other vertex is reachable by a nonempty legal word and therefore has an incoming edge from the prefix omitting its last occurrence; the source is unique. Finite branching and $T \leq T_{\max}$ give finitely many reachable words and hence finitely many quotient states. Each maximal legal path ends at `accept` or the horizon, so each state can reach a terminal. $\square$

E.3 PARENT SETS

For $s = \llbracket u \rrbracket$, define the dependence order on *occurrences* in u : occurrence i precedes j if $i < j$ and $(u_i, u_j) \in D_{\mathcal{I}}$, followed by transitive closure. Let $\max(s)$ contain the maximal occurrences in this finite poset. The notation $u \setminus i$ deletes occurrence i , not every occurrence with its label.

Proposition 28 (Parents are the Maximal Occurrences). *For every non-source trace $s = [u]$,*

$$\text{Pa}(s) = \{[u \setminus i] : i \in \max(s)\}, \quad |\text{Pa}(s)| = |\max(s)| \leq \text{width}(s) \leq |s|. \quad (80)$$

Thus parents are precisely the legal states that can precede s by deleting an occurrence that could have been last, and are explicitly enumerable.

Proof. If i is maximal, its label is independent of every occurrence following it in a representative u ; otherwise it would have a successor in the dependence order. Move it to the end by adjacent independent swaps. Swap-closure of $\mathcal{L}$ preserves legality at each step, so $[u \setminus i] \rightarrow [u]$ is an edge.

Conversely, if $[v] \rightarrow [u]$ is an edge, some representative of $[u]$ is va . Its final occurrence is maximal, since no dependent occurrence follows it, and deleting it yields $[v]$. Dependence orders are preserved by the generating swaps, so this identifies a maximal occurrence of s .

Distinct maximal occurrences have different action labels: equal labels are dependent and hence comparable in occurrence order. Deleting two different maximal occurrences therefore yields different label multisets and different parent traces. This proves the cardinality equality. Maximal elements form an antichain, whose size is at most the width; the width is at most the number of occurrences. Equality with the width need not hold: two incomparable occurrences both preceding a third give width two but only one maximal occurrence. $\square$

For the fixed horizon, $|\text{Pa}(s)| \leq T_{\max}$. This bound concerns the parent softmax, not the potentially much larger forward candidate set. Each candidate’s score must still be evaluated using its own deterministic pair (s, p) , as in Eq. (44).

E.4 MULTIPLICITY BIAS OF THE UNQUOTIENTED FORMULATION

Proposition 29 (Multiplicity Bias). *Let $\text{Lin}(x) = |\{u \in \mathcal{L} : [u] = x\}|$ count the legal linearizations of terminal trace x . If TB is enforced on raw action histories with each linearization assigned terminal reward $\tilde{R}(x)^\beta$, then*

$$P_{\text{hist}}(x) = \frac{\text{Lin}(x)\tilde{R}(x)^\beta}{\sum_y \text{Lin}(y)\tilde{R}(y)^\beta}, \quad P_{\mathcal{G}}(x) = \frac{\tilde{R}(x)^\beta}{\sum_y \tilde{R}(y)^\beta}. \quad (81)$$

If the dependence poset of x contains an antichain of k occurrences, then $\text{Lin}(x) \geq k!$.

Proof. On the history tree each legal word u is a separate terminal with reward $\tilde{R}([u])^\beta$. Theorem 6 assigns mass proportional to this value, and summing the $\text{Lin}(x)$ copies yields the first expression. On the quotient there is one terminal per trace, giving the second expression.

For the bound, fix an antichain of k occurrences. Any prescribed total order of these occurrences can be extended to a linear extension of the whole poset: adjoining that order creates no cycle, since a pre-existing directed path between distinct selected occurrences would contradict their incomparability. The resulting finite acyclic order has a topological ordering. Different prescribed orders give distinct linearizations, and swap-closure makes them legal. The selected occurrences have distinct labels because equal labels are dependent, so these orders yield distinct words. There are $k!$ choices. Pairwise independence of selected labels alone does not suffice if intervening dependent occurrences make them transitively comparable. $\square$

Remark 10 (Why This Matters for Diversity). *History-level balancing tilts the terminal-strategy distribution by the number of its admissible orderings, a factor unrelated to its reward. For example, an antichain of three or four occurrences permits at least six or twenty-four linearizations. Quotient-level TB avoids this tilt by assigning the terminal mass once and allocating it among paths through the normalized $P_B(\tau \mid x)$ factor.*

Proof of Proposition 1 (main text). Theorem 27 supplies the finite pointed graph; Proposition 28 identifies its parent sets. Choose any positive normalized backward kernel and positive terminal rewards. The reverse recursion of Lemma 18 constructs a positive conservative flow with those terminal values, hence the graph admits such a flow. A legal commuting pair with distinct labels yields two distinct predecessor states after both actions. Existence of the graph-level flow does not

imply that an arbitrary neural policy can realize it: in the deterministic action model the outgoing choices are controllable, while the stochastic model must satisfy Eq. (43) and the shared parameter constraints. These are precisely the structural and realizability qualifications in the proposition. $\square$

F SKILL CURATION DETAILS

This appendix specifies (i) phase-boundary detection, (ii) the curation classes used by Φ , (iii) the Skill Creator Ψ and its trigger steps, (iv) the resulting curation algorithm, and (v) atomic composability and its preservation.

F.1 PHASE-BOUNDARY DETECTION RULE

Within phase k , track the raw squared TTB residual over a window of W training steps:

$$\overline{\Delta}_w^{(k)} := \frac{1}{W} \sum_{i=w-W+1}^w \frac{1}{|\mathcal{B}_i|} \sum_{\tau \in \mathcal{B}_i} \Delta(\tau | \mathcal{S}^{(k)})^2. \quad (82)$$

Each residual is evaluated with the parameters at its training step. This observed statistic describes optimization progress; it is not the infimum in Eq. (9). In particular, a small gradient or a flat sequence of windows does not imply that the infimum has been attained.

Definition 14 (Plateau Trigger). *Let $\rho > 0$ be a relative-change tolerance, $M \geq 1$ a persistence count, and $\epsilon_{\text{pl}} > 0$ a residual tolerance. At check times $w = jW$ with $j \geq 2$, compare two completed, non-overlapping windows:*

$$\frac{|\overline{\Delta}_w^{(k)} - \overline{\Delta}_{w-W}^{(k)}|}{\max\{\overline{\Delta}_{w-W}^{(k)}, \epsilon_{\text{pl}}\}} < \rho, \quad \overline{\Delta}_w^{(k)} > \epsilon_{\text{pl}}. \quad (83)$$

A boundary is triggered at the first eligible check for which both conditions hold in M consecutive window comparisons. The first such trigger requires at least $M + 1$ completed windows. Window and comparison counters restart after each boundary.

The denominator guard makes the rule defined near zero; the residual threshold avoids treating an already negligible residual as evidence for another edit. Taking an absolute change also prevents a large increase from being counted as a plateau. Eligibility may impose a minimum completed-experience budget, but neither eligibility nor Eq. (83) establishes a capacity limitation. W , ρ , M , and ϵ_{pl} are operational hyperparameters, not consequences of the TB theorem.

F.2 CURATION CLASSES

At a boundary, use a batch whose TTB scores correspond to one common current-phase parameter snapshot. Let $\mathcal{S}_{\text{sup}}^{(k)} = \{s \in \mathcal{S}^{(k)} : |V_s| \geq n_{\min}\}$, with $n_{\min} \geq 2$, and hold over $\mathcal{H}_k = \mathcal{S}^{(k)} \setminus \mathcal{S}_{\text{sup}}^{(k)}$. The statistics of Lemmas 22–24 are defined only when visits exist, and relative averages use only $\mathcal{S}_{\text{sup}}^{(k)}$. Let $n_k^-(s)$ count consecutive supported boundaries with negative centered share:

$$n_k^-(s) = \begin{cases} n_{k-1}^-(s) + 1, & s \in \mathcal{S}_{\text{sup}}^{(k)}, \tilde{\Lambda}_k(s) < 0, \\ 0, & \text{otherwise.} \end{cases} \quad (84)$$

Initialize $n_{k-1}^-(s) = 0$ for seed tips. New or refined tip versions start with count zero; an unsupported boundary breaks the streak.

