

COLLABFLOW: RECURSIVE SELF-IMPROVEMENT OF AGENT COLLABORATION

Anonymous authors

Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) lets a system improve from its own outcomes; in LLM-based multi-agent systems, Agents refine one another within a task, and outcomes improve how they collaborate across tasks. However, existing multi-agent collaboration leaves this loop open: collaboration is pre-defined at the operator level, topology-only learning keeps verbatim exchange that propagates errors, and reward maximization on a system's own outcomes concentrates on a few teams. To address these challenges, we propose CollabFlow, an RSI system of Learned Agent Collaboration: a trainable Collab-Director constructs teams of complete Agents, a frozen executor runs them, and each round's outcomes retrain the director. Within each round, the edges of a collaboration graph carry protocols of Evidence-Conditioned Communication: a receiver adopts a differing answer only when the sender's evidence is stronger by a margin, so the director learns who communicates and how. Across rounds, we further propose Collaborative Trajectory Balance (CTB), a flow-based objective that credits each team once across its construction orders and targets a reward-proportional distribution over teams, so several good teams stay in play. We also bound how far this self-generated target moves between rounds, which shrinks as records accumulate. On twelve datasets, CollabFlow outperforms all baselines and keeps improving across rounds. Code is available at <https://anonymous.4open.science/r/CollabFlow-631E>.

1 INTRODUCTION

Recursive self-improvement (RSI), in which a system improves from its own outcomes, is becoming a concrete systems problem (Chen et al., 2026a). LLM-based multi-agent systems (Li et al., 2023; Wu et al., 2023; Hong et al., 2024) are a natural setting for it: their Agents already refine one another's answers, and every exchange leaves an execution outcome.

Such a system can improve at two levels, as shown in Figure 1. Within a task, communication is the improvement step: specialized Agents supply complementary evidence and verify one another (Qian et al., 2024; Liu et al., 2023). Across tasks, the outcomes in turn improve how the team communicates, so the team level improves the message level. In practice, however, teams still rely on hand-written roles, fixed protocols, and prompt-level rules (Hong et al., 2024; Qian et al., 2024), which are costly to transfer across new tasks, Agent populations, or environments and which no execution outcome revises, so the loop from outcomes back to collaboration stays open.

To address these issues, three main paradigms of multi-agent orchestration have emerged, as shown in Figure 2. First, workflow orchestration optimizes executable operator sequences for a single executor from execution feedback (Zhang et al., 2025; 2026a;b). Second, adaptive organization

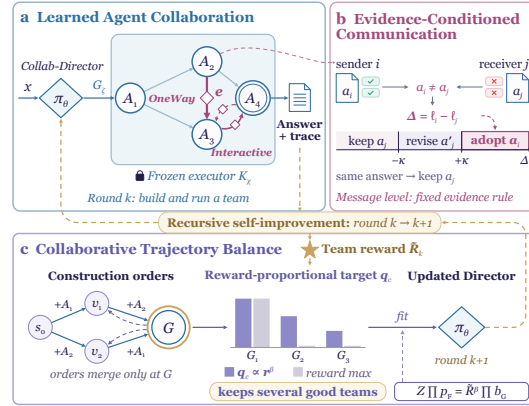


Figure 1: CollabFlow overview. (a) The director π_θ builds a team for executor K_θ . (b) An evidence rule gates each message. (c) CTB retrains π_θ .

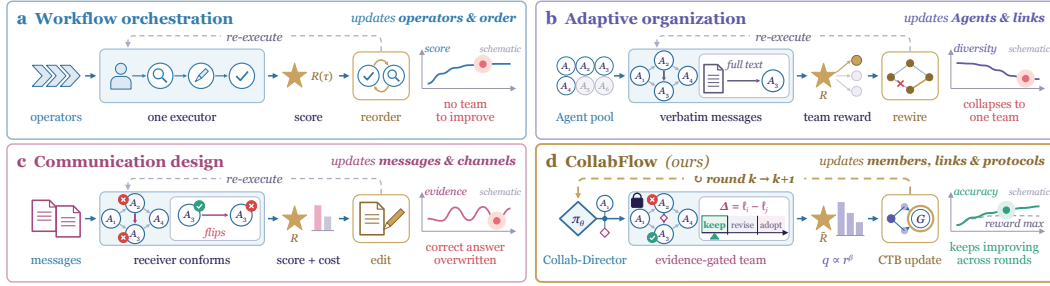


Figure 2: Multi-agent orchestration paradigms as self-improvement loops. Each paradigm runs the same loop of candidate designs, execution, outcome, feedback assessment, update, and re-execution, but updates a different object: (a) workflow orchestration updates operators and their order for one executor; (b) adaptive organization updates which Agents are used and linked; (c) communication design adjusts messages and channels; (d) CollabFlow jointly updates members, links, and edge protocols, trains the Collab-Director with CTB, and admits each message through an evidence gate.

trains an orchestrator via policy gradient or group relative policy optimization (GRPO) to activate or connect Agents (Dang et al., 2026; Tastan et al., 2026; Chen et al., 2026b; Wang et al., 2026). Third, communication design—such as debate, sparse consensus, multi-order messaging, and latent channels (Du et al., 2023; Gou & Liu, 2026; Guan et al., 2026; Liu, 2026)—studies what information Agents exchange and how densely they interact within a task.

However, these methods leave the self-improvement loop open at three points: **(i) Pre-defined, operator-level collaboration**—existing orchestration composes operators for a single executor (Zhang et al., 2025; 2026a) or fixes roles and message routes in advance (Hong et al., 2024; Dang et al., 2026; Tastan et al., 2026), so Agents are never the units that communicate; **(ii) Topology-only learning under verbatim exchange**—learned topologies decide which Agents exchange messages (Tastan et al., 2026; Yu et al., 2026; Jiang et al., 2026), but every received message re-enters the receiver’s reasoning in full, spreading conformity and errors (Ann et al., 2026; Bertalanic & Fortuna, 2026; Chun & Ahmed, 2026) and letting concurrent writers corrupt the environment (Yang et al., 2026a); the policy learns who talks but not how messages are used; **(iii) Collapse of the team-level loop under reward maximization**—trained on its own outcomes, a reward-maximizing policy concentrates on a few teams (Yu et al., 2026; Li et al., 2026), leaving later rounds fewer teams to learn from, while the construction orders of one team split its credit.

To address these challenges, we propose **CollabFlow**, an RSI system of *Learned Agent Collaboration*: a trainable Collab-Director builds teams of complete Agents, a frozen executor runs them, and each round’s outcomes retrain the director. Within a round, Agents refine one another’s answers over a collaboration graph whose edges carry protocols of *Evidence-Conditioned Communication*, which the Collab-Director selects together with the topology, so it learns who communicates and how. On each edge, a receiver adopts a differing answer when the sender’s evidence is stronger by a margin, keeps its own in the reverse case, and otherwise makes one bounded revision; on stateful tasks, a single Executor holds write permission. Across rounds, the Collab-Director is trained via *Collaborative Trajectory Balance* (CTB), a flow-based objective that credits each team once across its construction orders (Malkin et al., 2022; Bengio et al., 2023) and targets a reward-proportional distribution over teams, so every positive-reward team keeps probability. Each round thus improves both the team and the records that define the next reward, and we bound how far this self-generated target moves between rounds, a bound that shrinks as the sampled teams’ records accumulate.

We evaluate on twelve datasets spanning question answering, mathematical and expert reasoning, embodied planning, code generation, and web shopping, with held-out suites probing transfer to unseen task types. CollabFlow outperforms direct-inference, fine-tuning, workflow-search, and agent-RL baselines in and out of distribution, lifts six other frozen executors, and uses the fewest tokens per item. Across evolution steps, most of its gains persist and its team distribution stays diverse and stable, whereas reward-maximizing training shifts more between steps and finds fewer distinct solutions. Ablations confirm each part: removing communication or director training costs the most, and every ablation costs more out of distribution than in it. As Figure 2 shows, CollabFlow turns pre-defined teams into a collaboration that recursively improves itself under an explicit budget.

2 RELATED WORK

Multi-Agent Collaboration and Self-Evolution. Early multi-agent systems fixed roles and message routes by hand (Hong et al., 2024; Qian et al., 2024). Learned orchestration now picks Agents sequentially (Dang et al., 2026), predicts sparse topologies (Tastan et al., 2026; Yu et al., 2026), or designs what Agents exchange (Du et al., 2023; Gou & Liu, 2026). In all three, messages are still exchanged verbatim by convention (Motger et al., 2026), which induces conformity (Ann et al., 2026; Bertalaníč & Fortuna, 2026). Self-evolving systems adapt topologies at inference time through trace auditing (Huang et al., 2026) or refine latent messages over recursion rounds (Zou et al., 2026). CollabFlow makes team communication itself the object of recursive self-improvement: it jointly selects topology and protocols and retrains this choice on its own outcomes.

Recursive Self-Improvement and RL for Agents. Recursive self-improvement studies systems that produce better versions of themselves, mostly improving prompts, harnesses, skills, or weights against a fixed evaluator (Chen et al., 2026a). Agent RL is a common outer step of such loops, using trajectory rewards to sequence Agents (Dang et al., 2026), edit workflow graphs (Zhang et al., 2026a), or evolve skills (Zhang et al., 2026b) under group-relative (Shao et al., 2024) or counterfactual (Chen et al., 2026b) credit. These objectives maximize reward and collapse to one mode, narrowing what later rounds learn from. Building on trajectory balance (Malkin et al., 2022; Bengio et al., 2023), CollabFlow scores the canonical complete-Agent team rather than each construction order.

3 PRELIMINARIES

Definition 1: Collaboration Graph. A collaboration graph is an annotated directed graph $G = (V, E, o)$, where $V \subseteq \mathcal{A}_x$ selects complete Agents with immutable profiles, $E \subseteq V \times V \times \mathcal{P}$ labels each edge with a protocol from the fixed registry $\mathcal{P} = \{\text{FINALONLY}, \text{ONEWAY}, \text{INTERACTIVE}\}$, and o is the output rule. Agent profiles specify roles, tools, and inherited permissions, and stateful teams select exactly one authorized Executor. The loop rebuilds this graph in every round: its edges fix who communicates, and its protocols fix how each message may improve the receiver.

Definition 2: Collaboration Trajectory. Fix a context $c = (x, \chi, k)$ with task x , round k , and a specification χ of Agents, tools, contracts, and separate budgets $\mathbf{B}^{\text{con}} + \mathbf{B}^{\text{exec}} \leq \mathbf{B}^{\text{total}}$ (Appendix B.1). The Collab-Director builds each team through a construction trajectory on sufficient states v_t ,

$$\zeta = (v_0, e_1, v_1, \dots, e_T, z), \quad e_t = (v_{t-1}, u_t, v_t), \quad z \in \{g_G\}_{G \in \mathcal{G}_c} \cup \{\dagger\}, \quad (1)$$

where T includes the accepted stop or the forced abort $\dagger$, and all accepted stops for one canonical team enter the same absorbing node g_G , so a valid trajectory yields $G_\zeta = \text{Can}(\text{Decode}(\zeta))$.

Definition 3: Self-Improvement Trajectory. Across rounds, CollabFlow updates the parameters $\Theta_k = (\theta_k, \phi_k, \psi_k)$ of the director, backward policy, and normalizer, and the reliability snapshot $\mathcal{S}_k = \{(s_k(G), n_k(G))\}_G$, from the batch $\mathcal{B}_k$ of round k :

$$\Theta_{k+1} = \mathcal{U}_{\text{CTB}}(\Theta_k; \mathcal{B}_k, \mathcal{S}_k), \quad \mathcal{S}_{k+1} = \mathcal{S}_k + \text{Count}(\mathcal{B}_k), \quad (2)$$

where Count adds each unique outcome once (Section 4.3), so the teams of round k both retrain the director and define the reward of round $k+1$ through their success counts.

Problem Statement. Let P_θ be the law of construction attempts, $\mathbb{V}$ the event of valid completion, and $\mu_\theta^\mathbb{V}(G | c) = P_\theta(z = g_G | \mathbb{V}, c)$ the resulting team distribution. For a fixed positive reward table r_G and $\beta > 0$, each round pursues the team-level objective

$$\min_{\theta} \mathbb{E}_{x \sim \mathcal{D}_X} D(\mu_\theta^\mathbb{V}(\cdot | c) \| q_c), \quad q_c(G) \propto r_G^\beta, \quad (3)$$

where D is a divergence such as total variation, with context and reward fixed per round.

4 METHODOLOGY: COLLABFLOW

As illustrated in Figure 3, CollabFlow nests two improvement steps in the loop of Eq. 2: within a round, Agents improve one another’s answers under evidence-conditioned protocols, and across rounds, CTB retrains the Collab-Director on its own outcomes (Algorithm C.1). Section 4.1 defines the team, Section 4.2 the message-level step, and Section 4.3 the team-level step.

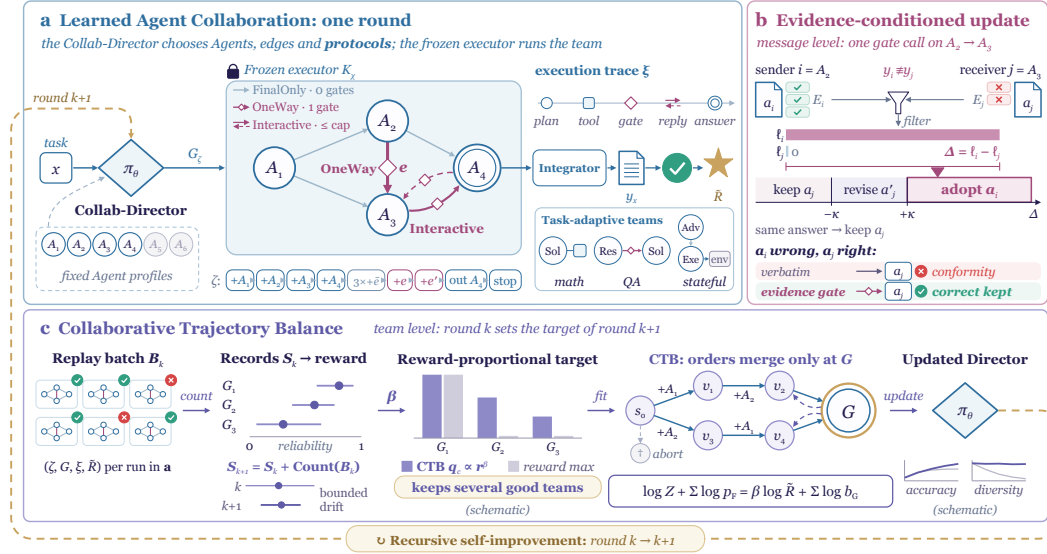


Figure 3: The self-improvement loop of CollabFlow. (a) Within a round, the Collab-Director builds a team from fixed Agent profiles through checked actions, the frozen executor runs it, and the evaluated outcome becomes a replay record (Section 4.1). (b) On each edge, the protocol sets how many evidence gates run, and a fixed evidence rule keeps, adopts, or revises the receiver’s candidate (Section 4.2). (c) Across rounds, CTB merges construction histories only at the complete canonical team and updates the director with the length-normalized residual and a KL term (Section 4.3).

4.1 LEARNED AGENT COLLABORATION: WHAT EACH ROUND CHANGES

As shown in Figure 3a, CollabFlow follows a Director–Executor paradigm: the trainable Collab-Director designs the team and a frozen executor runs its Agents, so each round changes only how teams are designed, from which Agents they use to how those Agents communicate.

Team Construction. A valid trajectory ζ yields a team G_ζ , which the frozen executor K_χ runs:

$$G_\zeta = \text{Can}(G_0 \oplus u_1 \oplus \dots \oplus u_T), \quad (\xi, y_x) \sim K_\chi(\cdot \mid G_\zeta, x; \mathbf{B}^{\text{exec}}). \quad (4)$$

where $\oplus$ applies checked updates, ξ is the execution trace, and failures give $y_x = \perp$. Canonicalization merges histories that differ only in serialization order and keeps Agents, protocols, permissions, and outputs distinct, so a team’s histories share its execution reservation and reward snapshot.

Graph State and Valid Actions. The state s_t records the partial team and construction resources, and v_t adds the accepted history and decision index. The director samples only from the legal set $\mathcal{U}_c(v_t)$ of actions that pass the construction checks at v_t , so no sampled proposal is rejected:

$$s_t = (G_t, \mathcal{A}_x, \mathbf{b}_t^{\text{con}}), \quad v_t = (s_t, H_t, t),$$

$$\bar{\pi}_\theta(u \mid v_t, c) = \frac{\mathbf{1}\{u \in \mathcal{U}_c(v_t)\} \pi_\theta(u \mid v_t, c)}{\sum_{u' \in \mathcal{U}_c(v_t)} \pi_\theta(u' \mid v_t, c)}. \quad (5)$$

Actions select Agents, bind skills, add protocol-labeled edges, select the output, or stop; a stop requires a complete valid team, and a timeout or exhausted budget returns an explicit abort (Appendix A.3).

Marginalization and Decoupling. Conditional on valid construction,

$$P(dy \mid c, \mathbf{V}) = \sum_{G \in \mathcal{G}_c} \mu_\theta^\mathbf{V}(G \mid c) K_\chi(dy \mid G, x; \mathbf{B}^{\text{exec}}). \quad (6)$$

Across rounds, only $\mu_\theta^\mathbf{V}$ changes in Eq. 6, so every change in the system’s output distribution comes from the teams the director builds, not from updates to the executor or the Agents.

Proposition 1. Under checked preservation (C1), every accepted trajectory yields a valid canonical team, and under component completeness and feasible prefixes (C2–C3), every target team keeps a legal construction trajectory in every round (Appendix B.2).