Definition 15 (Curation Classes). *For thresholds $\Phi_{\text{thr}}^G \in \mathbb{R}$, $\Phi_{\text{thr}}^J > 0$, and $K^- \geq 1$, partition the supported tips as*

$$\mathcal{D}_k^- := \{s \in \mathcal{S}_{\text{sup}}^{(k)} : n_k^-(s) \geq K^-\}, \quad (85)$$

$$\mathcal{R}_k := \{s \in \mathcal{S}_{\text{sup}}^{(k)} \setminus \mathcal{D}_k^- : \tilde{G}(s) \geq \Phi_{\text{thr}}^G, J(s) \leq \Phi_{\text{thr}}^J\}, \quad (86)$$

$$\mathcal{U}_k := \mathcal{S}_{\text{sup}}^{(k)} \setminus (\mathcal{D}_k^- \cup \mathcal{R}_k). \quad (87)$$

Together with $\mathcal{H}_k$, these are four disjoint classes covering the library. Pruning takes precedence, retention requires a high relative mean and small gap, and the other supported tips enter refinement.

The hold-over set prevents zero visits or a one-visit gap of zero from being treated as strong evidence. Repeated invocations count as visits in the empirical statistic; they do not automatically constitute independent statistical evidence. The zero choice $\Phi_{\text{thr}}^G = 0$ compares a tip with the supported library mean through $\tilde{G}(s)$; it does not require a post-action state to exceed the query’s total flow. Thresholds guide a descriptive curation rule, not a significance test or a guarantee of improvement.

F.3 SKILL CREATION FROM SUCCESS/FAILURE TRAJECTORY PAIRS

The Skill Creator Ψ uses contrasting trajectories from the same query, with step importance selecting candidate decision gaps.

Definition 16 (Trigger Steps). *For a query q in the development pool used for evolution, let τ^+ and τ^- be current-phase trajectories with $R(\tau^+) \geq R_{\text{hi}} > R_{\text{lo}} \geq R(\tau^-)$. Define*

$$\mathcal{T}_q^{\text{trig}} := \{t \in \{1, \dots, |\tau^+|\} : |\log I_{\tau^+}(t)| \geq \zeta_{\text{trig}}, t \notin \text{cov}(\mathcal{H}_k \cup \mathcal{R}_k \cup \mathcal{U}'_k)\}. \quad (88)$$

Here $\zeta_{\text{trig}} > 0$ and cov is a fixed coverage rule marking positions already addressed by surviving tips. A step and its contrasting trajectory context guide generation; equal step indices in the two trajectories need not represent equal states. The ratio’s sign may inform a prompt heuristic, but it does not determine whether an action is causally helpful or harmful.

Definition 17 (Skill Creator Ψ). Ψ is a frozen LLM-based candidate generator with a typed admission interface: it returns an accepted atomic tip (Definition 19) or $\perp$. Given $c = (q, \tau^+, \tau^-, t)$ and the development trajectory buffer $\mathcal{T}$,

$$s_{\text{new}} = \Psi(c, \mathcal{T}, \mathcal{S}^{(k)}) \in \mathcal{S}_{\text{atomic}} \cup \{\perp\}. \quad (89)$$

Candidates are prompted to be self-contained, bounded by L_{max} , and phrased as strategic guidance rather than executable actions. Admission checks these structural requirements and binds the accepted payload to the read-only activation interface. The atomic-preservation result assumes that this admission predicate is sound for the stated interface; prompting alone is not the premise. In refine mode, Ψ proposes a new version of a tip under the same constraints. The experimental generator choices are described in Appendix F.9.

F.4 THE CURATION OPERATOR Φ

Definition 18 (Evolution Operator). *The CGF-based operator assembles*

$$\begin{aligned} \mathcal{S}^{(k+1)} &:= \Phi(\mathcal{S}^{(k)}, n_{k-1}^-, \mathcal{T}; \{\tilde{G}(s), \tilde{\Lambda}(s), J(s)\}_{s \in \mathcal{S}_{\text{sup}}^{(k)}}, \\ &\quad \{|V_s|\}_{s \in \mathcal{S}^{(k)}}, \{\log I_{\tau}(t)\}_{\tau, t}) \\ &= \mathcal{H}_k \cup \mathcal{R}_k \cup \mathcal{U}'_k \cup \Psi_k^{\text{new}}. \end{aligned} \quad (90)$$

$\mathcal{U}'_k$ contains one accepted replacement for each tip in $\mathcal{U}_k$, or the original tip if refinement returns $\perp$. For new tips, first form the accepted candidate set

$$\mathcal{C}_k := \bigcup_{(q, \tau^+, \tau^-)} \bigcup_{t \in \mathcal{T}_q^{\text{trig}}} (\{\Psi((q, \tau^+, \tau^-, t), \mathcal{T}, \mathcal{S}^{(k)})\} \setminus \{\perp\}). \quad (91)$$

Remove existing versions and duplicate payloads from the new-candidate set. Assuming $|\mathcal{S}^{(0)}| \leq B_{\text{max}}$, Ψ_k^{new} selects at most $B_{\text{max}} - |\mathcal{H}_k \cup \mathcal{R}_k \cup \mathcal{U}'_k|$ candidates from $\mathcal{C}_k \setminus \mathcal{S}^{(k)}$, in decreasing trigger magnitude $|\log I(t)|$ with canonical lexical tie-breaking. Refinement replaces only its own version. Thus pruned versions cannot re-enter through the new-candidate set, while rejected refinements preserve their original tips.

This operator may grow, shrink, or leave unchanged the library. Its inputs prioritize edits; no monotone increase in cardinality, terminal reward, or attainable flow quality is implied.

F.5 ATOMIC COMPOSABILITY AND ITS PRESERVATION

Definition 19 (Atomic Tip). *A tip is atomic if (i) it is a self-contained guidance payload of at most L_{max} tokens, with no runtime dependency on another tip; (ii) its versioned `skill` activation only increments its count in the canonical guidance multiset, without modifying other tips or the external*

task environment; and (iii) it is treated as guidance rather than an executable environment command. Canonical rendering retains occurrence counts and applies no order-dependent truncation. The payload bound controls the size of one tip, not the wall-clock cost of rendering or processing the whole context.

Definition 20 (Atomic Composability of a Library). *A library is atomically composable when all of its tips satisfy Definition 19. Commutation of distinct activations additionally requires their legal-order conditions in Lemma 26; repeated activations are permitted.*

Lemma 30 (Φ Preserves Atomic Composability). *If $\mathcal{S}^{(k)}$ is atomically composable and the admission interface of Ψ accepts only atomic tips, then $\mathcal{S}^{(k+1)}$ defined by Eq. (90) is atomically composable.*

Proof. We verify the four constituents of Eq. (90).

Held-over and retained tips ($\mathcal{H}_k \cup \mathcal{R}_k \subseteq \mathcal{S}^{(k)}$) inherit atomicity unchanged.

Refined tips in $\mathcal{U}'_k$ either pass the sound atomic admission predicate or are their original atomic versions after rejection. Thus each remains atomic.

New tips in Ψ_k^{new} are accepted outputs from $\mathcal{C}_k$; duplicate removal and capacity selection take a subset without changing their payload or interface.

Pruning removes versions and introduces no new tip. Atomicity is a per-tip structural property, preserved by these unions, subset selections, and atomic replacements. Every member of the next library therefore satisfies the definition. $\square$

Remark 11 (Action-Space Constancy Within a Phase). *The library and the state-based support rule change only between phases, after current episodes terminate. They are fixed within a phase even though the available candidate set can differ between states. The next phase re-establishes the same canonical activation and swap-legality conditions; merely retaining the library’s atomicity would not by itself prove state sufficiency for arbitrary other tools.*

F.6 CURATION ALGORITHM

At a triggered phase boundary $k \rightarrow k + 1$, the procedure is:

Algorithm 1: Skill-Library Curation at Phase Boundary $k \rightarrow k + 1$.

1. From the boundary batch’s current-phase TTB scores, form visit multisets V_s and prefix log-shares X_v . Use a common parameter snapshot for the curation statistics; do not mix stale scores from different policies as one flow.
2. For supported tips, compute $\Lambda_1^{(s)}$, $G(s) = |V_s|^{-1} \sum_v X_v$, $J(s)$, and the supported-library centers. Values of the CGF at zero and one alone do not determine its derivative at zero.
3. Hold over $\mathcal{H}_k$, update the consecutive negative-share counters, and partition supported tips into $\mathcal{D}_k^-$, $\mathcal{R}_k$, and $\mathcal{U}_k$ using Definition 15.
4. Refine each tip in $\mathcal{U}_k$; accept an atomic replacement or keep the original to form $\mathcal{U}'_k$.
5. From the development buffer, select same-query high/low-reward pairs and identify $\mathcal{T}_q^{\text{trig}}$ using Definition 16.
6. Generate candidates at those positions, apply atomic admission and duplicate removal, and fill the remaining capacity to obtain Ψ_k^{new} .
7. Assemble $\mathcal{S}^{(k+1)}$ by Eq. (90). Install the next-phase canonical state and candidate rules between episodes; initialize counters for new versions and restart the plateau windows.
8. Warm-start the forward, backward, and partition parameters. Their numerical initialization transfers, but the target $Z^{(k+1)}(q) = \sum_{x \in \mathcal{X}_q^{(k+1)}} \tilde{R}(x)^\beta$ generally changes. New-phase balance must be learned on the new support; old-library trajectories are not reused without legality and probability re-evaluation.

Lemma 30 establishes the structural invariant of this procedure. Theorem 27 then supplies a pointed DAG under the maintained support and state rules. Exact balance, when attainable and satisfied in a phase, supplies its flow guarantee separately.

F.7 EVOLUTION HYPERPARAMETERS

Algorithm 1 uses the centered-CGF gates defined above.

Symbol	Role	Appears in	Value
<i>When to evolve — plateau trigger (Definition 14)</i>			
ρ	relative-change tolerance	Eq. (83)	0.05
—	distinct experiences required before the first boundary	Appendix F.1	16
—	trajectories required before a boundary may fire	Appendix F.1	24
<i>What to keep — curation classes (Definition 15)</i>			
Φ_{thr}^G	centered mean-log-flow threshold for retention	§F.2	0.0
Φ_{thr}^J	Jensen-gap threshold (stability diagnostic)	§F.2	1.0
K^-	consecutive boundaries with $\tilde{\Lambda}(s) < 0$ before pruning	§F.2	2
$n_{\min}$	minimum s -visits in the batch before s is classified	§F.2	5 visits
$B_{\max}$	library capacity	Appendix F.4	60 skills
<i>Where to create — trigger steps and Skill Creator</i>			
ζ_{trig}	absolute $ \log I(t) $ threshold selecting trigger steps	Definition 16	1.0
$ \mathcal{S}^{(0)} $	initial library size (grown from scratch)	§5	0
—	Skill Creator Ψ	Appendix F.3	GPT-5.5, frozen

The window length and curation thresholds are hyperparameters rather than consequences of flow matching. In Definition 15, $\Phi_{\text{thr}}^G = 0$ acts on the relative statistic $\tilde{G}(s)$, so a retained tip has a mean log-flow share at least as high as the supported-library mean and a sufficiently small Jensen gap. Applying zero directly to raw $G(s)$ would instead compare state flow with total source flow. The separate residual tolerance $\epsilon_{\text{pl}} > 0$ in Definition 14 specifies numerical stopping sensitivity and is not derived from the reward floor.

F.8 WARM-STARTING Z_θ ACROSS LIBRARY CHANGES

Step 8 warm-starts $Z_\theta(q)$ rather than reinitializing it. Because the head is conditioned on the query and its task type—not indexed by the skill library—its input signature is unchanged when $\mathcal{S}$ grows or shrinks, so the estimate transfers directly.

F.9 SKILL CREATOR ROBUSTNESS ACROSS GENERATOR MODELS

The reported runs instantiate Ψ with GPT-5.5 (Appendix F.3), which invites a sharp objection: perhaps the gains simply reflect a strong proprietary model distilling skills, with the flow machinery contributing little. We test this directly. Holding everything else fixed—Supervisor, TTB objective, training data and order, executor, tool access, and the evolution rules—we swap only the generator behind Ψ for three models from different families: Claude Opus 5, Qwen3.8-27B, and DeepSeek-V4-flash.¹ Qwen3.8-27B is the decisive case: it is open-weight and far smaller than GPT-5.5, so if the method were a thin wrapper around a proprietary generator, that substitution should break it.

The substitution does not break the method. Replacing GPT-5.5 with the open-weight Qwen3.8-27B costs 4.84 points on average, yet the resulting system still matches or exceeds the strongest run-1 baseline on 10 of 14 benchmarks, falling short only on AIME (−3.33), MATH-Hard (−4.69), MedQA (−3.13) and HumanEval (−0.79). On WebShop, ALFWorld, ScienceWorld and Mind2Web—the interactive tasks where orchestration rather than raw generation is the bottleneck—even this weakest configuration leads the strongest baseline by 3.9 to 4.7 points. The gains therefore do not reduce to the proprietary generator.

¹The same DeepSeek-V4-flash model appears in Table 1 as the `v4-flash` direct-LLM baseline; here it acts as the skill generator, not as the answering model.

Skill Creator Ψ	IID							OOD						
	Hotpot	Trivia	AIME	MedQA	WShop	ALFW	SWE	MuSQ	NQ	MATH	GPQA	HEval	SciW	M2W
	EM	EM	Acc	Acc	SR	SR	Res	EM	EM	Acc	Acc	pass@1	Succ	Step Acc
Claude Opus 5	90.63	96.09	76.67	92.19	96.09	96.88	53.91	81.25	82.81	96.09	89.06	98.44	60.94	52.34
Qwen3.8-27B	89.84	94.53	60.00	87.50	87.50	89.06	46.88	79.69	82.81	89.06	85.94	96.09	52.34	45.31
DeepSeek-V4-flash	90.63	96.09	63.33	89.06	91.41	92.97	47.66	81.25	82.81	94.53	87.50	97.66	54.69	48.44
SkillFlow (Full)	92.19	96.09	70.00	92.19	93.75	96.09	52.34	85.16	82.81	96.09	89.84	98.44	57.81	51.56
<i>Strongest baseline (run 1)</i>	89.84	92.97	63.33	90.63	82.81	85.16	43.75	79.69	80.47	93.75	84.38	96.88	47.66	41.41

Table 3: Swapping the Skill Creator Ψ for three generators from different model families, with every other component held fixed. *SkillFlow (Full)* is run 1 of the reported system, whose Ψ is GPT-5.5. For reference, the last row gives the best run-1 score in each column among the non-SkillFlow methods of Table 1. Bold marks the best of the four SkillFlow rows.

Generator capability sets the level: averaged over the fourteen benchmarks the variants sit at +0.64 (Opus 5), −2.60 (DeepSeek-V4-flash) and −4.84 (Qwen3.8-27B) relative to run 1 of the reported system, with a spread of 16.67 points between best and worst on AIME. The curation loop therefore transmits creator quality rather than capping it: a stronger generator lifts the system further (+6.67 on AIME with Opus 5) instead of saturating it.

Each row is a full training run scored on the same evaluation protocol as Table 1, so this is a held-out measurement of creator quality rather than an LLM-as-judge assessment. Isolating *which* property of a generator drives the gap—prompt, temperature or seed at a fixed generator—is a separate study.

G ε -SMOOTHING ANALYSIS

When $R(\tau)$ can be zero (e.g., failed orchestration), strictly positive smoothing keeps those terminals within the supported logarithmic TB formulation. We use the following additive smoothing.

Definition 21 (ε -Smoothing). *We define the smoothed reward as:*

$$\tilde{R}(\tau) := R(\tau) + \varepsilon_{\min}, \quad (92)$$

where $\varepsilon_{\min} > 0$ is a small constant.