4.2 MESSAGE-LEVEL IMPROVEMENT: EVIDENCE-CONDITIONED COMMUNICATION

As shown in Figure 3b, communication is the inner improvement step: within a round, Agents revise one another’s answers. Verbatim exchange lets every message re-enter the receiver in full, which the DeGroot model (DeGroot, 1974) treats as fixed-weight averaging whose consensus is set by the initial answers rather than by evidence (Lemma B.11). CollabFlow instead lets checkable evidence decide how each message is used, and the director learns how many evidence gates each edge runs.

Evidence-Gated Update. Agent v ends its local episode with a candidate version a_v and evidence records E_v . Its score $\ell_v = \max\{w(e) : e \in \text{Filter}_c(E_v, a_v)\} \cup \{0\}$ keeps the strongest record that still checks this version, with $w = 2$ for a passed executable check and $w = 1$ for retrieval support (Appendix B.4). A directed update $i \rightarrow j$ between differing answers compares the evidence gap $\Delta = \ell_i - \ell_j$ of the sender over the receiver with a margin κ :

$$a_j^+ = a_i \text{ if } \Delta > \kappa, \quad a_j^+ = a_j \text{ if } \Delta < -\kappa, \quad a_j^+ = \text{Revise}_c(a_j, a_i) \text{ if } |\Delta| \leq \kappa, \quad (7)$$

where Revise_c is one bounded, tool-free call in place of a re-execution, and its output is filtered and rescored, so every record stays with the version it checked. With integer scores in $\{0, 1, 2\}$, we fix $\kappa = 1$, the only integer margin that adopts on a two-level gap but revises on a one-level gap. Then a sender whose answer passed a unit test ($\ell_i = 2$) overrides a receiver whose answer is only asserted ($\ell_j = 0$), while two asserted answers trigger a revision; stated confidence never enters Δ .

Who Communicates and How. A protocol p fixes the edge program g_e^p that edge e runs: the identity for FINALONLY, one gate for ONEWAY, and a capped number of gates in alternating directions for INTERACTIVE, stopping at agreement or a decisive update. One fixed-order sweep over the edges of G , topological for acyclic teams and repeated up to a fixed cap for cyclic ones, passes the candidates $\mathbf{a} = (a_v)_{v \in V}$ through the team’s message-level operator

$$\mathcal{C}_G = g_{e_1}^{p_1} \circ \dots \circ g_{e_1}^{p_1}, \quad \mathbf{a}^+ = \mathcal{C}_G(\mathbf{a}), \quad (8)$$

so the director’s edges decide who communicates, their protocols decide how, and the recursion of Section 4.3 improves both (Appendix B.6). On stateful tasks, only the Executor may write and Advisors act through its next decision, so internal writes are serial (Papadimitriou, 1979).

Measurable Harm. Equation 7 also fixes how a correct receiver can be hurt. Let $C = \{z_j = 1, z_i = 0\}$, $J = \{|\Delta| \leq \kappa\}$, the false-adoption rate $\delta_C = P(\Delta > \kappa \mid C, c)$, and the revision error $c_J = P(z_j^+ = 0 \mid J, C, c)$. On C the answers differ, so the three branches partition C :

$$P(z_j^+ = 0 \mid C, c) = \delta_C + P(J \mid C, c) c_J. \quad (9)$$

Choosing $\kappa = 1$ over $\kappa = 0$ removes $P(\Delta = 1 \mid C, c)$ from δ_C by sending one-level gaps to revision, and the same partition gives the rate at which a gate corrects a wrong receiver (Corollary B.8); RQ4 ablates each of these branches. A flipped answer lowers its team’s answer score and posterior success rate, so the next round’s target moves toward teams whose messages help (Section 4.3).

Proposition 2. *With version-bound evidence, ordered gating, and a single writer (S1–S3), evidence-conditioned communication keeps each record with the version it checked, flips a correct receiver only by false adoption or revision error, and serializes internal writes (Appendices B.4, B.5, and B.7).*

4.3 TEAM-LEVEL IMPROVEMENT: COLLABORATIVE TRAJECTORY BALANCE

As shown in Figure 3c, the team-level step retrains the Collab-Director across rounds on the outcomes of the teams it built and must keep several good teams in play. CTB builds on generative flow networks (Bengio et al., 2021; 2023) and trajectory balance (TB) (Malkin et al., 2022), with the complete-Agent team as its scored object: all construction orders of a team end at one canonical terminal, so the team receives its reward once, read from the loop’s own records.

Path Kernels. The backward kernel decides how a team’s credit is shared among the construction orders that reach it (Shen et al., 2023). Let $e_t = (v_{t-1}, u_t, v_t)$ be a complete construction transition. With deterministic validation, the forward kernel is the legal-set policy of Eq. 5, $p_F(e_t \mid v_{t-1}, c) = \bar{\pi}_\theta(u_t \mid v_{t-1}, c)$. The learned backward kernel $b_\phi(e_t \mid v_t, c)$ normalizes over the actual incoming parent–action pairs in the G -ancestral subgraph. Interior states carry their history and thus have one parent (Lemma B.4), so b_ϕ acts only at the canonical terminal, where it picks one of the construction orders $\mathcal{O}(G)$ of G by removing one component at a time; each removal is normalized, so b_ϕ sums

to one over these orders (Appendix C.3). The uniform kernel $1/|\mathcal{O}(G)|$, by contrast, would give all construction orders of a team the same share of its credit.

Balance and Training Surrogate. In round k , for a fixed positive graph reward r_G , ideal balance asks every valid complete path ζ that ends at g_G to satisfy

$$\log Z_\psi(c) + \sum_{t=1}^{T_\zeta} \log p_F(e_t | v_{t-1}, c) = \beta \log r_G + \sum_{t=1}^{T_\zeta} \log b_\phi(e_t | v_t, c). \quad (10)$$

In product form, $Z_\psi(c)$ times the forward probability of ζ equals r_G^β times the backward probability of this order. Since backward probabilities sum to one over the orders of G , summing gives $Z_\psi(c)P_\theta(z = g_G | c) = r_G^\beta$, which fixes each team’s probability and leaves its split over orders to b_ϕ . Let $\rho_{\text{TB}}(\zeta)$ be the left minus the right side of Eq. 10. For the fixed replay batch $\mathcal{B}$ collected in round k and its declared state weighting d_B , CTB minimizes a length-normalized loss with KL weight α_{KL}

$$\mathcal{L}(\theta, \phi, \psi) = \mathbb{E}_{\zeta \sim \mathcal{B}} (\rho_{\text{TB}}(\zeta)/T_\zeta)^2 + \alpha_{\text{KL}} \mathbb{E}_{v \sim d_B} D_{\text{KL}}(\pi_\theta(\cdot | v) \| \pi_{\text{ref}}(\cdot | v)). \quad (11)$$

Dividing the residual by path length puts short and long constructions on one scale without changing the zero-residual solutions. With $\pi_{\text{ref}} = \pi_{\theta_k}$, the director that collected the batch, the KL term is a proximal penalty as in TRPO and PPO (Schulman et al., 2015; 2017): it damps each update, while every balanced director stays a fixed point (Lemma B.24).

Reward-Proportional Teams. At exact valid-path balance with fixed rewards,

$$\mu_\theta^\vee(G | c) = \frac{r_G^\beta}{W_V}, \quad W_V = \sum_{G'} r_{G'}^\beta, \quad Z_\psi(c)P_\theta(V | c) = W_V. \quad (12)$$

At this fixed point, every positive-reward team keeps probability, so the next round starts from a spread of teams. For example, with $\beta = 1$ and rewards 0.6, 0.4, and 0.2, the director samples the three teams with probabilities $1/2$, $1/3$, and $1/6$, the purple bars of Figure 1c, whereas reward maximization places all mass on the first. Under inexact balance, a residual of magnitude at most η on every valid path keeps $\mu_\theta^\vee$ within total variation $\tanh(\eta/2)$ of q_c (Theorem B.17).

Self-Generated Team Reward. The reward of each round is partly generated by the loop itself. With counters frozen before batch k , an execution ξ of team G on task x supplies

$$\tilde{R}_k(G, x, \xi) = \varepsilon_{\min} + r_{\text{ans}}(G, x, \xi) \frac{s_k(G) + 1/2}{n_k(G) + 1} \exp[-L_{\text{rel}}(G)]. \quad (13)$$

Here $\varepsilon_{\min} > 0$ keeps $\log \tilde{R}_k$ finite, $0 \leq r_{\text{ans}} \leq 1$, and $0 \leq s_k \leq n_k$; the fixed structural code, measured in nats, and the reference class define $L_{\text{rel}} \geq 0$. In words, the reward multiplies the answer score by two factors with no tuned weight: the team’s Jeffreys-smoothed posterior success rate (Jeffreys, 1946) and a minimum-description-length structural prior (Rissanen, 1978), so past records decide which teams the next round favors. Counters add each unique outcome once after the batch, so each round generates the target of the next, and the more records a team has, the less one round can move its target; with fresh $\tilde{R}_k$, the ideal r_G is the team’s geometric-mean reward $\exp \mathbb{E}[\log \tilde{R}_k]$, which favors teams that succeed consistently (Theorem B.25).

Proposition 3. *Under exact valid-path balance at fixed rewards ($T1$ – $T3$, $T4'$), CTB samples each canonical team in proportion to its tempered reward among valid completions, and with a fixed answer-score table and structural code, the targets of consecutive rounds differ by a total-variation bound that shrinks as the sampled teams’ records accumulate (Theorem B.16, Corollary B.22).*

5 EXPERIMENTS

Six research questions test the claims of Sections 1 and 4, asking whether learned collaboration wins in distribution, out of distribution and across frozen executors (RQ1–RQ3), and what the paradigm and components, the team-level objective and the recursion contribute (RQ4–RQ6).

Setup. Six in-distribution (IID) datasets train the Collab-Director and six out-of-distribution (OOD) datasets are held out, 128 items each (30 for AIME 2026) (Table 1; Wu et al., 2026; Yang et al., 2018; Joshi et al., 2017; Zuo et al., 2025; Shridhar et al., 2020; Trivedi et al., 2022; Kwiatkowski et al.,

		Baseline	SFT	GRPO	AFlow	Agent+RL		Skill evolution		Ours
Dataset	Metric	Qwen3.5	Qwen3.5	Qwen3.5	Qwen3.5	FlowSteer	Evolving	SkillFlow	SkillIRL	CollabFlow ($\Delta \uparrow$)
(a) In-Distribution (IID) benchmarks										
HotpotQA	EM	46.25 $\pm$ 0.86	47.34 $\pm$ 0.89	49.53 $\pm$ 1.62	53.75 $\pm$ 1.78	53.28 $\pm$ 2.02	60.16 $\pm$ 2.14	60.78 $\pm$ 0.86	58.28 $\pm$ 2.04	63.91 $\pm$ 1.69 (+17.7)
	F1	65.59 $\pm$ 0.89	67.32 $\pm$ 1.46	67.80 $\pm$ 2.02	69.97 $\pm$ 1.50	71.38 $\pm$ 2.18	72.95 $\pm$ 1.05	78.37 $\pm$ 1.95	75.71 $\pm$ 1.96	79.72 $\pm$ 1.96 (+14.1)
TriviaQA	EM	69.69 $\pm$ 1.28	69.53 $\pm$ 1.66	69.84 $\pm$ 1.96	87.34 $\pm$ 1.60	87.34 $\pm$ 1.02	86.88 $\pm$ 1.16	86.56 $\pm$ 1.50	82.97 $\pm$ 0.35	87.97 $\pm$ 1.88 (+18.3)
	F1	74.67 $\pm$ 2.11	77.03 $\pm$ 2.10	76.62 $\pm$ 1.52	89.04 $\pm$ 0.43	88.55 $\pm$ 1.87	89.71 $\pm$ 1.10	89.41 $\pm$ 1.18	86.28 $\pm$ 1.62	91.23 $\pm$ 2.14 (+16.6)
AIME 2026	Acc.	49.33 $\pm$ 1.49	47.33 $\pm$ 2.79	49.33 $\pm$ 2.79	58.67 $\pm$ 2.98	66.67 $\pm$ 2.36	70.67 $\pm$ 1.49	71.33 $\pm$ 1.83	51.33 $\pm$ 2.98	75.33 $\pm$ 1.83 (+26.0)
MedXpertQA	Acc.	65.78 $\pm$ 0.86	68.59 $\pm$ 1.69	75.78 $\pm$ 1.10	95.47 $\pm$ 1.50	96.09 $\pm$ 1.83	95.62 $\pm$ 0.89	90.00 $\pm$ 1.69	89.84 $\pm$ 2.21	97.97 $\pm$ 0.70 (+32.2)
ALFWorld	SR	33.59 $\pm$ 1.56	40.47 $\pm$ 2.10	43.44 $\pm$ 1.62	79.69 $\pm$ 1.46	93.59 $\pm$ 2.02	95.47 $\pm$ 1.87	92.81 $\pm$ 1.40	87.34 $\pm$ 1.02	98.44 $\pm$ 1.99 (+64.8)
MBPP+	Pass@1	74.53 $\pm$ 1.80	75.47 $\pm$ 2.57	76.56 $\pm$ 1.75	77.66 $\pm$ 1.52	78.12 $\pm$ 1.35	77.34 $\pm$ 1.91	81.41 $\pm$ 1.40	75.78 $\pm$ 1.24	84.84 $\pm$ 1.80 (+10.3)
Avg. (IID)	EM	57.97 $\pm$ 0.77	58.44 $\pm$ 0.94	59.69 $\pm$ 1.27	70.55 $\pm$ 1.20	70.31 $\pm$ 1.13	73.52 $\pm$ 1.22	73.67 $\pm$ 0.86	70.62 $\pm$ 1.03	75.94 $\pm$ 1.27 (+18.0)
	F1	70.13 $\pm$ 1.15	72.18 $\pm$ 1.28	72.21 $\pm$ 1.27	79.50 $\pm$ 0.78	79.97 $\pm$ 1.44	81.33 $\pm$ 0.76	83.89 $\pm$ 1.14	81.00 $\pm$ 1.27	85.47 $\pm$ 1.45 (+15.3)
	Acc.	55.81 $\pm$ 0.73	57.97 $\pm$ 1.16	61.28 $\pm$ 0.96	77.87 $\pm$ 0.99	83.62 $\pm$ 0.96	84.78 $\pm$ 0.80	83.89 $\pm$ 0.79	76.08 $\pm$ 1.01	89.15 $\pm$ 0.83 (+33.3)
(b) Out-of-Distribution (OOD) benchmarks										
MuSiQue	EM	38.59 $\pm$ 2.57	38.75 $\pm$ 1.80	39.38 $\pm$ 1.05	41.41 $\pm$ 1.46	41.72 $\pm$ 1.52	44.53 $\pm$ 1.10	43.59 $\pm$ 1.40	39.84 $\pm$ 0.78	46.41 $\pm$ 1.88 (+7.8)
	F1	38.69 $\pm$ 1.10	39.49 $\pm$ 1.40	39.08 $\pm$ 1.42	44.41 $\pm$ 0.86	43.84 $\pm$ 1.62	53.59 $\pm$ 2.02	46.23 $\pm$ 1.96	43.69 $\pm$ 1.66	59.93 $\pm$ 2.21 (+21.2)
NQ-Open	EM	27.97 $\pm$ 1.87	30.78 $\pm$ 1.71	28.44 $\pm$ 2.04	75.78 $\pm$ 1.66	78.44 $\pm$ 0.89	88.28 $\pm$ 2.14	84.53 $\pm$ 1.60	81.72 $\pm$ 1.96	93.44 $\pm$ 0.70 (+65.5)
	F1	49.41 $\pm$ 1.91	54.40 $\pm$ 1.05	48.49 $\pm$ 1.71	82.39 $\pm$ 1.40	84.85 $\pm$ 1.71	91.62 $\pm$ 1.95	88.27 $\pm$ 1.88	85.38 $\pm$ 1.62	93.73 $\pm$ 1.16 (+44.3)
MATH-Hard	Acc.	77.81 $\pm$ 1.71	77.50 $\pm$ 1.02	71.72 $\pm$ 1.28	85.78 $\pm$ 0.86	93.44 $\pm$ 1.18	87.34 $\pm$ 1.40	85.31 $\pm$ 1.78	78.75 $\pm$ 0.86	94.69 $\pm$ 1.95 (+16.9)
GPQA-D	Acc.	66.25 $\pm$ 1.28	70.31 $\pm$ 2.21	69.53 $\pm$ 1.46	74.69 $\pm$ 2.04	77.03 $\pm$ 1.31	73.75 $\pm$ 1.05	74.06 $\pm$ 1.95	73.28 $\pm$ 1.60	79.84 $\pm$ 1.69 (+13.6)
HumanEval	Pass@1	82.34 $\pm$ 1.88	80.00 $\pm$ 1.62	82.81 $\pm$ 0.55	86.41 $\pm$ 1.18	84.22 $\pm$ 1.40	83.91 $\pm$ 1.42	83.75 $\pm$ 2.02	82.03 $\pm$ 1.83	86.41 $\pm$ 1.80 (+4.1)
WebShop	SR	44.69 $\pm$ 1.40	46.88 $\pm$ 0.55	48.12 $\pm$ 1.52	65.94 $\pm$ 1.31	80.62 $\pm$ 1.78	78.28 $\pm$ 1.69	81.88 $\pm$ 1.28	76.41 $\pm$ 1.28	84.69 $\pm$ 1.52 (+40.0)
Avg. (OOD)	EM	33.28 $\pm$ 1.59	34.77 $\pm$ 1.24	33.91 $\pm$ 1.15	58.59 $\pm$ 1.10	60.08 $\pm$ 0.88	66.41 $\pm$ 1.20	64.06 $\pm$ 1.06	60.78 $\pm$ 1.06	69.92 $\pm$ 1.00 (+36.6)
	F1	44.05 $\pm$ 1.10	46.95 $\pm$ 0.87	43.79 $\pm$ 1.11	63.40 $\pm$ 0.82	64.35 $\pm$ 1.18	72.61 $\pm$ 1.40	67.25 $\pm$ 1.36	64.53 $\pm$ 1.16	76.83 $\pm$ 1.25 (+32.8)
	Acc.	67.77 $\pm$ 0.79	68.67 $\pm$ 0.74	68.05 $\pm$ 0.63	78.20 $\pm$ 0.71	83.83 $\pm$ 0.72	80.82 $\pm$ 0.70	81.25 $\pm$ 0.89	77.62 $\pm$ 0.72	86.41 $\pm$ 0.87 (+18.6)

Table 1: Main results on twelve IID and OOD benchmarks. SFT and GRPO are applied on Qwen3.5-9B; all trainable orchestration methods use Qwen3.5-9B as the frozen executor under the same per-task budget. $\Delta \uparrow$ is the gap to direct Qwen3.5-9B. Every cell is the mean over five runs; the subscript is the standard deviation across those five, propagated across benchmarks for the Avg. rows. QA averages are taken separately for EM and F1, and Acc. averages the primary metrics (accuracy, success rate, or Pass@1) of the other four benchmarks in each part.