Proposition 31 (Ordering Preservation). *Smoothing preserves the strict reward ordering: for any τ_1, τ_2 with $R(\tau_1) > R(\tau_2) \geq 0$:*

$$\tilde{R}(\tau_1)^\beta > \tilde{R}(\tau_2)^\beta, \quad (93)$$

hence the relative quality ranking of trajectories is preserved.

Proof. Since $R(\tau_1) > R(\tau_2) \geq 0$, we have:

$$\tilde{R}(\tau_1) = R(\tau_1) + \varepsilon_{\min} > R(\tau_2) + \varepsilon_{\min} = \tilde{R}(\tau_2). \quad (94)$$

Both $\tilde{R}(\tau_1), \tilde{R}(\tau_2) > 0$. Since $x \mapsto x^\beta$ is strictly monotone increasing on $\mathbb{R}_{>0}$ for $\beta > 0$:

$$\tilde{R}(\tau_1)^\beta > \tilde{R}(\tau_2)^\beta. \quad (95)$$

□

Proposition 32 (Flow Perturbation Bound). *For $\beta \geq 1$, by the mean-value theorem applied to the convex function $x \mapsto x^\beta$ on $[R(\tau_x), \tilde{R}(\tau_x)]$,*

$$|\tilde{R}(\tau_x)^\beta - R(\tau_x)^\beta| \leq \beta \varepsilon_{\min} \max(\tilde{R}(\tau_x), R(\tau_x))^{\beta-1}, \quad \beta \geq 1. \quad (96)$$

Note that ε -smoothing is not a negligible numerical patch but the mechanism that keeps zero-reward trajectories inside the support: with $R \in [0, 1]$, the pair $(\varepsilon_{\min}, \beta)$ bounds the target terminal-probability ratio by $((1 + \varepsilon_{\min})/\varepsilon_{\min})^\beta$ and is thus an explicit diversity–quality control.

Remark 12 (Practical Choice of β). *SkillFlow uses $\beta \geq 1$ in all reported experiments; the perturbation bound therefore always falls in the regime of Eq. (96).*

H GRADIENT VARIANCE ANALYSIS

This section analyzes the per-trajectory TTB regression gradient and its residual-dependent variance bound, then distinguishes that bound from general policy-gradient behavior.

H.1 TTB SELF-ANNEALING THEOREM WITH CHAIN RULE EXPANSION

Theorem 33 (TTB Gradient Self-Annealing). *For a recorded trajectory with fixed discrete states, candidate supports, state-rule outputs, and length T , the regression gradient satisfies*

$$\nabla_{\theta} \mathcal{L}_{TTB}(\tau) = \frac{2\Delta(\tau)}{T^2} \nabla_{\theta} \Delta(\tau). \quad (97)$$

Assume the trainable forward/partition block θ and backward block ϕ are disjoint, with a frozen shared backbone. In the action-only model,

$$\nabla_{\theta} \Delta(\tau) = \nabla_{\theta} \log Z_{\theta}(q) + \sum_t [\nabla_{\theta} \ell_{\theta}(a_t; r_t, s_{t-1}) - \mathbb{E}_{a \sim \pi_{\theta}(\cdot | r_t, s_{t-1})} \nabla_{\theta} \ell_{\theta}(a; r_t, s_{t-1})]. \quad (98)$$

The resulting complete expression is

$$\nabla_{\theta} \mathcal{L}_{TTB}(\tau) = \frac{2\Delta(\tau)}{T^2} \left[\nabla_{\theta} \log Z_{\theta}(q) + \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | r_t, s_{t-1}) \right], \quad (99)$$

where π_{θ} is normalized as in Eq. (4). If $\|\nabla_{\theta} \Delta\|$ stays bounded, $\Delta(\tau) \rightarrow 0$ implies $\|\nabla_{\theta} \mathcal{L}_{TTB}(\tau)\| \rightarrow 0$.

Proof. The chain rule gives

$$\nabla_{\theta} \left(\frac{\Delta(\tau)^2}{T^2} \right) = \frac{2\Delta(\tau)}{T^2} \nabla_{\theta} \Delta(\tau). \quad (100)$$

For the forward log-normalizer,

$$\nabla_{\theta} \log C_F(s, r) = \frac{\sum_a e^{\ell_{\theta}(a; r, s)} \nabla_{\theta} \ell_{\theta}(a; r, s)}{C_F(s, r)} = \mathbb{E}_{a \sim \pi_{\theta}(\cdot | r, s)} \nabla_{\theta} \ell_{\theta}(a; r, s).$$

Substitute this into Definition 9. The reward and backward block do not depend on θ for the fixed sample, giving Eq. (98) and hence Eq. (99). The backward derivative similarly retains its own normalizer:

$$\nabla_{\phi} \Delta(\tau) = - \sum_t [\nabla_{\phi} b_{\phi}(s_t, s_{t-1}) - \mathbb{E}_{p \sim P_B(\cdot | s_t)} \nabla_{\phi} b_{\phi}(s_t, p)]. \quad (101)$$

Finally, the norm is bounded by $2|\Delta| \|\nabla_{\theta} \Delta\|/T^2$, proving the conditional self-annealing statement. Shared trainable parameters would require derivatives of every affected forward, backward, and partition term rather than the disjoint-block expansion. $\square$

For a marginal stochastic edge, differentiating Eq. (42) gives the posterior-weighted derivative of its joint event probabilities; the reasoning factor is included. On an expanded graph this is simply the sum of the derivatives of the reasoning and action log-probabilities, with the fixed chance log-probability retained in Δ even though its direct derivative vanishes.

These are gradients of the loss for a recorded batch. If τ is drawn from a parameter-dependent rollout measure μ_{θ} , then

$$\nabla_{\theta} \mathbb{E}_{\mu_{\theta}} \mathcal{L}_{TTB} = \mathbb{E}_{\mu_{\theta}} \nabla_{\theta} \mathcal{L}_{TTB} + \mathbb{E}_{\mu_{\theta}} [\mathcal{L}_{TTB} \nabla_{\theta} \log \mu_{\theta}].$$

The regression update conditions on the collected batch and concerns the first term, not an assertion that sampling is differentiated through. The zero-set results use a full-support measure; none of these identities guarantees that the training dynamics attain that set.

H.2 VARIANCE BOUND AND COMPARISON TO REINFORCE

Proposition 34 (Variance Bound under Bounded Gradient). *Assume $\|\nabla_\theta \Delta(\tau)\| \leq G_\nabla < \infty$ uniformly over the trajectory distributions being considered, and $T \geq T_{\min} \geq 1$. Then*

$$\|\nabla_\theta \mathcal{L}_{TTB}(\tau)\| = \frac{2|\Delta(\tau)|}{T^2} \|\nabla_\theta \Delta(\tau)\| \leq \frac{2G_\nabla |\Delta(\tau)|}{T_{\min}^2}. \quad (102)$$

Writing vector variance as $\text{Var}(g) = \mathbb{E}\|g - \mathbb{E}g\|^2$,

$$\text{Var}_\tau[\nabla_\theta \mathcal{L}_{TTB}(\tau)] \leq \mathbb{E}_\tau \|\nabla_\theta \mathcal{L}_{TTB}(\tau)\|^2 \leq \frac{4G_\nabla^2}{T_{\min}^4} \mathbb{E}_\tau[\Delta(\tau)^2]. \quad (103)$$

Consequently, mean squared residual tending to zero under those distributions implies vanishing variance of this per-trajectory regression gradient.

Proof. The norm bound follows from Theorem 33. Squaring and averaging gives the final inequality. The first inequality uses $\text{Var}(g) = \mathbb{E}\|g\|^2 - \|\mathbb{E}g\|^2 \leq \mathbb{E}\|g\|^2$. The limiting implication follows with uniform G_∇ and $T_{\min}$. It requires the mean-squared-residual premise, not merely pointwise stationarity of the optimizer. $\square$

For comparison, a REINFORCE estimator with an action-independent baseline b has the form $g_{\text{PG}} = (R - b)\nabla_\theta \log \pi_\theta(\tau)$. Its variance depends jointly on rewards, scores, and the baseline, rather than on reward variance alone.

Lemma 35 (REINFORCE Variance Has No Universal Residual Analogue). *For $S(\tau) = \nabla_\theta \log \pi_\theta(\tau)$,*

$$\text{Var}(g_{\text{PG}}) = \mathbb{E}[(R - b)^2 \|S\|^2] - \|\mathbb{E}[(R - b)S]\|^2. \quad (104)$$

Stochastic sampling and heterogeneous rewards alone imply neither a strictly positive lower bound nor a vanishing variance. No universal variance ordering between REINFORCE and TTB follows.

Proof. Equation (104) is the second-moment identity. For a Bernoulli action a with probability $1/2$, parameterized by a logit, the score is $S = a - 1/2$. Set $R = a$. With baseline $b = 0$, the estimator is $1/2$ or 0 with equal probabilities, hence has variance $1/16$. With baseline $b = 1/2$, it is $1/4$ for either action and has variance zero, although the policy is stochastic and rewards differ. Conversely, if an independent fair Bernoulli reward is assigned to the action, the expected reward is constant across policies, yet the baseline- $1/2$ estimator at $P(a = 1) = 1/2$ takes values $\pm 1/4$ and has variance $1/16$. Thus even an optimum of expected reward need not force the estimator variance to vanish. The TTB conclusion is instead the conditional bound of Proposition 34. $\square$

I PROOF OF PROPOSITION 2 (MAIN TEXT)

Proposition 2 (main text) concerns the full-support zero-residual sampling guarantee, reuse of its diagnostic quantities, and the conditional gradient-variance bound. We establish these parts with the same kernels as in Sections 4.1–4.2.

Proof. (i) Residual-dependent gradient variance. Theorem 33 gives $\nabla_\theta \mathcal{L}_{TTB} = (2\Delta/T^2)\nabla_\theta \Delta$, including both local normalization derivatives. Under the uniform bound in Proposition 34, $\text{Var}(\nabla_\theta \mathcal{L}_{TTB}) \leq 4G_\nabla^2 \mathbb{E}[\Delta^2]/T_{\min}^4$. Therefore $\mathbb{E}[\Delta^2] \rightarrow 0$ implies vanishing gradient variance. The premise is not deduced from the definition of the loss, and Lemma 35 makes no universal claim about another optimizer’s variance.

(ii) Terminal and path sampling. If the residual vanishes on every supported complete path, Theorem 9 applies with terminal reward $\tilde{R}(x)^\beta$. The forward kernel is the normalized action kernel under deterministic causal state rules, or the complete stochastic kernel of Eq. (42); the backward is normalized at a fixed child. Exponentiating and summing over paths gives

$$Z_\theta(q)P_F(x | q) = \tilde{R}(x)^\beta \sum_{\tau \rightarrow x} P_B(\tau | x) = \tilde{R}(x)^\beta. \quad (105)$$

Summing over terminals then identifies $Z_\theta(q) = \sum_x \tilde{R}(x)^\beta$. For each path separately,

$$P_F(\tau | q) = \frac{\tilde{R}(s_T)^\beta P_B(\tau | s_T)}{Z_\theta(q)}. \quad (106)$$

Thus path ordering is allocated by P_B , not assigned an independent copy of the terminal mass. The $\text{Lin}(x)$ tilt in Proposition 29 belongs to the different objective on raw-history terminals. On the finite supported graph, zero expected TTBB loss under full-support μ implies the pathwise premise by Lemma 10. Attainability by the stochastic rollout family requires Eq. (43).

(iii) *State-flow and prefix diagnostics*. The constructive part of Theorem 9 yields Detailed Balance, hence

$$I(t) = \frac{P_F(s_t | s_{t-1})}{P_B(s_{t-1} | s_t)} = \frac{F_q(s_t)}{F_q(s_{t-1})}, \quad F_q(s_T) = Z_\theta(q) \prod_{t=1}^T I(t). \quad (107)$$

The product recovers terminal *state* flow, whereas trajectory flow equals $F_q(s_T)P_B(\tau | s_T)$. Before exact balance, Eq. (63) defines the path-indexed estimate without assuming equality of different prefixes. Proposition 20 computes these estimates, ratios, and skill summaries from the TTBB evaluations, while Lemmas 22–23 retain the per-query normalization and visit multiplicity. Reusing these quantities requires no additional diagnostic-only policy evaluations; it does not eliminate the cost of forming the kernels. $\square$

J PROOF OF PROPOSITION 3 (MAIN TEXT)

Proposition 3 (main text) separates the algorithmic trigger from the structural invariant and the per-phase flow condition. We prove these three parts in turn.

Proof. (i) Well-defined phase-boundary trigger. For fixed phase support and a full-support measure $\mu^{(k)}$, the representational floor is

$$\bar{\Delta}^{*(k)} := \inf_{\theta, \phi, Z_\theta} \mathbb{E}_{\mu^{(k)}} [\Delta(\tau | \mathcal{S}^{(k)}, \theta, \phi, Z_\theta)^2] \geq 0. \quad (108)$$

Nonnegativity is immediate. If some parameters attain expectation zero, every supported residual vanishes; conversely, exact TB gives zero expectation. These statements concern representability, and an unattained infimum of zero is possible. They do not equate a flat observed training sequence with that infimum.

The practical rule is instead Definition 14. Its denominator is positive, and its comparisons use completed within-phase windows. If its two inequalities hold for the prescribed consecutive comparisons at an eligible check, the algorithm triggers curation by definition. Otherwise it continues the phase. The rule remains defined at zero residual, where its second inequality prevents a plateau edit. Thus the trigger is an explicit procedure, without a conclusion that skill inadequacy caused the observed plateau or that an edit must lower the residual.

(ii) *Preservation of atomic composability.* Assume $\mathcal{S}^{(0)}$ is empty or contains only accepted atomic tips; an empty library satisfies the property vacuously. At phase k , Lemma 30 shows that holding over, retaining, removing, and replacing tips through the sound atomic admission interface preserves the property. The same is true of selecting a capacity-limited subset of accepted new tips. Induction gives atomic composability of every $\mathcal{S}^{(k)}$.

The CGF is defined on nonempty visit multisets; unsupported tips are explicitly held over, so no undefined mean or logarithm is used. Within the supported set, pruning precedence and complement construction make the curation classes disjoint. Failed refinements keep their old atomic versions. These details ensure that the union in Eq. (90) has the invariant used in the induction.

(iii) *Graph structure and conditional flow conservation within each phase.* After an edit, the next phase installs its fixed library, legal support rule, and sufficient canonical state interface. Theorem 27 applies anew: action count increases strictly and all states are reachable from one source. Lemma 26 supplies commuting pairs when its legality conditions hold; non-trivial parent sets need not exist in every phase.

For any positive backward kernel on that graph, Lemma 18 constructs a unique reward-matching graph flow. If the phase’s actual forward/backward kernels satisfy full TB, Theorem 9 identifies this flow and gives, at each internal non-source state,

$$F_q(s) = \sum_{s' \in \text{Ch}(s)} F_q(s \rightarrow s') = \sum_{p \in \text{Pa}(s)} F_q(p \rightarrow s). \quad (109)$$

The source and terminals obey the boundary definitions in Appendix A.1. For stochastic execution, the forward kernel must additionally be realizable under its fixed chance transitions. Holding a library fixed makes the graph fixed; it does not make arbitrary learned kernels balanced.

Warm-starting carries numerical parameters into the next phase. Since its reachable terminal set can change, its exact partition and internal flow generally change as well; the next phase’s zero-residual condition must be established separately. Therefore the structural invariant holds inductively, while the reward-matching guarantee is conditional within each phase, not an optimizer or capability-growth guarantee across phases. $\square$

K BACKWARD POLICY DESIGN AND IMPLEMENTATION

This section specifies the fixed-child scoring interface of P_ϕ and its connection to normalized complete-action probabilities.

K.1 HINDSIGHT CONDITIONING

The backward evaluates predecessor choices using the state after the action, while the forward selects an action from the state before it.

Definition 22 (Hindsight-Enriched Candidate Context). *For a fixed child x and a legal candidate parent p , define*

$$h(x, p) := \text{can}(x) \oplus \text{can}(p). \quad (110)$$

Canonical child records include the retained execution evidence. The scorer evaluates the candidate predecessor action or predecessor serialization against this context. All data used for one score are determined by (x, p) , and the candidate contexts for every parent are constructed by the same rule.