2019; Rein et al., 2023; Chen et al., 2021). Baselines are direct Qwen3.5-9B, SFT and GRPO (Shao et al., 2024) on it, AFlow (Zhang et al., 2025), FlowSteer (Zhang et al., 2026a), the evolving orchestrator (Dang et al., 2026), SkillFlow (Zhang et al., 2026b) and SkillIRL (Xia et al., 2026). Qwen3.5-9B serves as both the Collab-Director and the frozen executor, and all methods share one Agent population, tool set and per-task budget, which also charges messages. We report exact match (EM, primary) and F1 for QA and accuracy, success rate or Pass@1 elsewhere; Table 1 and Figure 4b use five runs with Welch’s t -test, and Figure 4a,d uses six other executors (Appendices D–G).

In distribution (RQ1). CollabFlow leads every IID dataset and metric (Table 1), 2.27 EM and 4.37 accuracy points above the strongest baseline on average, and every average gain is significant at $p < 10^{-2}$ (Figure 4b). SFT and GRPO add at most 5.5 accuracy points to the backbone, whereas learned teams add 33.3: as Eq. 6 states, with the executor and Agents frozen, every gain comes from the teams the director builds and how their members exchange evidence.

Out of distribution (RQ2). With everything frozen, CollabFlow again leads every OOD row but HumanEval, where it ties AFlow, and its margins widen to 3.51 EM and 4.22 F1, while fine-tuning moves the backbone by under one accuracy point. The director learns which roles to combine and how messages are used rather than task answers, and held-out tasks also yield checkable evidence.

Across executors (RQ3). CollabFlow raises every IID and OOD average of six other frozen executors (Figure 4a,d; Table H.1), and weaker executors gain more (Pearson $r = -0.96$ in distribution), narrowing the IID spread across executors from 23.3 to 9.3 points: gated teams of complete Agents supply the checks that a weak executor lacks alone.

Variant	IID						OOD					
	HotpotQA	TriviaQA	AIME	MedXpertQA	ALFWorld	MBPP+	MuSiQue	NQ-Open	MATH	GPQA	HumanEval	WebShop
	EM	EM	Acc.	Acc.	SR	Pass@1	EM	EM	Acc.	Acc.	Pass@1	SR
Single agent with tools	47.66	78.12	56.67	82.03	85.16	66.41	25.78	72.66	78.91	56.25	70.31	32.81
Parallel vote	58.59	85.94	73.33	96.88	89.06	82.03	42.19	89.84	92.97	75.78	82.81	50.78
Fixed chain	53.91	81.25	73.33	84.38	96.09	83.59	39.06	82.03	94.53	68.75	84.38	53.91
Fully connected debate	52.34	86.72	70.00	94.53	92.19	80.47	43.75	87.50	93.75	78.91	81.25	44.53
Evolving orchestrator	57.03	85.16	63.33	91.41	95.31	78.91	35.94	84.38	89.84	73.44	78.91	60.16
— Director training	50.78	78.91	53.33	89.84	82.81	71.88	28.91	66.41	73.44	64.84	75.78	58.59
— Recursion	57.81	83.59	60.00	94.53	89.06	78.91	32.03	76.56	79.69	69.53	80.47	65.62
Fixed team	61.72	87.50	70.00	97.66	93.75	82.81	44.53	88.28	92.97	76.56	84.38	78.12
Random legal graph	58.59	84.38	63.33	93.75	90.62	79.69	38.28	82.03	88.28	71.88	82.81	72.66
— Communication	46.88	75.78	50.00	88.28	78.91	68.75	21.09	63.28	70.31	60.94	78.91	55.47
— Evidence gate	57.03	86.72	63.33	96.09	92.19	81.25	30.47	72.66	78.12	67.97	81.25	68.75
All ONEWAY	55.47	82.81	60.00	94.53	87.50	77.34	27.34	75.00	75.78	70.31	80.47	62.50
Always revise	53.91	85.16	56.67	92.97	85.94	75.78	24.22	68.75	86.72	65.62	79.69	67.19
CollabFlow (full)	63.28	88.28	76.67	98.44	97.66	85.16	46.88	92.97	95.31	79.69	85.94	84.38

Table 2: Paradigm comparison and ablation of CollabFlow on the six IID and six OOD benchmarks (Qwen3.5-9B executor; EM for QA; one run). Paradigm rows fix how Agents collaborate before execution: one tool-using Agent, parallel voting, a fixed chain, fully connected debate, and the evolving orchestrator. Team-level rows remove director training or recursion across rounds, or replace the learned team with a fixed team or a random legal graph; message-level rows remove communication or the evidence gate, force every edge to ONEWAY, or revise on every update.

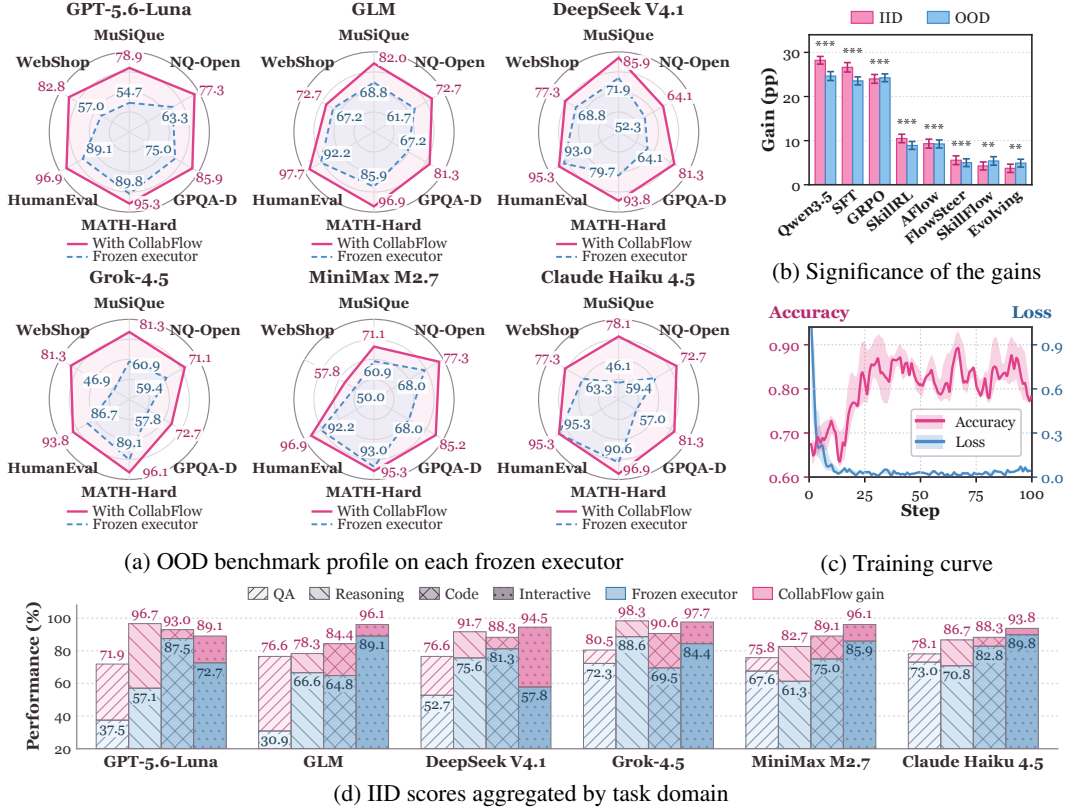


Figure 4: Other frozen executors, significance of the gains, and training dynamics. (a) OOD scores of each frozen executor used directly (dashed) and with CollabFlow (solid); each axis is linear over its own score range (values in Table H.1). (b) Gain of CollabFlow over each baseline in the main average, five-run mean with standard-error bars; Welch t -test: *** $p < 10^{-3}$, ** $p < 10^{-2}$. (c) Training accuracy and CTB loss; bands give the spread over runs. (d) IID scores averaged per task domain (QA: HotpotQA, TriviaQA; Reasoning: AIME 2026, MedXpertQA; Code: MBPP+; Interactive: ALFWorld); blue bars give the frozen executor alone and pink bars the gain from CollabFlow.

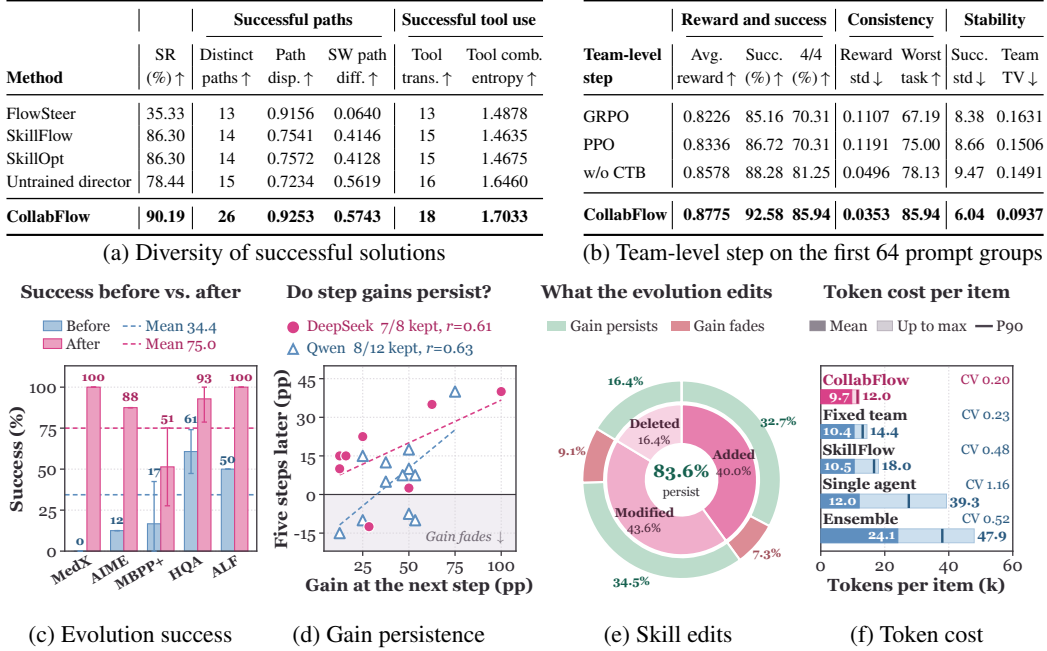


Figure 5: Solution diversity, team-level training, skill evolution and cost. **(a)** Successful solutions on 1,438 items, with SkillOpt (Yang et al., 2026b); SW: success-weighted. **(b)** First 64 prompt groups of training (8 steps, 256 rollouts per method); TV between consecutive team distributions. **(c)** Success over 8 rollouts before and after the twenty evolution steps with the largest one-step gain; dashed: means. **(d)** Gain at the next step against five steps later. **(e)** The 55 skill edits by type (inner) and by whether their step still gains five steps later (outer). **(f)** Tokens per item.

Paradigms and components (RQ4). Every paradigm and ablated variant falls below the full model on all twelve datasets (Table 2), each losing more out of distribution; a single tool-using Agent is lowest, and fully connected debate trails parallel voting. Removing director training costs 13.7 IID and 19.5 OOD points, of which recursion across rounds supplies 7.6 and 13.6. Removing communication costs most, while verbatim exchange, forced revision and all-ONEWAY edges cost 5.5–9.8 IID but 14.3–15.6 OOD points: by Proposition 2, a correct receiver flips only by false adoption or revision error, and each variant widens one route, most under shift, where more candidates err.

Team-level objective (RQ5). Under matched rollouts and reward, CTB beats GRPO, PPO and CollabFlow without CTB on every team-level measure, with the smallest team-distribution shift (Figure 5b). Its reward-proportional target (Eq. 12) keeps every positive-reward team in play and Proposition 3 bounds how far that target moves, so CollabFlow finds 26 distinct successful paths against at most 15 (Figure 5a), and training accuracy passes 0.80 within 25 steps (Figure 4c).

Recursive self-improvement (RQ6). Between evolution steps, the loop edits the skills the director binds to Agents (Appendix A.2). Over the twenty steps with the largest one-step gain, success on the evolved task rises from 34.4% to 75.0%, 15 steps stay ahead five steps later, and they carry 83.6% of the 55 edits (Figure 5c–e), as the reward of Eq. 13 counts each outcome once and smooths it with the team’s record. It also uses the fewest tokens per item with the lowest spread (Figure 5f; Table H.2).

6 CONCLUSION

CollabFlow makes agent collaboration recursively self-improving: evidence-conditioned protocols let Agents refine one another’s answers within a round, and Collaborative Trajectory Balance retrains the Collab-Director on its own records across rounds while keeping several good teams in play, and the target it generates moves between rounds by a bound that shrinks as records accumulate. On twelve datasets it leads every baseline in and out of distribution, beats every fixed paradigm and ablated variant, lifts six other frozen executors, keeps most of its step gains, and uses the fewest tokens per item (limitations and future work in Appendices I and J).

AI USE STATEMENT

In this work, generative AI tools were used to edit human-written text for clarity and readability, assist with L^AT_EX typesetting and page layout, and adjust the placement and sizing of existing figures and tables. They were not used to generate or modify experimental data or scientific-figure content, generate synthetic data, formulate the research hypotheses or mathematical claims, develop the theoretical or conceptual framework, design the methodology or experiments, implement the proposed method, clean or reformat datasets, perform qualitative data analysis, or interpret the experimental results. The authors manually reviewed and verified all AI-assisted text, figures, numbers, citations, and formatting against the implementation and experimental records. The authors take full responsibility for the final content of the paper and all AI-assisted artifacts.

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 every reported number is produced by the released code. We appropriately acknowledge prior work and use existing public benchmarks, models, in accordance with their intended purposes and licenses. CollabFlow’s Agents act only through permitted tools, a single Executor holds write permission on stateful tasks, and results on medical questions do not authorize clinical use (Appendix K). This study does not involve human participants or the collection of personal information.

REPRODUCIBILITY STATEMENT

We release the code at <https://anonymous.4open.science/r/CollabFlow-631E>, including the training and evaluation code, the trained Collab-Director weights, and the training and test splits of all twelve datasets. Section 4 specifies the team construction, the evidence-gated update and the CTB objective, and Section 5 specifies the benchmarks, baselines, shared budget and five-run protocol. The appendix states the assumptions and proofs of the three propositions (Appendix B), the Collab-Director interface and Agent population (Appendix A), the training procedure (Appendix C), the datasets and evaluation protocol (Appendix D), baselines and ablations (Appendix E), metrics (Appendix F), and per-benchmark results with other executors and token costs (Appendix H).

REFERENCES

- Summer Eunhyung Ann, Haokun Liu, and Chenhao Tan. The interaction tax: When communication erases diversity in multi-agent teams. *arXiv preprint arXiv:2608.23541*, 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.
- Blaž Bertalaníć and Carolina Fortuna. The cost of consensus: Isolated self-correction prevails over unguided homogeneous multi-agent debate. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 311–329, 2026.
- 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.
- Mingguang Chen, Licheng Wang, and Bo Qu. Recursive self-improvement in ai: From bounded self-refinement to autonomous research loops. *arXiv preprint arXiv:2607.07663*, 2026a.
- Xudong Chen, Yixin Liu, Hua Wei, and Kaize Ding. Lemon: Learning executable multi-agent orchestration via counterfactual reinforcement learning. *arXiv preprint arXiv:2605.14483*, 2026b.

- Yong Jin Chun and Iftekhar Ahmed. What do agents communicate? characterizing information exchange in multi-agent systems. *arXiv preprint arXiv:2605.20548*, 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. *Advances in neural information processing systems*, 38:165025–165059, 2026.
- Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical association*, 69(345): 118–121, 1974.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.
- Wanshuang Gou and Zihan Liu. Dynamic trust-aware sparse communication topology for llm-based multi-agent consensus. *arXiv preprint arXiv:2606.01828*, 2026.
- Yao Guan, Lin Wang, Zhihu Lu, Ziyi Wang, Wenzhu Yan, and Qiang Duan. Moc: Multi-order communication in llm-based multi-agent systems. *arXiv preprint arXiv:2606.02359*, 2026.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.
- Mao-xun Huang, Jerry Wang, Yi-Cheng Lai, Zhengxin Zhang, Claire Cardie, and Hen-Hsen Huang. Manta: Multi-agent network topology adaptation for self-evolving multi-agent systems. *arXiv preprint arXiv:2607.28527*, 2026.
- Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and physical sciences*, pp. 453–461, 1946.
- Yangxiao Jiang, Jiarun Fan, Mingcong Xu, Yanxi Guo, Jiwen Feng, Shanqing Xu, Mengchen Qian, Wei Chen, and Xiaojin Zhang. When evidence shapes collaboration: Knowledge-conditioned topology generation for multi-agent systems. *arXiv preprint arXiv:2608.27984*, 2026.
- 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.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for” mind” exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.
- Xiaozhe Li, Yang Li, Xinyu Fang, Shengyuan Ding, Peiji Li, Yongkang Chen, Yichuan Ma, Tianyi Lyu, Linyang Li, Dahua Lin, et al. Beyond mode collapse: Distribution matching for diverse reasoning. *arXiv preprint arXiv:2605.19461*, 2026.
- Yingzhuo Liu. Beyond tokens: a unified framework for latent communication in llm-based multi-agent systems. *arXiv preprint arXiv:2606.05711*, 2026.
- Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration. *arXiv preprint arXiv:2310.02170*, 2023.
- 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.