Unlike a context built solely from the *actual* pre-action history, this construction defines the whole vector at fixed x . The vector is normalized only after all legal candidates have been scored.

Lemma 36 (Information Asymmetry and Credit Signal). *In the action-only model, let $p = s_{t-1}$, $x = s_t$, and let a_t be the occurrence identifying their edge. Then*

$$\log I(t) = \log \pi_\theta(a_t \mid f^{(k)}(p), p) - b_\phi(x, p) + \log \sum_{p' \in \text{Pa}(x)} e^{b_\phi(x, p')}.$$

This compares forward selection at p with normalized reverse attribution at x . At full TB it is $\log F_q(x) - \log F_q(p)$; at finite residual it is an increment of the prefix estimate.

Proof. Substitute the fixed-child softmax into Definition 11. The balanced identity is Corollary 19; the finite-residual identity follows from Eq. (63). Post-action information can affect the backward vector, but the algebra does not imply that hindsight strictly improves prediction or that the ratio is a success probability. $\square$

K.2 THINK-ACTION SEPARATION

The action-only formulation separates the representation used to score a decision from the graph probability assigned to that decision.

Definition 23 (Action Token vs. Reasoning Token). *Each turn contains reasoning text r_t and the structured complete action $a_t = (\alpha_t, o_t)$. Let $\mathcal{A}_t$ be the action-token positions, including the end marker, and $K_t = |\mathcal{A}_t|$. Under the deterministic state rule $r_t = f^{(k)}(s_{t-1})$,*

$$\log \pi_\theta(a_t \mid r_t, s_{t-1}) = \frac{1}{K_t} \sum_{j \in \mathcal{A}_t} \log p_\theta(\text{token}_j \mid \text{token}_{< j}, r_t, \text{can}(s_{t-1})) - \log C_F(s_{t-1}, r_t). \quad (111)$$

The reasoning text determines the scoring context but has unit conditional mass as a state-rule output. A sampled reasoning string instead has its own ρ_θ factor, which is retained or marginalized as specified in Lemma 13.

Thus action-only TB is not obtained by masking sampled reasoning tokens after rollout. A state rule is defined on every state before action selection and is held fixed within the phase; fixing realized future text, or stopping its gradient, does not provide the same probability model. Likewise, excluding chance tokens from the gradient does not remove their probability from a stochastic TB residual.

K.3 IMPLEMENTATION DETAILS

Architecture P_ϕ shares the frozen base LLM (Qwen3.5-9B) with π_θ but uses a separate LoRA adapter. Named adapter selection separates the forward and backward scoring blocks without changing the common backbone.

Adapter Configuration ϕ -LoRA uses rank 16 with $\alpha = 32$, targeting $\{q, v\}$ projections in attention layers. The θ -LoRA has rank 64 and targets $\{q, k, v, o\}$.

Candidate Construction At each child, enumerate the legal parents from maximal occurrence deletions on the trace DAG, or from the legal predecessor relation on the richer stochastic graph. Construct $h(x, p)$ for every candidate, not only the sampled parent. Canonical action serialization removes duplicates from the forward support; candidate construction is fixed as a function of the phase state.

Token-Level Conditioning The autoregressive model scores each complete candidate according to Eq. (36); its action tokens do not condition on later candidate tokens. This produces the score $b_\phi(x, p)$, not yet a parent probability. Subtracting the log-sum-exp over *all* legal parents produces $\log P_B(p \mid x)$. Forward scoring similarly subtracts $\log C_F$ over complete actions before a categorical draw. The loss and sampler use the same supported distributions.

Probability and Gradient Interface Store the selected edge log-probabilities together with their local normalizers. Both normalizers remain differentiable in TTB, as in Eqs. (98)–(101). Any approximate candidate or marginal-probability scheme defines a different approximation unless its missing probability is accounted for; the exact theorem concerns the specified normalized kernels.

L COMPARISON PROTOCOL

Because SkillFlow adds components—a backward policy, a partition head, and an LLM-based Skill Creator—a comparison is only informative if every method is given the same experience, the same tools and the same optimisation budget, so that, in Figure 5(a), the objective is the sole free variable. Exact values are stated below so that the comparison can be replicated without guesswork.

L.1 SHARED CONFIGURATION

All methods share the following fixed settings. Every trained method in Table 1 and Figure 5(a) runs from the same pinned artefacts and the same budget:

Held identical across methods	Value
Supervisor backbone	Qwen3.5-9B, one pinned revision (SHA-256 verified)
Frozen Executor	the same Qwen3.5-9B instance; no method uses a stronger Executor
Initial skill library	empty for every method
Tool registry and permissions	identical set, no method-specific tools
Training questions	same pool, same order, same seed within each run
Trajectories per question	4 (Tree-GRPO: its published 6 branches)
Questions per optimiser update	7 (i.e. 28 trajectories per update)
Optimiser updates	250; results read from the same update index
Independent runs	five for Table 1; the other results use run 1
Max episode length T	12 steps
Sampling at rollout	temperature 1.0, top- p 1.0, top- k disabled
Trainable parameters	LoRA θ rank 64 (q, k, v, o); LoRA ϕ rank 16 (q, v), SkillFlow only
Optimiser	AdamW, learning rate 1.0×10^{-4} , max grad-norm 3.0
Context and observation budget	uncapped for every method
Evaluation	same harness, same official evaluators, same instances
Hardware	4× NVIDIA A100-SXM4 (80 GB) for every run
Reported checkpoint	the final update (250); no best-of-training selection

In the objective-level comparison of Figure 5(a), the only intended difference is the policy objective itself. Because no method is allowed to select its own best checkpoint, none benefits from a longer effective search over training time.

L.2 BASELINE IMPLEMENTATION

Baselines fall into two groups and are handled differently.

Objective-level baselines (Figure 5(a)). GRPO[‡], Tree-GRPO and HCAPO are training objectives rather than complete systems, so they are instantiated against the shared trainer with a versioned implementation identifier and swapped in for TTB. The Supervisor, Executor, library, rollout loop, tool registry and evaluation path are then literally the same code for all four cells; the configuration files differ only in the objective name and the output path. Each objective follows its own published formulation: Tree-GRPO uses the authors’ branching parameters ($M=2$ roots, $N=2$ expansions, $L=1$ round, i.e. $G = M(LN+1) = 6$ branches per question), and HCAPO uses its hindsight-credit estimator with the authors’ temporal smoothing and do-no-harm masking. We keep those values rather than forcing our own, and note that Tree-GRPO’s branching gives it more rollouts per question than the 4 granted to every other method.

System-level baselines (Table 1). AFlow, AgentFlow, FlowSteer and SkillIRL are run from their authors’ original public code, with each method’s recommended method-specific hyperparameters rather than settings imposed by us, so no baseline is penalised by a configuration chosen for SkillFlow. Our adaptation is confined to two things: the interface layer—model loading, environment and tool bindings, data reading and the evaluation entry point, so that every method sees the same Executor, the same instances and the same scorer—and the shared budget of Appendix L.1, which unifies rollouts per question, maximum episode length, sampling parameters, and LoRA and optimiser settings. No method’s objective, advantage estimator, or update rule is modified.

L.3 WHAT THIS COMPARISON DOES NOT CONTROL

Three things lie outside what the protocol controls. Table 1 averages five runs; the other tables report run 1. The protocol equalises experience and hardware but not tuning: each baseline keeps its authors’ recommended method-specific hyperparameters, while SkillFlow is read at its final update rather than at its best. Tree-GRPO’s larger branching factor is not budget-matched, as noted above.

L.4 ENVIRONMENTS AND EVALUATION HARNESSES

One Supervisor, one Executor, fourteen benchmarks. A single trained Supervisor is evaluated on all 14 benchmarks; no per-benchmark checkpoint, prompt or hyperparameter is selected. The frozen Executor is likewise the same model everywhere—Qwen3.5-9B, the same size as the Supervisor—so

at no point does the 9B orchestrator delegate to a stronger model. The third-party models of §5 enter only as *alternative* Executors in that study; they are never used for the numbers in Table 1.