- Quim Motger, Marc Oriol, Jordi Marco, and Xavier Franch. Multi-agent debate strategies: Survey, taxonomy, and challenges. *arXiv preprint arXiv:2607.26212*, 2026.
- Christos H Papadimitriou. The serializability of concurrent database updates. *Journal of the ACM (JACM)*, 26(4):631–653, 1979.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.
- Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. Pmlr, 2015.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 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.
- Max W Shen, Emmanuel Bengio, Ehsan Hajiramezani, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International conference on machine learning*, pp. 30956–30975. PMLR, 2023.
- 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.
- Nurbek Tastan, Alex Iacob, Lorenzo Sani, Meghdad Kurmanji, Nicholas D Lane, Samuel Horvath, and Karthik Nandakumar. Response-conditioned parallel-to-sequential orchestration for multi-agent systems. *arXiv preprint arXiv:2605.15573*, 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.
- Siyu Wang, Ruotian Lu, Zhihao Yang, Yuchao Wang, Yanzhou Zhang, Lei Xu, Qimin Xu, Guojun Yin, Cailian Chen, and Xinping Guan. Agentconductor: Topology evolution for multi-agent competition-level code generation. *arXiv preprint arXiv:2602.17100*, 2026.
- Haoning Wu, Xiao Huang, Yaohui Chen, Ya Zhang, Yanfeng Wang, and Weidi Xie. Spatialscore: Towards comprehensive evaluation for spatial intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31029–31041, 2026.
- 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 conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- 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.
- Xin Yang, Letian Li, Zimo Ji, Terry Jingchen Zhang, and Wenyuan Jiang. Position: Multi-agent systems should prioritize concurrency control. *arXiv preprint arXiv:2608.18092*, 2026a.
- Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 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.

Jinxi Yu, Yubei Li, Eric Hanchen Jiang, Zhi Zhang, Dong Liu, Wenxiao Zhao, Levina Li, Kai-Wei Chang, and Ying Nian Wu. Codebook agent: Amortized topology design for llm multi-agent systems. *arXiv preprint arXiv:2609.02264*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.

Mingda Zhang, Wenjin Liu, Tiesunlong Shen, Qika Lin, Rui Mao, Erik Cambria, Xiaoying Tang, and Haoran Luo. Flowsteer: Towards agents designing agentic workflows via reinforced progressive canvas editing. *arXiv preprint arXiv:2602.01664*, 2026a.

Mingda Zhang, Tiesunlong Shen, Haoran Luo, Wenjin Liu, Zikai Xiao, Erik Cambria, and Xiaoying Tang. Skillflow: Flow-driven recursive skill evolution for agentic orchestration. *arXiv preprint arXiv:2605.14089*, 2026b.

Jiaru Zou, Rui Pan, Ruizhong Qiu, Pan Lu, Shizhe Diao, Jindong Jiang, Hanghang Tong, Tong Zhang, Markus J Buehler, Jingrui He, et al. Recursive multi-agent systems. *arXiv preprint arXiv:2604.25917*, 2026.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. Medxpertqa: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*, 2025.

A COLLAB-DIRECTOR PROMPTS, AGENT POPULATION, AND RUNTIME INTERFACES

This appendix supplements the collaboration graph, the communication semantics, and the training procedure in the main text. Appendix B states the assumptions of the three propositions and proves them; the remaining sections specify the algorithms and evaluation protocol.

A.1 COLLAB-DIRECTOR INPUT AND OUTPUT CONTRACT

The Collab-Director receives the task x , compatible Agent profiles $\mathcal{A}_x$, the partial graph G_t , the protocols permitted for the task family, and the remaining construction budget b_t^{con} . Its context H_t also contains the accepted construction decisions. Reference answers, hidden evaluation tests, and evaluator-only labels are not exposed as construction inputs. Feedback produced by a permitted tool during ordinary execution is distinct from hidden evaluator information.

The Collab-Director selects one structural action at a time. The following is a *schematic prompt contract*, not a verbatim prompt recovered from a training run:

Construct an Agent team for the supplied task using the available profiles. Select one currently legal construction action. Choose Agents, typed communication relations, and an output mechanism; leave task solving to the executor. Respect compatibility, write-permission, and resource constraints. Stop only when the team is valid. Do not invent Agent identifiers, and do not use evaluator-only information.

The frozen system prompt, user template, structured-action schema, and invalid-action correction prompt are released with the code.

A.2 AGENT PROFILES

An Agent is a complete execution role rather than an individual operator. Table A.1 lists the profile fields that the Collab-Director can read when constructing a team. Static profile capability tags

describe what an Agent may do; they are not modified during training. Agent profile definitions and executor parameters remain fixed; the Collab-Director changes which profiles are selected, which skills they carry, and how their instances communicate.

Table A.1: Conceptual schemas used by the method. Exact registry identifiers and serialization are configuration-dependent.

Object / field	Meaning
Agent identity and role	Stable profile identifier, role description, and local objective.
Compatibility	Supported task families and capability labels.
Tool permissions	Permitted tools and whether an operation can modify protected environment state.
Visibility	Task evidence, messages, and observations visible to the Agent.
Execution mode	Stateless Agent, mutable EXECUTOR, or read-only ADVISOR.

The construction interface also supports a `bind_skill` action that attaches a retrieved textual strategy to one selected Agent; it grants no tool permissions and never overrides write permission. Between evolution steps, the loop adds, modifies or deletes library skills from the outcomes of the teams that used them; Figure 5c–e reports these edits.

A.3 STRUCTURAL ACTIONS AND VALIDATION

Table A.2 expands the action space of Section 4.1. Its arguments are mathematical objects. The legal finite action registry is exposed through a constrained schema. Schema legality and probability normalization are separate contracts: the director scores every legal action string and samples from the renormalized kernel of Eq. 5, whereas a grammar-masked token sampler would use its own normalized likelihood (Appendix B.3). The runtime checks each sampled action before changing the graph, so every sampled action is accepted; scoring tokens, latency, and model calls count toward the construction budget, and a timeout follows the declared abort rule.

Table A.2: Structural-action semantics and local validity conditions. Terminal constraints are checked separately from partial-state invariants.

Action	Structural effect	Required conditions
<code>add_agent(a)</code>	Add a to V .	$a \in \mathcal{A}_x \setminus V$; profile, permission, capacity, and budget checks pass.
<code>add_edge(a_i, a_j, p)</code>	Add (a_i, a_j, p) to E .	Both endpoints have been selected; $p \in \mathcal{P}$ is supported for the task and endpoint modes; the edge is not already present; relation and budget checks pass.
<code>bind_skill(a, k)</code>	Attach library skill k to Agent a .	$a \in V$; the skill grants no tool permission and never overrides write permission.
<code>set_output(o')</code>	Set output configuration to o' .	The output is not yet set; the configuration is supported, all referenced Agents exist, and its execution is compatible with task permissions and budget.
<code>stop</code>	Commit the terminal organization.	The team is nonempty, output is defined, and all task-specific terminal constraints are satisfied.

Partial-state validity permits an unfinished graph. For example, the output configuration can be unset before termination, and a stateful partial team may have *at most* one mutable Executor, whereas a completed stateful team requires *exactly* one. Enforcing every terminal condition on the empty initial graph would prevent construction. Candidate resources are held fixed within a construction episode; a changed population defines a different resource context.

A.4 COMMUNICATION AND OUTPUT SEMANTICS

The execution model first produces stateless candidates and then processes communication under the fixed schedule of Appendix B.6. The resulting revision trace records actual messages and responses rather than inferring communication from graph edges alone. For organization-class statistics, a graph with multiple Agents but no enabled intermediate communication is treated as independent, even when final results are routed to an output component. Report both planned edges and messages actually executed.

Within ONEWAY and INTERACTIVE, Revise is a single model call with tools disabled and at most τ_{rev} tokens, so an one visit to an edge program costs at most r_{max} revision calls of τ_{rev} tokens. A

Table A.3: Communication semantics. Bounds and task-specific availability are set by the frozen runtime configuration.

Protocol	Execution meaning
FINALONLY	Make a completed result available without rerunning the receiver.
ONWAY	Send evidence or a proposal to the receiver, which may revise once under the configured resource bound.
INTERACTIVE	Allow a bounded sequence of mutual correction rounds where the task and runtime support it.
Stateful restriction	Internal Advisors provide bounded one-way guidance to the sole mutable Executor, and write permission stays with the Executor.

cyclic schedule may visit the program up to R_{comm} times; all such calls are charged to its execution quota. A round denotes one elementary directed gate evaluation. Revision never re-executes the receiver’s tool episode. Each evidence item is bound to a candidate version and carries a provenance identifier. The context supplied to REVISE may contain the union of peer evidence, but after a revision only evidence that remains valid for the returned candidate is kept and duplicate identifiers are removed, so a revised candidate earns its own verification score. Under the stateful restriction an Advisor’s message is a structured handoff (state, evidence, next subgoal, completion check, recovery) produced without tools, and edges are restricted to ADVISOR→EXECUTOR ONWAY.

A SINGLE output selects one Agent result. An Integrator deduplicates records, filters them for its chosen candidate representative, and applies the declared score rule of Section 4.2; a Verifier evaluates candidates under its configured contract, and a Formatter changes submission format without resolving the task again. SINGLE and Formatter are counted as single-output modes, and the configuration declares which output modes are available.

A.5 EXECUTION RECORDS AND INFORMATION BOUNDARIES

An episode records the construction trajectory ζ , canonical graph G_ζ , execution trace ξ , output $\hat{y}$, factual score r_{ans} , and resource ledger. Construction decisions are the organization-policy learning targets. Child-Agent messages, tool results, and environment observations are execution evidence, separate from the Collab-Director’s action tokens. The evaluator receives the information required for factual scoring, while inference-time Agents receive only task-permitted inputs.

For reproducibility, records should include the task/split identifier, resource versions, accepted and rejected proposals, communication protocol and round, sender/receiver identifiers, before/after revisions, tool outcomes, stopping reason, and measured costs. The exact logging keys are given in the released code. This separation also prevents a post-execution reference answer from being reintroduced as a Collab-Director input.

B THEORETICAL FOUNDATIONS AND EXECUTION CONTRACTS

This appendix proves the results used in the main text: the invariants of checked construction, the identities of evidence-conditioned communication, trajectory-balance consistency for canonical teams, the bounded movement of the self-generated target between rounds, and the path diagnostics. Trajectory balance (TB) and reward-proportional terminal sampling build on GFlowNet results (Bengio et al., 2021; Malkin et al., 2022; Bengio et al., 2023); we carry them to complete-Agent teams with checked construction and terminal interfaces and explicit communication and execution contracts.

B.1 OBJECTS, CONDITIONING, AND THE TWO SOURCES OF RANDOMNESS

Fixed resource context. Fix a task x and a resource context χ . The context specifies the finite compatible Agent population $\mathcal{A}_x$; immutable Agent identifiers, profiles, tool versions, and permissions; the protocol set $\mathcal{P} = \{\text{FINALONLY}, \text{ONWAY}, \text{INTERACTIVE}\}$; the permitted output configurations; the validator and action encoding; message scheduling and tie-breaking rules; and the executor and initial environment law. It also specifies a construction decision cap $T_{\text{max}} \geq 1$, bounded action encodings, and separate resource vectors $\mathbf{B}^{\text{con}}$ and $\mathbf{B}^{\text{exec}}$. Inequalities between resource vectors are componentwise. A reward snapshot k fixes the reliability counters, reward evaluation rule,

structural reference code, and temperature $\beta > 0$. We write $c = (x, \chi, k)$ when all three are needed. Conditioning on c is suppressed within an argument in which it is fixed.

The two budgets are separate reservations: $\mathbf{B}^{\text{con}} + \mathbf{B}^{\text{exec}} \leq \mathbf{B}^{\text{total}}$. Construction never borrows from the execution reservation. Consequently, identical terminal graphs are evaluated under the same execution budget, regardless of their construction order. Changing an executor, permission, budget, scheduler, or reward snapshot defines a new conditioned problem.

Definition B.1 (Team and canonical identity). A team is an annotated directed graph $G = (V, E, o)$, where $\emptyset \neq V \subseteq \mathcal{A}_x$, $E \subseteq V \times V \times \mathcal{P}$, and o is a permitted output configuration. Agent identifiers identify complete, fixed profiles rather than interchangeable, unlabeled vertices. A canonicalizer Can is idempotent and identifies exactly the representations that differ only in serialization of the same vertices, protocol-labeled edges, and output configuration. Different permissions, output rules, Agent implementations, and execution contracts remain distinct. Write $\mathcal{G}_c$ for the finite set of reachable, valid canonical teams at context c .

All discrete configuration domains in this finite model are either finite registries or finite strings over a finite alphabet with declared length caps. For finite-DAG results, construction resource debits are fixed functions of these records or have a finite discretization. These restrictions give finiteness. All results below use the four structural actions of the current CollabFlow interface.

Construction state and complete transition. The display state is $s_t = (G_t, \mathcal{A}_x, \mathbf{b}_t^{\text{con}})$. The proof state v_t additionally retains every variable needed by the validator, sampler, and transition rule, including the decision index t and the construction history H_t . A sufficient choice is $v_t = (s_t, H_t, t)$. A smaller quotient state is permissible when histories identified by that quotient induce the same legal actions, conditional transition probabilities, and successor quotient states.

A transition $e = (v, u, v')$ includes an accepted action and its successor; parallel transitions are distinguished by the action. The active actions are `add_agent`, `add_edge`, `set_output`, and `stop`. The director samples from the legal-action kernel of Appendix B.3, so every sampled action is accepted and advances the decision index. Timeouts, empty admissible supports, and exhaustion before valid completion lead to an explicit abort outcome $\dagger$.

A complete trajectory is the ordered sequence $\zeta = (v_0, e_1, v_1, \dots, e_T, z)$, with $1 \leq T \leq T_{\max}$; T includes the transition to the absorbing outcome z . Every accepted `stop` for graph G enters the *same* absorbing node g_G . The terminal node retains the canonical graph and its fixed context, shared by every incoming history. Nonterminal history states can remain distinct. Denote complete paths to g_G by Ω_G , and let $\mathbf{V} = \{z = g_G : G \in \mathcal{G}_c\}$ be valid completion. Each path in Ω_G is one construction order of G : a prerequisite-respecting order of its Agent and edge insertions and its single output selection, followed by `stop`.

Runtime execution and reward. For a valid graph, let $K_\chi(d\xi, dy \mid G, x; \mathbf{B}^{\text{exec}})$ be the normalized joint law of all runtime randomness ξ and the final answer y . The trace ξ contains model draws, tool outcomes, message events, and any permitted environmental randomness. A bounded failure or timeout returns $y = \perp$ and is included in this law. Freezing executor parameters fixes this kernel, whose draws remain stochastic. The construction history affects execution only through $(G, x, \chi, \mathbf{B}^{\text{exec}})$ under this contract.

Write $\tilde{R}_k(G, x, \xi) > 0$ for an observed execution reward and r_G for the *fixed* graph reward used in a particular ideal TB argument. A deterministic evaluator or an immutable cached reward table defines r_G , and a fresh evaluation supplies the noisy training signal $\tilde{R}_k$. Sections B.11 and B.12 make the reward construction and the role of noise precise.

Lemma B.2 (Two probability spaces). *Suppose the forward construction law P_θ is normalized and $p_V := P_\theta(\mathbf{V}) > 0$. Then*

$$\mu_\theta^\mathbf{V}(G \mid c) = \frac{\sum_{\zeta \in \Omega_G} P_\theta(\zeta \mid c)}{p_V}, \quad \sum_{G \in \mathcal{G}_c} \mu_\theta^\mathbf{V}(G \mid c) = 1. \quad (\text{B.1})$$

Conditional on valid completion, the answer law is $\sum_G \mu_\theta^\mathbf{V}(G \mid c) K_\chi(dy \mid G, x)$. Unconditionally, it additionally includes the probability $1 - p_V$ assigned to construction abort.

Proof. The events consisting of different complete paths are disjoint. By terminal identity, their union over Ω_G is precisely the event of returning G , and those events partition V . Division by p_V gives the first claim. The answer law follows by conditioning first on G and then integrating ξ under its normalized execution kernel. $\square$

Example. Terminal merging makes the target graph-level. For two unit-reward graphs with two and one construction paths, equal mass per path would give graph masses $2/3$ and $1/3$; the graph law of Eq. B.1 gives $1/2$ and $1/2$, the target of the Preliminaries.

B.2 CHECKED CONSTRUCTION: SOUNDNESS AND COVERAGE

Let $\mathcal{I}_c(v)$ be the partial invariant: selected profiles are compatible, edge endpoints exist, protocols and permissions are permitted, output references are well formed when present, and remaining construction resources are nonnegative. On stateful tasks it permits at most one internal writer. The stronger terminal predicate $\mathcal{V}_c(v)$ additionally requires a nonempty team, a permitted output, and all terminal constraints, including exactly one writer when the task requires one.

C1–C3. **C1 (checked preservation)** requires $\mathcal{I}_c(v_0) = 1$, $\mathcal{I}_c(v) = 1 \Rightarrow \mathcal{I}_c(\mathcal{T}_c(v, u)) = 1$ for every accepted nonterminal action, and acceptance of `stop` only when $\mathcal{V}_c(v) = 1$. **C2 (component completeness)** requires that every missing component of a target expressible by the active interface can be added once its prerequisites hold, with no extra mask excluding all such orders. **C3 (feasible prefixes)** requires, for each target covered by the claim, a prerequisite-respecting order of its component insertions, output selection, and stop whose every prefix fits the construction budget and cap; component identities and meanings remain fixed along this order. C3 is a resource-feasibility assumption on the targets.

Theorem B.3 (Soundness and relative coverage). *Under C1, every accepted completed trajectory yields a valid canonical team. Under C1–C3, every target $G^* = (V^*, E^*, o^*)$ in the stated feasible target class has a legal construction trajectory. With one insertion per vertex and edge, one output selection, and one stop, it has length*

$$T^* \leq |V^*| + |E^*| + 2. \quad (\text{B.2})$$

If the sampler gives every transition of this witness positive probability, that witness also has positive sampling probability.

Proof. For soundness, induct on accepted actions. C1 establishes the base invariant at v_0 . An accepted vertex insertion preserves compatibility and inherited permissions; an edge insertion is checked against existing endpoints and available protocols; output selection is checked against the selected team. More generally, preservation follows directly from C1 for each accepted transition. The final stop additionally checks $\mathcal{V}_c$. Canonical serialization preserves the validated annotations, so the decoded terminal is valid. A rejected attempt preserves structural invariants, and the terminal predicate comes from the accepted stop.

For coverage, take the prerequisite order supplied by C3. C2 makes the first required component available. Assuming the first j components have been inserted, the next component’s prerequisites hold by the chosen order; C2 makes its action available and C3 makes its cost feasible. C1 accepts the checked transition. Induction constructs all target components without changing their meanings. Output selection and stop complete the target. Counting these actions gives the bound. Multiplying finitely many positive transition probabilities proves the final statement. $\square$

Theorem B.3 proves Proposition 1; C1–C3 concern the fixed registry χ , so the witness exists in every round. Single-Agent and independent teams are covered whenever their output and permission configurations satisfy the same predicates.

Lemma B.4 (Finite construction DAG). *With bounded encodings, a finite context, a finite decision cap, and sufficient states retaining the decision index, the construction multigraph is finite and acyclic. Canonical absorbing terminals can merge histories without introducing a cycle. At an interior full-history state, if the appended record uniquely identifies the previous history and action, there is exactly one incoming parent–action pair.*

Proof. There are finitely many bounded records and at most $T_{\max}$ decisions, so finitely many histories and states. Each interior transition strictly increases the decision index. Assign all absorbing terminals

rank $T_{\max} + 1$; every edge increases rank, ruling out a directed cycle. An interior full-history state has a unique predecessor by removing its last record, and that record identifies the action. Canonical terminals merge several histories by design, because their definition forgets the final history. $\square$

At a unique-parent interior state, the backward kernel equals one.

B.3 ACTION LIKELIHOODS AND LEGAL-SUPPORT NORMALIZATION

Let $\mathcal{U}_c(v)$ be a nonempty finite legal action set at an interior state, and $a_\theta(u \mid v) \geq 0$ a base action mass. The action-level masked kernel is

$$\bar{\pi}_\theta(u \mid v, c) = \frac{\mathbf{1}\{u \in \mathcal{U}_c(v)\} a_\theta(u \mid v)}{\sum_{u' \in \mathcal{U}_c(v)} a_\theta(u' \mid v)}, \quad \sum_{u' \in \mathcal{U}_c(v)} a_\theta(u' \mid v) > 0. \quad (\text{B.3})$$

An empty or zero-mass support uses the declared abort transition. For deterministic construction, $p_F(v, u, v' \mid v, c) = \bar{\pi}_\theta(u \mid v, c)$ on the prescribed successor. When several serializations encode the same semantic action, $a_\theta(u \mid v)$ must sum their masses, or the encoding must be canonical and unique. CollabFlow scores every legal action string and samples from Eq. B.3, so no proposal is rejected; a propose-and-validate sampler with retries defines a different kernel, and its own likelihood would then enter the residual.

Lemma B.5 (Sequence probability of an action string). *For a completed, uniquely encoded action string $w_{1:K}$, including its termination symbol, an autoregressive sampler assigns probability $\prod_{j=1}^K p_\theta(w_j \mid w_{<j}, v)$ and log probability equal to the sum of those token log probabilities. This gives a normalized action kernel when the actual token sampler terminates almost surely in the admissible encoding set. The action log probability is therefore the sum of token log probabilities rather than their mean $K^{-1} \sum_j \log p_\theta(w_j \mid \cdot)$.*

Proof. The chain rule gives the product. Completed strings with their termination symbols are mutually exclusive, and almost-sure completion exhausts their sample space. Taking logarithms gives the sum. Exponentiating the token average instead gives $p_\theta(w_{1:K} \mid v)^{1/K}$. These transformed masses need not sum to one. For two length-two actions of probability $1/2$ each, their transformed masses sum to $\sqrt{2}$. An additional action-level normalization defines a different sampler whose normalizer must enter TB. $\square$

Grammar masking as an exactly scored sampler. Constraining tokens to legal continuations yields its own normalized kernel, which can differ from Eq. B.3 formed from original complete-string masses. For legal strings aa and ba , let first-token probabilities be $0.9, 0.1$, and let the original probability of second-token a be 0.1 after a and 0.9 after b . The complete legal strings both have mass 0.09 , so complete-action conditioning gives $1/2, 1/2$. Prefix masking forces the second token but leaves the first-token distribution $0.9, 0.1$. Either sampler is usable when its own temperature, truncation, token masks, stop rules, and normalization enter its likelihood.

Sampled reasoning and construction feedback. If a reflection r is sampled before an action, then the joint transition mass includes $p_\theta(r \mid v) \pi_\theta(u \mid r, v)$, or r must be marginalized when it is absent from the state. If feedback o is stochastic, the full transition also includes its kernel $K(o \mid v, r, u)$. Frozen parameters make the derivative of that kernel zero, and its probability stays in the transition mass. An action-only model is exact when all omitted variables are deterministic functions of the modeled state and action, or when the stated marginal kernel actually integrates them.

B.4 CANDIDATE VERSIONS, EVIDENCE APPLICABILITY, AND DUPLICATION

A candidate $a = (\iota, y, \nu)$ contains an immutable candidate-version identifier ι , its answer or executable artifact y , and the relevant environment version ν . An evidence record $e = (\text{id}, \iota_e, \nu_e, \text{check}, \text{result}, \text{scope}, \text{source})$ has an immutable event identifier. Copying a record copies its identifier, so a copy adds no new verification event. The runtime authenticates each record's source, so a message that merely claims a passed test adds no evidence.

Let $A_c(e, a) \in \{0, 1\}$ be the applicability predicate. It can be one only when the recorded check concerns this candidate and environment version, or a sound, explicitly certified transfer of that check.

Applicability means that the record still expresses the same check on the current artifact. Define

$$\text{Filter}_c(E, a) = \text{Dedup}\{e \in E : A_c(e, a) = 1\}, \quad \ell(a, E) = s_c(a, \text{Filter}_c(E, a)). \quad (\text{B.4})$$

Here Dedup retains one authentic record per event identifier, and s_c is the declared finite real-valued evidence score, with $s_c(a, \emptyset) = 0$. Missing or nonfinite scores trigger a declared fallback.

Lemma B.6 (Evidence-validity invariant). *Assume initial evidence is filtered, copying preserves the entire candidate version, evidence records are immutable, and every changed candidate is filtered by Eq. B.4. Then after any finite sequence of copy, keep, and tool-free revision operations, every retained record is applicable to its current candidate and appears at most once per evidence set. A new candidate with no applicable incoming record has empty inherited evidence and inherited score zero.*

Proof. The base case follows from initialization. Keeping a candidate preserves the invariant. Copying the complete version and its filtered set preserves applicability and event identifiers. A revision creates a new version; filtering deletes every nonapplicable record and deduplication deletes repetitions. These observations establish the inductive step. The last statement follows directly from the empty-set convention for s_c . Equivalently, filtering is idempotent, and filtering a union agrees with deduplicating the union of its individually filtered sets, because the same candidate-specific predicate is applied to every immutable record. $\square$

Example. A revised program may fail two tests that its two parent programs passed. The lemma keeps those pass records with the parents, so the revision earns its own checks.

Scores and likelihood-ratio interpretation. The gate compares scores. The method uses the maximum applicable evidence category: 2 for an applicable successful executable check, 1 for applicable retrieval support if there is no category-2 record, and 0 otherwise. Repeated copies leave that maximum unchanged. The flip identity below holds for this surrogate and for a calibrated log-likelihood ratio (LLR) alike.

For an LLR interpretation, let Z_a indicate candidate correctness and let $E_1, \dots, E_m \in \{0, 1\}$ be outputs of a fixed check schedule. Conditional on the candidate-selection context, assume the checks are independent given Z_a , with known rates $0 < t_j = P(E_j = 1 \mid Z_a = 1) < 1$ and $0 < f_j = P(E_j = 1 \mid Z_a = 0) < 1$. Then

$$\log \frac{P(E_{1:m} \mid Z_a = 1)}{P(E_{1:m} \mid Z_a = 0)} = \sum_{j=1}^m \left[E_j \log \frac{t_j}{f_j} + (1 - E_j) \log \frac{1 - t_j}{1 - f_j} \right]. \quad (\text{B.5})$$

The equality follows by conditional factorization and the logarithm of a product. Adaptive check selection, omission of failed tests, and candidate selection must either be included in the likelihood or satisfy corresponding conditional assumptions. The finite-rate form above keeps the LLR finite.

Aggregation contract. To implement an Integrator, first select a deterministic representative a_y for each normalized-answer class, using a declared tie rule. Pool records from that class, deduplicate them, and filter again for a_y before computing $s_c(a_y, \cdot)$. Select the maximum-scoring representative with a fixed tie rule; an empty candidate set returns $\perp$. The normalization relation preserves task correctness, and evidence transfer uses A_c . This rule is implementable and keeps a copied event from counting as a new verification event. A formatter or verifier that generates a different artifact creates a new version and is subject to the same applicability contract.

B.5 ONE-EDGE REVISION IDENTITY

Fix one scheduled directed gate invocation $i \rightarrow j$. Let a_i, a_j be its input candidates, $z_i, z_j \in \{0, 1\}$ their correctness indicators, and $\Delta = \ell(a_i, E_i) - \ell(a_j, E_j)$. Fix a finite margin $\kappa \geq 0$. The results below hold for every such margin. The method uses $\kappa = 1$: scores lie in $\{0, 1, 2\}$, so $\kappa = 0$ adopts on every positive gap, $\kappa \geq 2$ never adopts, and $\kappa = 1$ is the only integer margin that permits adoption but not on a one-level gap. The normalized-answer equivalence $\equiv$ is assumed to preserve correctness.

To avoid overlapping branches, apply the following priority:

$$a_j^+ = \begin{cases} a_j, & y_i \equiv y_j, \\ a_i, & y_i \not\equiv y_j \text{ and } \Delta > \kappa, \\ a_j, & y_i \not\equiv y_j \text{ and } \Delta < -\kappa, \\ \text{Revise}_c(a_j, a_i, E_i \cup E_j), & \text{otherwise.} \end{cases} \quad (\text{B.6})$$

Copies retain their candidate identity and evidence. The final branch creates a version and applies Eq. B.4. Revision randomness and a declared tool-free failure fallback are part of Revise_c ; $\perp$ counts as incorrect. Self-reported confidence enters neither s_c nor the threshold comparison; the revision kernel may read it among its permitted textual inputs.

Theorem B.7 (Single-edge, conditionally selected flip identity). *Assume the preceding gate contract and $P(C \mid c) > 0$, where $C = \{z_j = 1, z_i = 0\}$. Put $J = \{|\Delta| \leq \kappa\}$ and define*

$$\delta_C = P(\Delta > \kappa \mid C, c), \quad c_J = P(z_j^+ = 0 \mid J, C, c).$$

When $P(J \mid C, c) = 0$, set $c_J = 0$ by convention. Then

$$P(z_j^+ = 0 \mid C, c) = \delta_C + P(J \mid C, c)c_J. \quad (\text{B.7})$$

If $c_J \leq \bar{c}$ is a justified upper bound, the right-hand side is at most $\delta_C + P(J \mid C, c)\bar{c}$.

Proof. On C , correctness-preserving equivalence excludes $y_i \equiv y_j$. Thus the three disjoint events $\Delta > \kappa$, $\Delta < -\kappa$, and J partition C . In the first branch the incorrect peer is copied and the conditional flip probability is one. In the second branch the correct receiver is retained and it is zero. In the third branch it is c_J by definition. Applying the law of total probability *conditional on C, c* proves the identity, including null branches. Substitution of a valid upper bound proves the inequality. $\square$

Conditioning of the identity. This proves the flip part of Proposition 2 with the same c_J , δ_C , and conditioning context c . It requires no independence between the two candidates. The quantities are evaluated under the actual pair-selection and evidence-generation law, and the same proof conditions on a pre-invocation selection history whenever the stated conditional event has positive probability.

Corollary B.8 (Correction rate and the margin). *Under the same gate contract, let $C' = \{z_j = 0, z_i = 1\}$ with $P(C' \mid c) > 0$, $\delta^+ = P(\Delta > \kappa \mid C', c)$, and $c_J^+ = P(z_j^+ = 1 \mid J, C', c)$. Then*

$$P(z_j^+ = 1 \mid C', c) = \delta^+ + P(J \mid C', c)c_J^+. \quad (\text{B.8})$$

As κ increases, δ_C and δ^+ are nonincreasing, while $P(J \mid C, c)$ and $P(J \mid C', c)$ are nondecreasing.

Proof. On C' the answers differ, so the three branches again partition the event. Adoption copies the correct sender, keeping retains the incorrect receiver, and revision is correct with probability c_J^+ ; the law of total probability gives the identity. The events $\{\Delta > \kappa\}$ shrink and the events $\{|\Delta| \leq \kappa\}$ grow as κ increases, which proves monotonicity. $\square$

The margin therefore trades false adoption for revision on C and true adoption for revision on C' , and the two identities split the net effect of a gate into four rates that revision logs record.

Example B.9 (Event deduplication in voting). After a_i is copied into a_j , their outputs and evidence are shared. With weights 3 on a correct answer and 2 on a wrong answer, a third candidate that copies the wrong answer would turn the vote into 3 versus 2 + 2 if copies were counted separately. Deduplication by event identity keeps the vote at 3 versus 2.

B.6 RUNTIME SCHEDULING, FINITE EXECUTION, AND A LINEAR BASELINE

Construction edges add team components; runtime edges exchange messages between selected Agents. These are different graphs. Fix a total order on Agent identifiers and on protocol-labeled edges. For an acyclic runtime E , choose the lexicographically resolved topological order and visit each outgoing edge program once, with a fixed order among edges having the same source. For a cyclic runtime E , visit edge programs in fixed order for at most $R_{\text{comm}} < \infty$ sweeps. Budget exhaustion stops either schedule with the declared current-output or failure rule. The schedule is asynchronous and serial: every elementary update sees the state left by previous updates, rather than an unspecified mixture of snapshots.

FINALONLY performs no revision invocation. ONEWAY evaluates the directed gate once. For definiteness, one INTERACTIVE round is one elementary gate evaluation; its directions alternate, starting with the edge’s declared source-to-receiver direction. Agreement stops that local program before revision; a decisive copy or keep is applied and then stops it; otherwise it continues for at most $r_{\max} < \infty$ elementary rounds. Reciprocal updates are internal to this bounded program. A single topological pass visits each *edge program* once, and an interactive program replies locally within that visit.

Stateful Advisor messages are tool-disabled and feed the Executor’s next decision; they are delivered at declared decision boundaries, between state mutations. Team profiles, output kernels, and all elementary calls have declared finite token, tool, and timeout limits. A consistent implementation reserves an execution quota for each Agent, each edge program, and the output operation. For integer resources, $\lfloor \mathbf{B}^{\text{exec}} / (|V| + |E| + 1) \rfloor$ is one such allocation; rounding leaves unused resources. All nested calls are charged to the responsible quota.

Lemma B.10 (Finite runtime and budget accounting). *Under the preceding bounded-call, quota, and scheduling contracts, the number of elementary gate evaluations is at most $|E|r_{\max}$ for the one-pass schedule and $R_{\text{comm}}|E|r_{\max}$ for the cyclic schedule, taking $r_{\max} \geq 1$. Execution consumes at most its reserved resource vector and returns an answer or a declared failure in finite time when timeouts are enforced.*

Proof. Each outer visit runs at most $r_{\max}$ elementary gate evaluations; FinalOnly runs none and OneWay at most one. There are at most $|E|$ visits per sweep and at most the declared number of sweeps. Every local operation terminates or times out, so a finite number of them terminates. Charging all nested costs to nonoverlapping quotas bounds their sum by the execution reservation. Forced stopping can only reduce the number of operations. $\square$

The fixed order makes runs reproducible, because updates from two incoming peers can depend on their order.

DeGroot linear baseline. Following the consensus model of DeGroot (1974), take a hypothetical linear opinion vector $h^{(r)} \in \mathbb{R}^n$ and suppose $h^{(r+1)} = Wh^{(r)}$, where $W \geq 0$, $W\mathbf{1} = \mathbf{1}$, and W is primitive: some integer $m \geq 1$ has all entries of W^m positive.

Lemma B.11 (Primitive stochastic linear baseline). *Under those conditions, $W^r \rightarrow \mathbf{1}\pi^\top$ for a unique probability vector π satisfying $\pi^\top W = \pi^\top$. Hence the linear baseline reaches the consensus $\pi^\top h^{(0)}$, which the initial opinions and the weights W fix without reference to evidence about correctness.*

Proof. Put $a = \min_{i,j}(W^m)_{ij} > 0$. For any real vector h , subtract the common contribution $a \sum_j h_j$ from every coordinate of $W^m h$. The remaining nonnegative row weights sum to $1 - na$, so $\max(W^m h) - \min(W^m h) \leq (1 - na)(\max h - \min h)$. Repeated blocks contract the diameter to zero (immediately if $na = 1$). Stochastic averaging makes minima nondecreasing and maxima nonincreasing, so each initial vector converges to a common scalar. Applying this fact to each coordinate basis vector gives $W^r \rightarrow \mathbf{1}\pi^\top$, with $\pi \geq 0$ and $\pi^\top \mathbf{1} = 1$. Taking limits in $W^{r+1} = W^r W$ gives stationarity. Any stationary probability row vector $q^\top$ satisfies $q^\top = q^\top W^r \rightarrow \pi^\top$, proving uniqueness. $\square$

The evidence gate replaces this fixed weighting with the evidence comparison of Eq. B.6.