Task environments are the official ones. What the Executor acts *on* differs by task family, and in every case we use the benchmark’s own released environment rather than a reimplement: the official WebShop and ALFWorld simulators, the official ScienceWorld and Mind2Web environments, the official retrieval corpora for the QA benchmarks, and, for SWE-bench, the actual repositories checked out at each issue’s base commit. Tool access is a single shared registry—retrieval, Python execution, file reading and editing, code search, and test execution—exposed identically to every method; no method receives a benchmark-specific tool.

SWE-bench harness. Because SWE-bench is the benchmark whose protocol most affects the reported number, we state it in full. We use SWE-bench *Verified* and the official `swebench` package (v4.1.0), taking per-repository environment and test specifications from the official `MAP_REPO_VERSION_TO_SPECS` rather than defining our own. For each instance the repository is checked out at its base commit and its environment built from that official specification; the model patch is applied with `git apply`, falling back to a three-way merge, and an instance whose patch fails to apply is scored as unresolved rather than retried; the benchmark’s own test patch is then applied and its own test command executed (`pytest` for most repositories, the project’s `runtests.py` for Django). Each evaluation runs in its own `git` worktree, which is discarded afterwards, so instances cannot contaminate one another.

The harness runs as a separate service: the training host does not permit container execution, so it is hosted externally and the trainer submits a patch and receives a verdict. The dataset, environment specifications, patch-application semantics and test commands are the official ones; reproducing our SWE-bench numbers therefore requires standing up the same harness alongside the training code.

L.5 DATA SPLITS AND OOD HYGIENE

Splits are defined at the level of *questions*, not rows. Two rows belong to the same group when they share a source and a case-folded, whitespace-normalised question string; grouping before splitting prevents a paraphrase or duplicate of a test question from appearing in training. Group assignment is a deterministic hash of the group identifier, so the split is reproducible from the manifest alone and is not tuned. Three disjoint roles are used: a training pool, a development pool consulted only by skill evolution, and a held-out test pool used exclusively for the reported numbers. Exact-group overlap between all three pairs is zero.

The seven OOD benchmarks are never seen during training in any form, and the skill library is frozen at end-of-training before OOD evaluation, so no OOD instance can influence library content. Each benchmark contributes 128 evaluation instances drawn at random from its held-out pool (for GPQA Diamond, 128 of its 198 questions), except AIME, tested on all 30 problems of AIME 2026 after training only on earlier exams; the sampled instance lists are released with the code, and Table 1 entries are means over five runs.

M DATASET DETAILS

We evaluate SkillFlow on 14 public benchmarks covering four task categories: question answering, mathematical reasoning, interactive decision making, and code generation. Seven datasets are used as in-distribution (IID) benchmarks for training and evaluation; the remaining seven are held out as out-of-distribution (OOD) generalization tests, with the skill library frozen at end-of-training.

M.1 IN-DISTRIBUTION DATASETS

- **HotpotQA** (Yang et al., 2018): A large-scale multi-hop QA corpus where each question requires reasoning across multiple Wikipedia paragraphs to derive the answer.
- **TriviaQA** (Joshi et al., 2017): A reading-comprehension dataset of trivia question–answer pairs paired with evidence documents.
- **MedQA** (Jin et al., 2021): A multiple-choice medical QA benchmark drawn from professional medical-licensing examinations, requiring multi-step clinical reasoning.

- **AIME**: American Invitational Mathematics Examination problems, a difficult mathematical-reasoning benchmark; training uses earlier AIME exams, and evaluation uses all 30 problems of AIME 2026.
- **WebShop** (Yao et al., 2022a): A simulated e-commerce environment where the agent interprets a natural-language instruction, navigates web pages, and selects the matching product.
- **ALFWorld** (Shridhar et al., 2020): A text-based interactive environment grounded in household tasks; the agent issues actions in language and receives observation feedback.
- **SWE-bench** (Jimenez et al., 2023): A benchmark of real-world GitHub issues that require generating code patches whose application resolves the issue.

M.2 OUT-OF-DISTRIBUTION DATASETS

- **MuSiQue** (Trivedi et al., 2022): A composite multi-hop QA dataset built by chaining single-hop questions, designed to stress reasoning composition.
- **NQ-Open**: An open-domain QA derivative of Natural Questions, where the model must produce free-form answers from user-issued queries.
- **MATH-Hard**: The hard subset of competition mathematics problems (algebra, geometry, number theory, etc.), evaluated by exact-match correctness on the final answer.
- **GPQA Diamond**: The most challenging split of GPQA, containing graduate-level science questions across physics, chemistry, and biology.
- **HumanEval** (Chen et al., 2021): A code-generation benchmark of hand-written Python problems with hidden unit tests; correctness is evaluated by execution.
- **ScienceWorld**: A text-based interactive science environment that requires the agent to perform multi-step experiments and reason over physical-world dynamics.
- **Mind2Web**: A web-navigation benchmark where the agent performs multi-step actions across real-world websites to complete user tasks.

N BASELINE DETAILS

We compare SkillFlow against four categories of baselines. Unless otherwise stated, all baselines that involve fine-tuning or reinforcement learning use the same Qwen3.5-9B backbone and the same training data as SkillFlow, isolating the orchestration objective and skill-evolution mechanism as the source of any performance gap.

N.1 EXECUTORS

The transferability study of §5 keeps the trained Supervisor fixed and swaps the pluggable executor $\mathcal{M}_{\text{exec}}$ for six frozen third-party models: `claude-sonnet-4-6`, `kimi-k2-0905-preview`, `gpt-5.4-mini`, `glm-5.1`, `deepseek-v4-pro` and `gemini-3-flash-preview`. Training itself uses either Qwen3.5-9B or `4o-mini` as the executor; Figure 4(b) contrasts the two.

N.2 DIRECT-LLM BASELINES

- **Qwen3.5-9B**: The same backbone we use as Supervisor, queried directly without any orchestration training, providing a faithful no-orchestration reference point.
- **v4-flash** (DeepSeek-V4-flash): A strong instruction-tuned LLM, queried with ReAct-style prompting and no orchestration training as a high-capacity baseline.

N.3 FINE-TUNING BASELINES

- **SFT (Qwen3.5-9B)**: Supervised fine-tuning on demonstration trajectories of orchestration plans, with no exploration or reward feedback.

- **GRPO[†] (Qwen3.5-9B)** (Shao et al., 2024): Group Relative Policy Optimization applied directly to the backbone, using terminal rewards and, like SkillFlow, an empty initial skill library. This is the fine-tuning baseline of Table 1; it should not be confused with the in-framework variant GRPO[‡] of §5, which retains the entire SkillFlow architecture—Supervisor, dynamic library, curation operator—and only substitutes GRPO for the TTB objective. The latter is correspondingly much stronger.

N.4 SEARCH-BASED WORKFLOW BASELINES

- **AFlow** (Zhang et al., 2024): A workflow-search method that explores compositions of pre-defined operators using MCTS guided by an LLM judge, with no parameter learning.

N.5 RL AGENT BASELINES

- **AgentFlow** (Li et al., 2025b): An in-the-flow agentic system that jointly optimizes planning and tool use under reinforcement learning over multi-turn interactions.
- **FlowSteer** (Zhang et al., 2026b): An end-to-end RL framework for interactive agentic workflow orchestration without evolving its skill library.
- **SkillRL** (Xia et al., 2026): A skill-augmented RL agent that grows its action space using heuristic skill-distillation triggers, representative of the dynamic-library RL paradigm.

O EVALUATION METRICS

We adopt task-appropriate evaluation metrics consistent with each benchmark’s standard protocol.

F1 Score. For QA-style benchmarks (HotpotQA, TriviaQA, MuSiQue, NQ-Open), we report the token-level F1 between the predicted answer y_i and the ground truth y_i^* after standard text normalization:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad (112)$$

where $\text{Precision} = |\text{tok}(y_i) \cap \text{tok}(y_i^*)| / |\text{tok}(y_i)|$ and $\text{Recall} = |\text{tok}(y_i) \cap \text{tok}(y_i^*)| / |\text{tok}(y_i^*)|$.