B.7 SINGLE-WRITER SERIALIZATION OF INTERNAL MUTATIONS

S1–S3. **S1 (enforced exclusive permission)** requires that one selected Executor is the only internal component authorized to issue a state-changing tool request, including indirect requests through delegated tools; Advisors cannot bypass this check. **S2 (serial completion)** requires that the Executor issue its next mutation only after the previous mutation and all its asynchronous effects have completed or been safely resolved, with no subsequent overlapping mutation. **S3 (scope)** restricts the single-writer assertion to components controlled by the team. A state composition claim additionally excludes external interleaving mutations or conditions explicitly on their occurrence in the environment model.

Theorem B.12 (Internal single-writer serialization). *Under S1–S3, every finite runtime trace has an ordered sequence of internal mutations $w_1, \dots, w_m$ such that*

$$w_1 \prec w_2 \prec \dots \prec w_m, \quad (\text{B.9})$$

where $\prec$ means that the previous mutation's effects complete before the next mutation starts. If no external mutation interleaves, then, after fixing the realized tool randomness $\xi_1, \dots, \xi_m$, the state is

$$S_m = f_{w_m, \xi_m} \circ \dots \circ f_{w_1, \xi_1}(S_0). \quad (\text{B.10})$$

In particular, internal writes are totally ordered, so the internal schedule is serial and hence serializable in the sense of Papadimitriou (1979).

Proof. S1 makes every internal mutation an Executor mutation. Order them by their issue times. S2 ensures that the effects of w_j complete before w_{j+1} begins, establishing the strict order. With no external interleaving, the input state of w_1 is S_0 and its output is $f_{w_1, \xi_1}(S_0)$. Inductively the input of w_{j+1} is the completed output of w_j . Composition proves the formula. Adaptive selection of w_{j+1} from prior observations does not alter the argument for the realized trace. $\square$

B.8 NORMALIZED REVERSE PATHS AND GRAPH-LEVEL CTB

We now fix c and distinguish an ideal normalized outcome model from valid-only replay. Let $\mathcal{X}_c$ be the complete terminal outcome set. It is either $\mathcal{G}_c$ for a process normalized on valid completion, or $\mathcal{G}_c \cup \{\dagger\}$ when abort is modeled explicitly. If there are multiple failure types, the same reasoning applies with a finite set of failure outcomes. Use z for an arbitrary outcome and G for a valid team. Every participating edge lies on a complete source-to-terminal path. The notation g_z includes an absorbing abort node when present.

T1–T4. **T1 (fixed terminating construction)** requires a fixed finite sufficient-state construction DAG with one source, normalized outgoing forward probabilities, and no unmodeled dead ends. Every forward path terminates and every permitted reverse path reaches this source. **T2 (terminal identity)** requires one absorbing terminal per canonical outcome, with all its stop transitions and the same execution budget and reward snapshot attached to that terminal. **T3 (matched normalized supports)** requires strictly positive forward probabilities on the participating transitions and a reverse kernel $b_z(e \mid v')$ normalized over the actual incoming parent–action pairs $e = (v, u, v')$ in the z -ancestral subgraph. These supports define exactly the paths used in both sides of CTB; no hidden parent or extra terminal alias is allowed. **T4 (fixed weights and exact balance)** requires a fixed finite $r_z > 0$ for every outcome, fixed $\beta > 0$, $Z > 0$, and zero unscaled CTB residual on every complete path. For explicit abort, $r_\dagger$ is also fixed and positive. **T4' (valid-path balance)** is T4 restricted to valid complete paths, with no constraint on abort paths; Algorithm C.1 trains under T4'. T4 and T4' are the ideal consistency premises, and the approximate-balance results below cover finite residuals.

For $\zeta = (v_0, e_1, v_1, \dots, e_T, g_z)$ define

$$P_F(\zeta) = \prod_{t=1}^T p_F(e_t \mid v_{t-1}), \quad Q_z(\zeta) = \prod_{t=1}^T b_z(e_t \mid v_t), \quad w_z = r_z^\beta, \quad W = \sum_{z \in \mathcal{X}_c} w_z. \quad (\text{B.11})$$

$Z = Z_\psi(c)$ is a learned positive scalar, whereas W is the prescribed sum of terminal weights. They coincide only under the hypotheses proved below. The unscaled residual and its length-scaled loss are

$$\rho(\zeta) = \log Z + \log P_F(\zeta) - \log w_z - \log Q_z(\zeta), \quad L_{\text{CTB}}(\zeta) = \rho(\zeta)^2 / T(\zeta)^2. \quad (\text{B.12})$$

This ρ is the main-text $\rho_{\text{TB}}(\zeta)$, and b_z is the main-text learned kernel b_ϕ at terminal g_z . Length normalization reweights the loss and leaves the probability kernels and the zero-residual set unchanged.

Lemma B.13 (Reverse-path normalization). *Under T1–T3, $\sum_{\zeta \in \Omega_z} Q_z(\zeta) = 1$ for every z .*

Proof. Let $b_z^{\leftarrow}(v)$ be the total mass of reverse paths from v to the source, within the z -ancestral subgraph. The empty source path has mass $b_z^{\leftarrow}(v_0) = 1$. Partitioning by the first reverse transition gives

$$b_z^{\leftarrow}(v') = \sum_{e=(v, u, v') \in \text{In}_z(v')} b_z(e \mid v') b_z^{\leftarrow}(v). \quad (\text{B.13})$$

In topological order every parent has already been assigned mass one. T3 then makes the sum equal to one. Finiteness and the unique reachable source exclude leftover reverse mass. At g_z this gives

$$\sum_{\zeta \in \Omega_z} Q_z(\zeta) = 1. \quad (\text{B.14})$$

Distinct actions sharing a parent and child enter this recursion as separate terms, so the backward normalization runs over parent–action pairs. $\square$

Theorem B.14 (TB consistency for canonical team outcomes). *Under T1–T4, the forward outcome law is $P_F(z) = w_z/W$, and $Z = W$. In particular, in the normalized valid-completion model,*

$$\mu_\theta^\vee(G \mid c) = \frac{r_G^\beta}{\sum_{G' \in \mathcal{G}_c} r_{G'}^\beta}. \quad (\text{B.15})$$

For explicit abort the same valid-graph expression holds after conditioning on $\vee$, whereas the unconditional denominator also contains $w_\dagger$.

Proof. Step 1: path identity. Exponentiating $\rho(\zeta) = 0$ gives

$$ZP_F(\zeta) = w_z Q_z(\zeta). \quad (\text{B.16})$$

Step 2: sum the paths of one team. T2 makes Ω_z precisely the disjoint complete paths of outcome z . The single fixed weight w_z can be factored out. Lemma B.13 therefore yields

$$ZP_F(z) = \sum_{\zeta \in \Omega_z} w_z Q_z(\zeta) = w_z. \quad (\text{B.17})$$

Step 3: identify the normalizer. All outcomes, including abort when present, partition the terminating process. Thus

$$Z = Z \sum_{z \in \mathcal{X}_c} P_F(z) = \sum_{z \in \mathcal{X}_c} w_z = W. \quad (\text{B.18})$$

Substituting proves the outcome law. In the explicit-abort model, $P_F(\vee) = \sum_G w_G/W$; dividing $P_F(G)$ by this quantity proves the conditional valid-graph formula. $\square$

At zero residual, $P_F(\zeta \mid G) = Q_G(\zeta)$, so the backward kernel also sets how probability spreads over the construction orders of a team.

Feasibility of exact balance. When one normalized reverse kernel $b(e \mid v')$ is shared across endpoints, an unrestricted finite tabular transition model has a solution. Set $F(g_z) = w_z$ and, in reverse topological order, set $F(v) = \sum_{e=(v,u,v')} F(v')b(e \mid v')$. Every participating vertex has a positive descendant weight, so $F(v) > 0$. Define $p_F(e \mid v) = F(v')b(e \mid v')/F(v)$. These forward probabilities normalize by construction; incoming flow to v' sums to $F(v')$ by reverse normalization. Thus source flow equals total terminal weight, and multiplication along a path telescopes to Eq. B.16. This establishes feasibility for a tabular transition model.

B.9 TERMINAL ALIASES, ABORT, AND THE MEANING OF VALID-ONLY REPLAY

Lemma B.15 (Alias weighting). *Suppose outcome-level TB holds on terminal nodes, but graph G has a set $\mathcal{Z}_G$ of terminal aliases with weights w_z . Its graph marginal is proportional to $\sum_{z \in \mathcal{Z}_G} w_z$. In particular, assigning the full weight r_G^β to each of $m(G)$ aliases gives*

$$P_F(G) = \frac{m(G)r_G^\beta}{\sum_{G'} m(G')r_{G'}^\beta} \quad (\text{B.19})$$

in a valid-only normalized model. The desired graph law is recovered by merging terminal nodes with their full incoming supports, or by assigning alias weights $w_z = \omega(z \mid G)r_G^\beta$, where $\omega(z \mid G) \geq 0$ and $\sum_{z \in \mathcal{Z}_G} \omega(z \mid G) = 1$.

Proof. Sum the terminal outcome probabilities of Theorem B.14 over $\mathcal{Z}_G$. Full repeated weights produce the multiplicity factor; weights summing to r_G^β remove it. Zero-weight aliases must be removed from the positive-log support, or all retained ω must be strictly positive. $\square$

Theorem B.16 (Raw valid-only balance). *Let P_F be a normalized construction process that can abort, with $p_\vee > 0$, and assume T1–T3 and T4': the reverse distributions normalize on every valid Ω_G , the fixed weights $w_G > 0$ are consistent, and $ZP_F(\zeta) = w_G Q_G(\zeta)$ holds on every valid*

complete path, with no balance constraint on abort paths. Then

$$\mu_\theta^\vee(G) = \frac{w_G}{W_V}, \quad Z = \frac{W_V}{p_V}, \quad W_V := \sum_{G \in \mathcal{G}_c} w_G. \quad (\text{B.20})$$

Proof. Summing the valid path identities for a fixed graph gives $ZP_F(G) = w_G$. Summing over valid graphs gives $Zp_V = W_V$. Dividing the first equality by the second proves the conditional law. $\square$

Algorithm correspondence. This is the precise ideal interpretation when Algorithm C.1 retains valid rollouts and evaluates their raw construction likelihoods. Rejection-based replay keeps those likelihoods, and the conditional target follows with the scale $Z = W_V/p_V$. Explicit abort rewards or a normalized valid-conditioned forward law give the two alternative conventions, each with its own normalizer. Because Z absorbs p_V , valid-path balance leaves the abort rate to the legal masks and budgets; a positive abort reward under T4 (Theorem B.14) prices aborts explicitly.

Graph-level temperature interpretation. For a finite nonempty valid set and fixed $r_G > 0$, define $q_\beta(G) = r_G^\beta/W_V$ and $H(p) = -\sum_G p(G) \log p(G)$, with $0 \log 0 = 0$. Direct expansion gives

$$\text{KL}(p||q_\beta) = \log W_V - \beta \mathbb{E}_p[\log r_G] - H(p). \quad (\text{B.21})$$

Hence q_β is the unique optimum of $\beta \mathbb{E}_p \log r_G + H(p)$ over graph distributions, since KL is nonnegative and zero exactly at $p = q_\beta$. The entropy is graph entropy, so duplicated construction paths earn no extra weight. As $\beta \downarrow 0$ the law becomes uniform on graphs, and as $\beta \rightarrow \infty$ it becomes uniform on the tied reward maxima; β thus sets how strongly CTB favors the best teams while keeping the others. It also sets the mass left on failing teams: a team with $r_{\text{ans}} = 0$ has $r_G = \varepsilon_{\min}$, so the $|F|$ such teams carry total weight $|F|\varepsilon_{\min}^\beta$, and $\beta \log(r_s/\varepsilon_{\min}) \geq \log |\mathcal{G}_c|$ keeps this weight below that of one team with reward r_s .

B.10 APPROXIMATE BALANCE AND PERTURBATION BOUNDS

The next statements use normalized true path probabilities and a fixed finite support, as in T1–T3. They apply either to all outcomes or, using Theorem B.16, to the graph law conditional on valid completion. Write $q(z) = w_z/W$ in the normalized model.

Theorem B.17 (Exact residual tilt and uniform distribution error). *For a possibly nonzero residual from Eq. B.12, let $a_z = \mathbb{E}_{\zeta \sim Q_z} \exp(\rho(\zeta))$. Then*

$$P_F(z) = \frac{q(z)a_z}{\mathbb{E}_q a_z}. \quad (\text{B.22})$$

If $|\rho(\zeta)| \leq \eta$ for every complete path, then

$$e^{-2\eta} \leq \frac{P_F(z)}{q(z)} \leq e^{2\eta}, \quad \text{TV}(P_F, q) \leq \tanh(\eta/2), \quad \max\{\text{KL}(P_F||q), \text{KL}(q||P_F)\} \leq 2\eta. \quad (\text{B.23})$$

Here $\text{TV}(p, q) = \frac{1}{2} \sum_z |p(z) - q(z)|$. The same bounds hold for valid-conditional graph laws when the uniform residual bound is only over valid paths and raw valid-only likelihoods are used.

Proof. Exponentiation gives $ZP_F(\zeta) = w_z Q_z(\zeta) e^{\rho(\zeta)}$. Sum within Ω_z to obtain $ZP_F(z) = w_z a_z$, then sum over outcomes to obtain $Z = W \mathbb{E}_q a_z$. This proves the tilt. Under the uniform bound, $a_z \in [m, M] = [e^{-\eta}, e^\eta]$, as does $A = \mathbb{E}_q a_z$; thus $P_F(z)/q(z) = a_z/A$ lies in $[e^{-2\eta}, e^{2\eta}]$. Taking the appropriate expectation of each log ratio bounds both KLs.

For total variation, convexity of $|a - A|$ bounds it on $[m, M]$ by the chord joining its endpoint values. Taking expectations gives $\mathbb{E}_q |a_z - A| \leq 2(M - A)(A - m)/(M - m)$. Therefore, for $M > m$,

$$\text{TV}(P_F, q) \leq \frac{(M - A)(A - m)}{A(M - m)} = \frac{M + m - A - Mm/A}{M - m} \leq \frac{\sqrt{M} - \sqrt{m}}{\sqrt{M} + \sqrt{m}} = \tanh(\eta/2),$$

using $A + Mm/A \geq 2\sqrt{Mm}$. For $M = m$ the distributions coincide. In the raw valid-only case, summation gives $Zp_V = W_V \mathbb{E}_{q^\vee} a_G$; division yields the same tilt for $\mu_\theta^\vee$, so the proof is unchanged. $\square$

For a length-scaled bound $|\rho/T| \leq \epsilon$, take $\eta = T_{\max}\epsilon$.

Corollary B.18 (Fixed reward perturbation and likelihood error). *Suppose a fit uses fixed rewards $\hat{r}_z > 0$ with $\sup_z |\log \hat{r}_z - \log r_z| \leq d_R$ and its true residual relative to $\hat{r}$ is bounded by η . Relative to the target from r , the ratio, TV, and KL bounds above hold with $\eta + \beta d_R$ in place of η . If the recorded residual differs from this true residual by at most d_L on every path, replace this quantity by $\eta + d_L + \beta d_R$.*

Proof. Relative to $w_z = r_z^\beta$, the tilt factor is $(\hat{r}_z/r_z)^\beta a_z$, in $[e^{-(\eta+\beta d_R)}, e^{\eta+\beta d_R}]$. Apply the previous proof. The residual triangle inequality proves the likelihood-error extension. For example, uniform per-transition log errors d_F, d_B and log-normalizer error d_Z give $d_L \leq d_Z + T_{\max}(d_F + d_B)$. $\square$

The errors compare recorded scores with the normalized sampling and reverse probabilities on the same support.

Lemma B.19 (What a finite-support population loss certifies). *Let ν be a fixed distribution on the finite complete-path set, with $\nu(\zeta) \geq \nu_{\min} > 0$ for every participating path. If $\mathbb{E}_\nu[(\rho/T)^2] \leq \epsilon^2$, then $\sup_\zeta |\rho(\zeta)| \leq T_{\max}\epsilon/\sqrt{\nu_{\min}}$.*

Proof. For any path, nonnegativity gives $\nu_{\min}\rho(\zeta)^2/T(\zeta)^2 \leq \mathbb{E}_\nu[(\rho/T)^2] \leq \epsilon^2$. Take square roots and use the length cap. $\square$

B.11 REWARD SNAPSHOTS, POSITIVE SUPPORT, AND MOVING COUNTERS

Fix a finite graph-description length $L_\chi(G) \geq 0$ and a fixed classification $a_\chi(G)$ for structural reference comparisons. For each nonempty class let $L_{\min,\chi}(a)$ be the minimum description length in that class, and set $L_{\text{rel}}(G) = L_\chi(G) - L_{\min,\chi}(a_\chi(G)) \geq 0$. The code and classes are fixed parts of the reward specification, and the results hold for any such specification.

At the beginning of batch k , freeze finite counts $0 \leq s_k(G) \leq n_k(G)$ and use

$$\hat{p}_k(G) = \frac{s_k(G) + 1/2}{n_k(G) + 1}, \quad \tilde{R}_k(G, x, \xi) = \varepsilon_{\min} + r_{\text{ans}}(G, x, \xi)\hat{p}_k(G)e^{-\lambda L_{\text{rel}}(G)}, \quad (\text{B.24})$$

where $\varepsilon_{\min} > 0$, $\lambda \geq 0$, and $0 \leq r_{\text{ans}} \leq 1$. The count factor $\hat{p}_k(G)$ is the posterior mean of the success rate under the Jeffreys prior $\text{Beta}(1/2, 1/2)$ (Jeffreys, 1946). The results below hold for every $\lambda \geq 0$. The method uses $\lambda = 1$ with code lengths in nats, so $e^{-L_{\text{rel}}(G)}$ is the minimum-description-length prior of G relative to the shortest team of its class (Rissanen, 1978); the structural-prior ablation sets $\lambda = 0$. Each execution record increments the counts once, however often it is copied. All rewards in a batch use its pre-batch counts; completed outcomes update counts only after that batch. Pooling counts across tasks or graph classes uses a fixed pooling rule.

Lemma B.20 (Reward support and the posterior's model). *The counts in Eq. B.24 give $0 < \hat{p}_k(G) < 1$ and $\varepsilon_{\min} \leq \tilde{R}_k \leq \varepsilon_{\min} + 1$. Under a model of conditionally independent Bernoulli outcomes with one fixed success parameter p and prior $\text{Beta}(1/2, 1/2)$, the posterior is $\text{Beta}(s + 1/2, n - s + 1/2)$ and its mean is $(s + 1/2)/(n + 1)$.*

Proof. The numerator is strictly between zero and $n + 1$, proving the first claim. The remaining multiplicative factors belong to $[0, 1]$, proving the reward bounds. For the posterior, multiply the prior density proportional to $p^{-1/2}(1-p)^{-1/2}$ by the Bernoulli likelihood $p^s(1-p)^{n-s}$ and normalize. Its two beta parameters and their ratio give the stated distribution and mean. $\square$

The positive floor keeps log rewards finite and gives failed outcomes a small positive target mass.

Lemma B.21 (Counter-induced target drift). *After $a \geq 0$ new unique outcomes with $b \in \{0, \dots, a\}$ successes,*

$$\hat{p}_{n+a} - \hat{p}_n = \frac{b - a\hat{p}_n}{n + a + 1}, \quad |\hat{p}_{n+a} - \hat{p}_n| \leq \frac{a}{n + a + 1}. \quad (\text{B.25})$$

If the answer-score table and structural code are fixed between two reward snapshots, write $r(G) = \varepsilon_{\min} + c_G \hat{p}(G)$ with $c_G = r_{\text{ans}}(G)e^{-\lambda L_{\text{rel}}(G)} \in [0, 1]$. The per-graph log-reward difference between the two snapshots, with rewards r and r' , is then at most

$$\frac{c_G a}{(n + a + 1) \min\{r, r'\}} \leq \frac{a}{(n + a + 1) \varepsilon_{\min}}. \quad (\text{B.26})$$

A uniform such bound can be substituted for d_R in Corollary B.18.

Proof. Subtract the two ratios, writing $s + 1/2 = (n + 1)\hat{p}_n$, to obtain the identity. Since $0 \leq b \leq a$ and $0 < \hat{p}_n < 1$, its numerator has absolute value at most a . The reward difference is $c_G(\hat{p}' - \hat{p})$, so $|r' - r| \leq c_G a / (n + a + 1)$. Between r and r' the derivative of $\log$ is at most $1 / \min\{r, r'\}$, and the mean-value theorem gives the first bound; $c_G \leq 1$ and $\min\{r, r'\} \geq \varepsilon_{\min}$ give the second. $\square$

Corollary B.22 (Movement of the self-generated target). *Fix the answer-score table, structural code, graph support, and β between snapshots k and $k + 1$, and let $q_k(G) \propto r_k(G)^\beta$ be the target of snapshot k . If team G gains a_G unique outcomes between the snapshots and $d = \max_G c_G a_G / ((n_k(G) + a_G + 1) \min\{r_k(G), r_{k+1}(G)\})$, then*

$$e^{-2\beta d} \leq \frac{q_{k+1}(G)}{q_k(G)} \leq e^{2\beta d}, \quad \text{TV}(q_{k+1}, q_k) \leq \tanh(\beta d/2). \quad (\text{B.27})$$

Proof. Lemma B.21 gives $|\log r_{k+1}(G) - \log r_k(G)| \leq d$ for every G . Hence $q_{k+1}(G) = q_k(G)\tau_G/\mathbb{E}_{q_k}\tau$ with $\tau_G = (r_{k+1}(G)/r_k(G))^\beta \in [e^{-\beta d}, e^{\beta d}]$. This is the tilt of Theorem B.17 with $\eta = \beta d$, and its ratio and total-variation bounds apply unchanged. $\square$

Teams with $a_G = 0$ or $c_G = 0$ contribute nothing to d , so d is set by the sampled teams with positive answer scores and shrinks as their records accumulate. It holds for any realized counts, whatever policy selected the teams.

B.12 THE PRACTICAL TRAINING LOSS

For a fixed replay distribution ν_k and fixed state weighting d_k , the regularized training surrogate is

$$\mathcal{L}_k(\vartheta) = \mathbb{E}_{\zeta \sim \nu_k} \frac{\rho_\vartheta(\zeta)^2}{T(\zeta)^2} + \alpha_{\text{KL}} \mathbb{E}_{v \sim d_k} \text{KL}(\bar{\pi}_\theta(\cdot | v) \| \pi_{\text{ref}}(\cdot | v)). \quad (\text{B.28})$$

Here $\vartheta = (\theta, \phi, \psi)$ collects all learned parameters, the reference policy is normalized on the same action domain and positive where the forward policy is positive, and $\alpha_{\text{KL}} \geq 0$. During an update, replay samples and recorded reward targets are treated as fixed. Shared parameters must be differentiated in *all* terms where they occur. This specification distinguishes a replay gradient from the full derivative of an on-policy sampling distribution.

Example B.23 (KL regularization tilts the balance point). Consider two one-step terminals with unit rewards and backward probabilities one. Let the forward law be $(p, 1 - p)$ and the reference $(0.9, 0.1)$. At $p = 1/2$, $Z = 2$, both TB residuals vanish and the squared-residual gradient is zero, while

$$\left. \frac{d}{dp} \alpha_{\text{KL}} \text{KL}((p, 1 - p) \| (0.9, 0.1)) \right|_{p=1/2} = -\alpha_{\text{KL}} \log 9.$$

With $\alpha_{\text{KL}} > 0$ the regularized optimum therefore moves from the reward-proportional point toward a fixed reference, and α_{KL} sets the size of this move.

Lemma B.24 (Proximal reference keeps balanced directors fixed). *Let $\pi_{\text{ref}} = \bar{\pi}_{\theta_k}$ be the director that collected batch k , and fix the rewards. If $\vartheta_k = (\theta_k, \phi_k, \psi_k)$ has zero residual on every replayed path, then ϑ_k minimizes $\mathcal{L}_k$. Hence a balanced director is a fixed point of the regularized update, and the KL term only damps moves away from it.*

Proof. Both terms of Eq. B.28 are nonnegative. At ϑ_k the residual term vanishes by assumption, and the KL term vanishes because $\bar{\pi}_{\theta_k} = \pi_{\text{ref}}$. Hence $\mathcal{L}_k(\vartheta_k) = 0 \leq \mathcal{L}_k(\vartheta)$ for every ϑ . $\square$

Because the reference moves with the director, the tilt of the preceding example vanishes at every balanced director, and the recursion keeps the reward-proportional point of each snapshot. This choice mirrors the proximal updates of TRPO and PPO (Schulman et al., 2015; 2017), which regularize each policy update toward the policy that collected the data.

Theorem B.25 (Random log-reward regression). *Fix a replay path ζ ending at G , its length, and its context. Assume execution randomness is fresh conditional on G, c and independent of the construction path, and $\log \tilde{R}_k(G, x, \xi)$ has finite second moment. Put $m_G = \mathbb{E}[\log \tilde{R}_k | G, c]$,*

$\sigma_G^2 = \text{Var}(\log \tilde{R}_k \mid G, c)$, and $A_\vartheta(\zeta) = \log Z + \log P_F(\zeta) - \log Q_G(\zeta)$. Then

$$\mathbb{E} \left[\frac{(A_\vartheta(\zeta) - \beta \log \tilde{R}_k)^2}{T^2} \mid \zeta, c \right] = \frac{(A_\vartheta(\zeta) - \beta m_G)^2}{T^2} + \frac{\beta^2 \sigma_G^2}{T^2}. \quad (\text{B.29})$$

In an unregularized fixed-replay model that can simultaneously realize all these conditional means, the corresponding ideal graph weights are $\exp(\beta m_G)$, the tempered geometric-mean reward of each team.

Proof. Write $\log \tilde{R}_k = m_G + \epsilon$ with conditional mean-zero ϵ and variance σ_G^2 . Expanding the square makes the cross term vanish and gives the identity. The variance term is independent of A_ϑ for fixed replay and context. If all means can be realized, the remaining squared terms are minimized at $A_\vartheta = \beta m_G$. These are exactly the fixed-weight path identities with $w_G = \exp(\beta m_G)$, to which the appropriate normalized or valid-only TB theorem applies. $\square$

Example (CTB favors reliable teams). One team with reward 0.1 or 0.9 equally often and another with constant reward 0.5 have the same mean reward, but their geometric-mean weights are 0.3 and 0.5. For $\beta = 1$, CTB samples them with probabilities 0.375 and 0.625 in the realizable two-terminal model, so under noisy rewards it prefers the team that succeeds consistently.

Lemma B.26 (Residual-gradient bound). *For a fixed sampled path, fixed reward, and differentiable finite residual,*

$$g(\zeta) := \nabla_\vartheta L_{\text{CTB}}(\zeta) = \frac{2\rho(\zeta)}{T(\zeta)^2} \nabla_\vartheta \rho(\zeta). \quad (\text{B.30})$$

If $T \geq T_{\min} \geq 1$ and $\|\nabla_\vartheta \rho(\zeta)\| \leq G_0 < \infty$ on the distribution under consideration, then

$$\mathbb{E}\|g - \mathbb{E}g\|^2 \leq \mathbb{E}\|g\|^2 \leq \frac{4G_0^2}{T_{\min}^4} \mathbb{E}[\rho^2]. \quad (\text{B.31})$$

Proof. Treating the recorded length as fixed, the chain rule gives the gradient. Squaring its norm and using the two uniform bounds yields the last inequality after expectation. Subtracting $\|\mathbb{E}g\|^2 \geq 0$ proves the first. Thus, as the mean-square residual shrinks, the variance of this CTB gradient component shrinks with it. $\square$

B.13 OCCUPANCY FLOWS, CONDITIONAL BRIDGES, AND PATH DIAGNOSTICS

Fix the normalized outcome model of Section B.8. For any learned forward policy, the reverse kernels define the reward-weighted path measure

$$M(\zeta) = w_z Q_z(\zeta), \quad \sum_{\zeta} M(\zeta) = W, \quad P^*(\zeta) = M(\zeta)/W. \quad (\text{B.32})$$

This measure allocates the target over paths. For a construction vertex v or transition e , define

$$\begin{aligned} F_z(v) &= \sum_{\zeta \in \Omega_z: v \in \zeta} M(\zeta), & f_z(e) &= \sum_{\zeta \in \Omega_z: e \in \zeta} M(\zeta), \\ F(v) &= \sum_z F_z(v), & f(e) &= \sum_z f_z(e). \end{aligned} \quad (\text{B.33})$$

States and transitions occur at most once on a construction-DAG path. Source flow is outgoing flow $F(v_0) = W$; terminal flow is incoming flow $F(g_z) = w_z$. Incoming and outgoing flows agree at every interior vertex.

Theorem B.27 (Occupancy conservation and terminal-conditioned bridge). *Under T1–T3, the path measure in Eq. B.32 satisfies $F_z(v) = \sum_{e \in \text{In}(v)} f_z(e) = \sum_{e \in \text{Out}(v)} f_z(e)$ at every interior v , and $f_z(e) = F_z(v') b_z(e \mid v')$ for $e = (v, u, v')$ in the z -ancestral subgraph, with zero occupancy elsewhere. If additionally exact balance holds, then*

$$F(v) = W P_F(v \text{ is visited}), \quad f(e) = F(v) p_F(e \mid v).$$

For a nonterminal visited state v with $h_z(v) := P_F(z \mid v \text{ is visited}) > 0$, define the forward bridge

$$p_F^z(e \mid v) = p_F(e \mid v) \frac{h_z(v')}{h_z(v)}. \quad (\text{B.34})$$

It is normalized over outgoing edges and obeys

$$f_z(e) = F_z(v)p_F^z(e | v) = F_z(v')b_z(e | v'). \quad (\text{B.35})$$

Thus the bridge supplies the forward side of an endpoint-conditioned occupancy.

Proof. For each interior visit of a complete path, exactly one incoming and one outgoing edge are used. Summing these pathwise indicator identities against M proves conservation. In the reverse process starting at g_z , once v' is visited the next reverse edge has law $b_z(\cdot | v')$. All remaining reverse prefixes have mass one by Lemma B.13; hence its edge-visit mass is $F_z(v')b_z(e | v')$.

At zero residual, $M(\zeta) = WP_F(\zeta)$. Summing gives the state-visit identity. A sufficient Markov state makes the next-edge law independent of the earlier prefix, so $f(e) = F(v)p_F(e | v)$. The same Markov property gives $h_z(v) = \sum_{e=(v,u,v')} p_F(e | v)h_z(v')$, proving bridge normalization. Furthermore $F_z(v) = F(v)h_z(v)$ and $f_z(e) = F(v)p_F(e | v)h_z(v')$. Substitution proves the two-sided conditional balance. $\square$

Example. For a source that chooses between two unit-reward terminals with probability $1/2$ each, the bridge toward the first terminal takes that branch with probability one, so $F_1(v_0)p_F^1(g_1 | v_0) = 1 = F_1(g_1)b_1$.

State-flow ratios and edge flows. Whenever a compatible local balance identity holds, its ratio is a ratio of state occupancies, $p_F^z/b_z = F_z(v')/F_z(v)$ on positive conditional support, while the reward an edge carries is its edge flow $f_z(e)$. For two parents of flows 1 and 99 entering a common child of flow 100 deterministically, the backward probabilities are 0.01 and 0.99, the flow ratios are 100 and 100/99, and the edge flows are 1 and 99; we therefore rank edges by $f_z(e)$.

Definition B.28 (Semantic transition occupancy). For a semantic communication decision $a = (i, j, p)$, let $N_a(\zeta)$ count accepted insertions of that edge on a valid complete trajectory and set $N_a(\zeta) = 0$ on abort trajectories. Define

$$H_c(a) = \sum_{\zeta} M(\zeta)N_a(\zeta) = W\mathbb{E}_{P^*}N_a(\zeta). \quad (\text{B.36})$$

Under the add-only interface with duplicate insertions excluded, $N_a(\zeta) = \mathbf{1}\{a \in E(G_\zeta)\}$, so $H_c(a) = \sum_{G: a \in E(G)} w_G$. The normalized statistic $H_c(a)/W$ is the target probability of returning a valid team containing that edge; $H_c(a)/W_V$ conditions this probability on valid completion.

The last identity follows by summing Q_G over Ω_G . It is independent of the allocation between different orders for a fixed graph, so semantic occupancy reads the canonical team rather than any one construction order.

Estimators and their sampling law. If $\zeta_1, \dots, \zeta_n$ are independent samples from P^* , then $n^{-1} \sum_i N_a(\zeta_i)$ is unbiased for $H_c(a)/W$ by linearity of expectation. Off policy, for a normalized proposal ν positive wherever $M(\zeta)N_a(\zeta) > 0$, the estimator

$$\hat{H}_c(a) = \frac{1}{n} \sum_{i=1}^n \frac{M(\zeta_i)}{\nu(\zeta_i)} N_a(\zeta_i), \quad \zeta_i \sim \nu, \quad (\text{B.37})$$

is unbiased for $H_c(a)$, since summing $\nu(\zeta)$ cancels the denominator. In raw valid-only exact balance, W_V and the valid-conditioned target replace W and P^* . At finite residual, unweighted forward frequencies describe the learned policy.

Cached path diagnostics and computation. For a scored path define $d_t(\zeta) = \log p_F(e_t | v_{t-1}) - \log b_z(e_t | v_t)$. Its complete sum has the exact algebraic identity

$$\sum_{t=1}^T d_t(\zeta) = \log w_z - \log Z + \rho(\zeta). \quad (\text{B.38})$$

This path diagnostic holds at any residual. The d_t values are computed by arithmetic from cached forward and backward log probabilities.

B.14 PROOF DEPENDENCIES

The preceding results form one dependency chain. Checked construction determines the legal support. Normalized action and reverse kernels turn this support into path laws. A fixed terminal identity and reward snapshot make the TB path sum meaningful, and exact balance on all paths (T4) or on valid paths (T4'), or the uniform residual bounds, link these laws to the prescribed terminal distribution. The reward-snapshot corollary bounds how this target moves between rounds, and the communication and single-writer results are runtime contracts that hold alongside this chain.

Result	Premises	Main-text result
Construction	C1 for soundness; C1–C3 for coverage	Proposition 1: valid canonical teams, every target reachable
Evidence update	Version-bound, authentic applicable records; ordered gate	Proposition 2: version-bound evidence, flip and correction identities
Runtime	Fixed schedule, finite rounds, charged bounded calls	Finite, budgeted execution
Single writer	S1–S3	Proposition 2: serialized internal writes
Graph sampling	T1–T3 with T4' (Algorithm C.1) or T4; uniform residual control	Proposition 3: reward-proportional teams
Proximal update	π_{ref} is the director that collected the batch	Balanced directors stay fixed (Section 4.3)
Target movement	Fixed answer-score table, structural code, and team support	Proposition 3: bounded target drift
Occupancy	Normalized reward path measure	Path diagnostics

All finite sums above use the declared supports, all reverse normalizers include parent–action multiplicity, and all probability comparisons hold at a fixed context. Conditional events of probability zero use stated conventions.