Exact Match (EM) / Accuracy. For the QA benchmarks we additionally report answer EM, the fraction of predictions that equal the ground truth after the same normalization. For mathematical reasoning (AIME, MATH-Hard) and multi-choice QA (MedQA, GPQA Diamond), we report exact-match accuracy after canonicalization:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\text{norm}(y_i) = \text{norm}(y_i^*)), \quad (113)$$

where $\text{norm}(\cdot)$ standardizes whitespace, casing, and final-answer extraction.

Average Score and Success Rate. For interactive decision-making (WebShop, ALFWorld, ScienceWorld), we follow each environment’s standard evaluator: *Average Score* reports the environment-defined task-specific score, while *Success Rate (SR)* is the fraction of episodes that end fully solved.

Resolved Rate. For SWE-bench (Jimenez et al., 2023), we report the fraction of issues whose generated code patch passes the held-out test suite (the standard “resolved” metric).

pass@1. For HumanEval (Chen et al., 2021), we report *pass@1*, the fraction of problems whose first generated program passes all hidden unit tests, with one program sampled per problem.

Step-Level Metrics for Web Navigation. For Mind2Web, we report *Step Accuracy* (fraction of steps with correctly predicted target element and action) and *Action F1* (token-level F1 over the predicted action string against the ground truth).

Pass@ K and strategy diversity. In Figure 5(d) every query is attempted five times; Pass@ K is the fraction of queries solved within the first K attempts, so Pass@1 is the single-attempt score. Div is the number of structurally distinct successful workflows (distinct step arrangements) among all trajectories sampled for a query, averaged over queries and pooled over the 14 benchmarks.

Aggregate rows in Table 1. The Avg. rows average each metric family within the IID and OOD halves: EM over the QA benchmarks (HotpotQA, TriviaQA; MuSiQue, NQ-Open); $F1$ over the same QA benchmarks, with Mind2Web action F1 included on the OOD side; $Acc./Pass$ over every remaining entry—accuracy (AIME, MedQA; MATH-Hard, GPQA), success rate and average score (WebShop, ALFWorld; ScienceWorld), resolved rate (SWE-bench), pass@1 (HumanEval) and step accuracy (Mind2Web). $\Delta \uparrow$ is the absolute gain of SkillFlow over the untrained Qwen3.5-9B on the same row. Each benchmark is evaluated on 128 instances (all 30 for AIME 2026), five runs each.

P COMPUTATIONAL RESOURCES

We report the hardware, the run-1 wall-clock and GPU-hour cost of training SkillFlow on Qwen3.5-9B, and the corresponding cost of each objective-level baseline under the shared protocol of Appendix L.

P.1 HARDWARE

All training and on-policy rollout were performed on a single server with $4 \times$ NVIDIA A100-SXM4 (80 GB) GPUs, 32 logical CPU cores, 549 GB RAM, and 1.6 TB local NVMe. Software: CUDA 12.8; PyTorch 2.9.1 with PEFT (training); vLLM 0.17.1 (executor); SGLang 0.5.9 (supervisor).

P.2 RUN 1: SKILLFLOW ON QWEN3.5-9B

Wall-clock 73 h, $\approx$ 292 GPU-hours, 250 training steps (LoRA checkpoint every 10 steps).

Item	Value
Base model (Supervisor)	Qwen3.5-9B-Instruct
Skill Creator Ψ (frozen)	GPT-5.5
LoRA θ (forward)	rank 64, $\alpha = 128$, on q, k, v, o_proj
LoRA ϕ (backward)	rank 16, $\alpha = 32$, on q, v_proj
Z head (partition fn)	conditioned on the query and its task type, separate optimizer
Optimizer	AdamW (default β_1, β_2); three optimizers (θ, Z, ϕ)
Learning rate	1.0×10^{-4}
Max grad norm	3.0
TTB temperature β	1.0
$\epsilon_{\min}$	0.1
Effective batch	$28 = 7 \text{ questions} \times 4 \text{ trajectories}$
Max episode length T	12
Mean step time	$\approx 17.5 \text{ min/step}$ (rollout + update + LoRA hot-swap)
Peak VRAM (per GPU)	$\approx 70 \text{ GB} / 80 \text{ GB}$ (gradient checkpointing on)

P.3 WHERE THE TIME GOES

Profiling one optimiser step of run 1 gives the following split. Percentages are measured; the seconds are those percentages applied to the mean step time of 17.5 minutes, so the column sums to the 73 wall-clock hours and 292 GPU-hours reported above.

Stage	s/step	Share
Rollout (generation + environment)	782	74.4%
∇_{ϕ} — backward-policy backward pass	124	11.8%
∇_{θ} — forward-policy backward pass	81	7.7%
No-grad rescoring	62	5.9%
Partition head + optimiser	2	0.2%
Total	1051	100%

Three quarters of a step is rollout—generating orchestration trajectories and waiting on the environment—whose budget the shared protocol fixes for every method: trajectories per question, episode length and Executor. What SkillFlow adds to a policy-gradient objective is the backward policy and the partition head, and those are the 11.8% and 0.2% rows: the per-step credit signal of §4.2 costs about one eighth of a step, and the learned Z_θ is nearly free.

P.4 TRAINING COST AGAINST THE OBJECTIVE-LEVEL BASELINES

SkillFlow trains a backward policy and a partition head in addition to the forward policy, so the natural question is what that machinery costs relative to a plain policy-gradient objective. Under the shared protocol of Appendix L—same backbone, same Executor, same rollout budget apart from Tree-GRPO’s branching, same hardware—the four objectives cost:

Objective	GPU-hours	Relative
GRPO [‡]	267	0.91×
SkillFlow (TTB)	292	1.00×
Tree-GRPO	424	1.45×
HCAPO	537	1.84×

Per-method throughput was measured at a fixed question budget and extrapolated linearly to the full 250-update run; the SkillFlow entry is anchored to the 292 GPU-hours measured for run 1 and the others are scaled by the same factor, so the ratios are the primary quantity and the absolute hours inherit the anchor’s accuracy.

Against GRPO[‡], which shares the rest of the architecture, SkillFlow costs roughly 9% more: the backward policy and partition head take about 12% of a step (Appendix P.3), partly offset by SkillFlow’s faster rollouts (Figure 5b). That is the price of the per-step credit signal, and it is well below what the other two credit-assignment baselines pay for theirs: Tree-GRPO costs 45% more than SkillFlow because its branching factor multiplies rollouts per question, and HCAPO 84% more because hindsight advantage estimation requires additional passes over each trajectory. Figure 5(b) repeats these GPU-hours in its top table, with per-call cost against skill-evolution baselines below.

We report GPU-hours, the budget that binds in practice. Total generated tokens and environment calls per method are recorded by the training loop but were not aggregated for these runs.

Q LIMITATIONS

SkillFlow targets one specific question: *can flow-based training drive recursive skill evolution from end-to-end task feedback alone?* Within that scope, the framework couples terminal reward-matching at full balance, reusable flow diagnostics, and plateau-driven curation; the sampling guarantee requires the support, state-sufficiency, and realizability conditions stated above. The method’s effectiveness, however, inherits two properties of the underlying language model that are themselves the subject of separate research lines.

Reliance on long-context memory. Multi-turn orchestration accumulates action and execution records, whose sufficient canonical representation forms the Supervisor’s context; order-dependent raw reasoning logs are not reintroduced into the action quotient. SkillFlow therefore relies on the backbone’s ability to attend to and faithfully use long histories—backbones with weaker long-context fidelity see the per-step credit signal degrade as trajectories grow, since the hindsight backward P_ϕ must condition on increasingly distant context. Improving long-context modeling itself is an active research line whose advances stack with our training recipe but lie outside its scope.

Reliance on backbone reasoning capacity. As shown in §5 (RQ3), SkillFlow lifts every execution backend but cannot replace the underlying base capability—orchestration amplifies what the model already knows about decomposition and tool use rather than installing new reasoning skills. Where the bottleneck is core reasoning rather than orchestration, additional pretraining or distillation is required; this remains complementary to, but outside the scope of, the present paper.