C ALGORITHMIC DETAILS

C.1 TRAINING PROCEDURE

Algorithm C.1 expands the self-improvement trajectory of Eq. 2 and the procedure specified in Section 4.3. Each round keeps the Agent population and executor fixed, collects several team rollouts for each sampled task, and updates the director, backward policy, and normalizer; its unique outcomes then update the reliability counters used by the next round.

The released configuration fixes batching, retry handling, and replay.

This pseudocode specifies raw valid-only replay. Its ideal fixed-reward, all-valid-path zero-residual interpretation (T4') is Theorem B.16: $ZP_\theta(V) = W_V$ and only the valid-conditional graph law is reward-proportional. Appendix B.12 treats fresh execution rewards and the KL term. Empty replay skips the loss update; failed execution of a valid graph remains a valid graph sample with its observed reward.

At inference, freeze the Collab-Director, construct a legal team, and execute it with the frozen executor. Ordinary environment observations may influence an executing Agent's local actions within the task.

C.2 RELIABILITY STATISTICS AND REWARD BOOKKEEPING

For a fixed graph/context key with s successes in n unique outcomes, conditionally independent Bernoulli outcomes with one fixed success parameter and a Beta(1/2, 1/2) prior give

$$p \mid \text{records} \sim \text{Beta}(s + 1/2, n - s + 1/2). \quad (\text{C.1})$$

The posterior mean under this model is

$$\hat{p}_k(G) = \frac{s_k(G) + 1/2}{n_k(G) + 1}, \quad (\text{C.2})$$

Algorithm C.1 CollabFlow self-improvement rounds: raw valid-only replay**Require:** Tasks, fixed resource registry and executor, separate $\mathbf{B}^{\text{con}}, \mathbf{B}^{\text{exec}}$, rollout count N .**Ensure:** Updated parameters (θ, ϕ, ψ) and versioned records.

```

1: for each self-improvement round  $k$  do
2:   Freeze the resource context, pre-batch counters, and reward-code version.
3:   Initialize empty replay batch  $\mathcal{B}$  and unique-outcome buffer.
4:   for each sampled task  $x$  and rollout  $i = 1, \dots, N$  do
5:     Construct by sampling from the legal-action kernel of Eq. 5; log every transition, stop,
    and abort with its log probability.
6:     if construction returns a valid canonical team  $G_i$  then
7:       Enter its canonical absorbing terminal with the full legal parent–action support.
8:       Reset the execution instance and use the fixed execution reservation.
9:       Run the version-filtered protocols under the fixed bounded schedule; obtain  $(\hat{y}_i, \xi_i)$ ,
    including declared execution failure.
10:      Compute observed reward  $\tilde{R}_k(G_i, x, \xi_i)$  using the pre-batch snapshot; retain the
    snapshot key.
11:      Add the complete valid path and observed reward to  $\mathcal{B}$ ; buffer the unique execution
    outcome for the later counter update.
12:    else
13:      Record construction abort; do not relabel it as a valid team.
14:    end if
15:  end for
16:  if  $\mathcal{B} \neq \emptyset$  then
17:    Evaluate current forward log probabilities and the backward log probability  $\log b_\phi$  of
    each replayed construction order.
18:    Optimize Eq. 11 with  $\pi_{\text{ref}} = \bar{\pi}_{\theta_k}$ , treating the replay distribution and recorded reward
    targets as fixed during the update.
19:  end if
20:  Update counters once per unique buffered outcome, after the batch.
21: end for

```

with counts keyed by the graph and its task cluster c_x . It is the posterior mean under the Bernoulli model of Lemma B.20, and event deduplication keeps a copied record from counting twice. Task-cluster pooling uses a fixed documented key. Binary success counts are separate from a continuous factual score such as F1.

At update k , all rollouts use pre-batch counts, so each execution is scored against the records that preceded it. The unique-outcome buffer updates counts only after the batch. Each record binds the task key, graph identity, resource and runtime versions, execution reservation, factual score, and snapshot identifier. A fresh execution reward is denoted $\tilde{R}_k(G, x, \xi)$; the fixed r_G in a theorem is a different, explicitly fixed graph functional or immutable table.

For a fixed graph code length L_χ and class map a_χ , define $L_{\text{rel}}(G) = L_\chi(G) - \min_{G': a_\chi(G') = a_\chi(G)} L_\chi(G')$. The nonempty reference class, code, and minima are fixed before the reward snapshot. This definition makes the structural factor nonnegative in length and at most one after exponentiation with $\lambda \geq 0$ (the method uses $\lambda = 1$); Appendix B.11 proves support and counter-drift bounds. The proof holds for any specified finite code.

C.3 ACTION LIKELIHOODS, MASKS, AND BACKWARD TERMS

For a canonical serialized structural action with tokens $(y_1, \dots, y_m)$ and context H , the autoregressive sequence log-probability is

$$\log p_\theta(u \mid H) = \sum_{j=1}^m \log p_\theta(y_j \mid H, y_{<j}). \quad (\text{C.3})$$

The action’s representation includes any required completion delimiter. When the legal policy is a renormalization of raw action probabilities, its probability is obtained by dividing by the total mass

of legal actions. A schema-constrained sampler or rejection sampler records its own probability convention, so the stored log score equals its normalized sampling probability.

Only intended Collab-Director action targets, including termination, contribute to organization-action likelihoods. Task text, runtime feedback, child-Agent outputs, and evaluator labels are context or auxiliary evidence. If sampled reflection text affects an action, its probability must either be included in the modeled path or be handled by an explicitly defined conditional/marginal model. Report the exact mask, serialization, reduction, and reference-policy conventions.

The action log probability is the sum of its token log probabilities, which normalizes over actions; this is the probability that Proposition 3 uses. Log-space evaluation avoids underflow and keeps this definition.

Backward kernel. Under legal-set sampling, the complete paths that reach the canonical terminal g_G are exactly the construction orders $\mathcal{O}(G)$ of G : the prerequisite-respecting orders of its Agent and edge insertions and its output selection, followed by `stop`. The learned kernel b_ϕ scores an order by unbuilding G : starting from G , it removes one component whose removal leaves a legal prefix (an edge, the output selection, or an Agent that no remaining edge or output references) through a softmax over these removable components, and multiplies the choices. Every step is normalized, so $b_\phi(\cdot | G)$ sums to one over $\mathcal{O}(G)$, which is T3 at the terminal; interior history states have one parent and backward probability one. As in TB (Malkin et al., 2022), b_ϕ is learned; Shen et al. (2023) analyze how the backward policy shares credit among paths. The uniform ablation sets $b(\zeta | G) = 1/|\mathcal{O}(G)|$, where $|\mathcal{O}(G)|$ is computed exactly by dynamic programming over subsets of the components of G .

C.4 TRAJECTORY CONTRASTS AND DIAGNOSTIC EVIDENCE

The transition statistic is the reward-weighted occupancy $H_c(a) = \sum_\zeta r_{G_\zeta}^\beta Q_{G_\zeta}(\zeta) N_a(\zeta)$ over valid paths, with abort contributions set to zero. Appendix B.13 defines the state and transition occupancies, the endpoint-conditioned forward bridge, the on-target frequency estimator, and the importance-weighted estimator for a declared proposal. No conditional average over invoking trajectories is substituted for an unconditional occupancy. At finite training error, raw forward frequencies describe the learned policy.

For scored paths, store $d_t = \log p_F - \log b_G$ and the complete-path identity in Eq. B.38. Cached probabilities can be reused; new backward evaluations, parent normalizers, sampling, and arithmetic remain chargeable computation.

An optional same-task contrast uses two recorded runs under the same context, orders them by the recorded factual score (ties are retained as ties), and reports their longest identical sequence of accepted structural actions, first divergence, subsequent messages, evidence versions, costs, and outcomes. All available distinct record pairs may be reported as descriptive error analysis.

C.5 STRUCTURAL COMPLEXITY AND COST ACCOUNTING

For a partial team with n selected Agents, A compatible candidates, P communication protocols, and at most C_o output configurations, a conservative candidate-action bound is

$$|\mathcal{U}(s)| \leq (A - n) + Pn^2 + nK + C_o + 1. \quad (\text{C.4})$$

Binding one of K library skills to a selected Agent adds the nK term. The bound permits all endpoint pairs and therefore remains valid when self-links or additional duplicates are subsequently excluded. A naive complete enumerator over T decisions consequently inspects at most

$$O(T(A + Pn^2 + nK + C_o + 1))$$

candidates when using the maximum n along the trajectory.

These are structural counting bounds on construction. End-to-end cost must additionally include Agent execution, revisions, tool calls, and evaluator work, using the accounting conventions in Appendix F.

D DATASET AND EVALUATION PROTOCOL DETAILS

D.1 BENCHMARK INVENTORY AND SPLITS

Table D.1 lists the twelve benchmarks, their metrics, and the number of evaluation items; the released code contains the training and test splits. IID suites keep their evaluation examples out of training.

Table D.1: Benchmarks, metrics, and evaluation items.

Benchmark	Suite	Task	Metric	Eval. items
HotpotQA	IID	Multi-hop question answering	EM / F1	128
TriviaQA	IID	Factual question answering	EM / F1	128
AIME 2026	IID	Mathematical reasoning	Accuracy	30
MedXpertQA-Text	IID	Medical reasoning	Accuracy	128
ALFWorld	IID	Embodied task completion	Success rate	128
MBPP+	IID	Function-level code generation	Pass@1	128
MuSiQue	OOD	Multi-hop question answering	EM / F1	128
NQ-Open	OOD	Open-domain factual answers	EM / F1	128
MATH-Hard	OOD	Mathematical reasoning	Accuracy	128
GPQA Diamond	OOD	Scientific reasoning	Accuracy	128
HumanEval	OOD	Function-level code generation	Pass@1	128
WebShop	OOD	Web shopping in a simulated store	Success rate	128

Train and test identifiers are disjoint, including derived prompts and duplicated source instances. Dataset citations remain those in the main paper’s bibliography.

D.2 TASK ADAPTERS AND FACTUAL SCORING

QA adapters specify supplied versus retrieved evidence, retrieval access, answer extraction, and accepted reference aliases. Mathematical and expert-reasoning adapters specify the answer schema, parsing failures, and correctness matcher. Code-generation adapters execute submitted functions against their permitted test harness and record the test outcome. Embodied and conversational adapters expose their respective observation/action interfaces and terminal evaluators.

HumanEval and MBPP+ use separate submission formats, and the transfer from ALFWorld to WebShop changes the interaction interface as well as the task text. Adapter revisions, timeouts, and retry policies are given in the released code. Evaluator-only labels and hidden tests must not enter Agent prompts.

D.3 FROZEN HELD-OUT EVALUATION

The OOD protocol freezes the Collab-Director, Agent population, and training-derived configuration. No OOD evaluation instance is used for policy training, prompt selection, or hyperparameter tuning. Benchmark-specific adapters were fixed before outcome inspection. The six OOD comparisons include within-family changes and stronger domain or interface shifts.

For WebShop, the single-Executor restriction covers CollabFlow’s internal Agents only. The benchmark-controlled web server is external, and the Advisor is consulted between bounded execution phases with a fresh page snapshot. We record both policy-controlled browser actions and server-side state.

E BASELINES, ABLATIONS, AND COMPARISON CONTROLS

E.1 BASELINE INVENTORY

The main comparison covers direct Qwen3.5-9B inference, SFT, GRPO, AFlow, FlowSteer, the evolving orchestrator, SkillFlow, and SkillRL; Table E.1 lists the configuration of each.

Where interfaces permit, match backbone, tool access, token/model-call/tool-call limits, and interaction horizon. Report unmatched settings explicitly rather than calling all systems identical. Include

Table E.1: Baselines and their configurations.

Method	Role in comparison	Configuration
Direct Qwen3.5-9B	Shared-backbone reference	Checkpoint, tools, prompt, decoding and submission limits.
SFT	Task-level supervised baseline	Demonstration source, training split, trainable components, optimization budget.
GRPO	Task-level RL baseline	Policy target, reward, sample group, optimizer and rollout budget.
AFlow	Workflow-search baseline	Search budget, backend, candidate components and task adapter.
FlowSteer	Workflow-orchestration baseline	Policy/backend versions, action interface, reward and adapter changes.
Evolving orchestrator	Adaptive-organization baseline	Orchestrator policy, Agent pool, reward and adapter changes.
SkillFlow	Orchestration/Skill baseline	Policy, Skill lifecycle, workspace initialization and adapter changes.
SkillRL	Policy/Skill baseline	Training data, Skill initialization/evolution, backend and adapter changes.
CollabFlow	Proposed full specification	Released code and configuration; all components enabled.

output aggregation and formatting in the ledger. Search, sampling, and training budgets are separate quantities; one must not substitute for another when attributing a difference to the organization policy.

E.2 FIXED TEAMS AND RANDOM LEGAL GRAPHS

The structural controls are a single Agent, independent parallel Agents, a fixed researcher–solver–verifier chain, a stateful Executor–Advisor team, and randomly sampled legal graphs. A fixed topology is selected using training or validation data and reused within its task family; no per-test-instance oracle chooses the topology.

The random-graph proposal and stopping distribution are recorded, because they define the distribution over terminal graphs.

Independent Agent outputs require a specified aggregation mechanism. An ensemble that votes is different from running several Agents and selecting one unchanged answer. Fix the output mechanism when comparing communication, and charge all aggregation computation. Controls requiring three Agents report their search space.

E.3 COMPONENT ABLATIONS

Table E.2 defines the paradigm and ablation rows of Table 2; every row keeps the frozen executor, Agent population, and per-task budget of the full model.

Table E.2: Paradigm and ablation rows of Table 2.

Row	Intervention
Single agent with tools	One tool-using Agent answers alone.
Parallel vote	Independent Agents answer; a majority vote selects the output.
Fixed chain	A fixed researcher–solver–verifier chain with verbatim messages.
Fully connected debate	Every Agent receives every other Agent’s full output each round.
Evolving orchestrator	The orchestrator of Dang et al. (2026) activates Agents in sequence.
– Director training	The untrained Collab-Director builds the teams.
– Recursion	The director is trained once and not retrained on later rounds.
Fixed team	The best hand-built team per task family, chosen on training data.
Random legal graph	Teams are sampled uniformly from the legal action set.
– Communication	Agents answer without intermediate messages; the output rule is kept.
– Evidence gate	Eq. (7) is replaced by verbatim exchange.
All ONEWAY	Every edge runs the ONEWAY protocol.
Always revise	Every update takes the revision branch of Eq. (7).

F EVALUATION METRICS

The definitions below make the manuscript’s reporting conventions explicit. Use the chosen benchmark evaluator and record any deviation, especially answer normalization, multiple references, partial task credit, and simulator-specific success rules.

F.1 TASK PERFORMANCE AND MAIN-TABLE AVERAGES

For QA item i with prediction $\hat{y}_i$ and accepted reference set $\mathcal{Y}_i^*$, normalized exact match is

$$\text{EM}_i = \max_{y \in \mathcal{Y}_i^*} \mathbb{I}[\nu(\hat{y}_i) = \nu(y)], \quad (\text{F.1})$$

where ν is the benchmark’s declared normalizer.

Token overlap uses multiset counts, not a set that discards repeated tokens. If $c(a, b)$ is the sum of minimum token counts between normalized answers a and b , token F1 for a nonempty combined token length is $2c(a, b)/(|a| + |b|)$, maximized over accepted references. Empty-answer behavior follows the frozen evaluator.

Accuracy, embodied and web success, and a single-attempt pass score are means of their declared binary item-success indicators, multiplied by 100 when reported as percentages. Keep continuous QA F1 separate from a binary success flag used for reliability counters and success-normalized cost.

For each suite, average EM and F1 separately over its two QA benchmarks; Acc./Pass is the unweighted mean of the primary scores of its four remaining benchmarks. QA EM and QA F1 stay separate from the other columns. The main table’s Δ remains the absolute difference from direct Qwen3.5-9B in the corresponding metric, expressed in percentage points.

F.2 RESOURCES, FAILURES, AND VARIABILITY

Count total model input and output tokens for Collab-Director construction, Executors, Advisors, revisions, aggregation, and formatting. Record model calls, tool calls, wall-clock latency, invalid attempts, and caching conventions. Evaluator and external simulator computation are recorded separately, with total service cost reported when relevant. Training additionally records rollout collection time, optimizer time, and peak memory.

For a declared cost unit c_i , cost per successful task is $\sum_i c_i / \sum_i q_i$, reported as undefined when no task succeeds. Invalid-action rate uses attempted environment actions as its denominator; repetition, conflicting modification, and recovery-step rules use fixed event definitions and windows.

Compute each suite score within each independent run before summarizing across runs. Report the arithmetic mean and sample standard deviation using denominator $N_{\text{run}} - 1$; a single run does not have an estimated run-to-run standard deviation. Separate training seeds from evaluation samples of one checkpoint. Use paired tasks and declared resource limits for comparative analysis. Any confidence interval or significance procedure must state its resampling unit and be added only after the corresponding analysis is performed.

G IMPLEMENTATION AND REPRODUCIBILITY DETAILS

G.1 CONFIGURATION

Qwen3.5-9B serves as both the Collab-Director and the frozen executor, and every method shares one Agent population, tool set and per-task budget. The released code contains the full configuration, prompts, seeds and checkpoints of every reported run.

G.2 IMPLEMENTATION CHECKS

For the communication claims, the evaluated configuration enables the conditional update of Eq. (7), the revision token cap, the no-tool constraint on revision and Advisor calls, and the single-writer checks, and the run logs record each call. Advisor usage is charged against the single total budget.

For the terminal distribution, the implementation scores every legal action, fixes the sufficient state and canonical terminal, normalizes b_ϕ over the construction orders of each team, and stores scores equal to the sampling log probabilities, so the premises T1–T3 of Proposition 3 hold by construction.

For stateful execution, the permission gate and commit mechanism are documented and tested with Advisor attempts to call a mutating operation and with attempts to overlap internal state-changing tool effects.

G.3 RELEASE ARTIFACTS

The anonymous repository (<https://anonymous.4open.science/r/CollabFlow-631E>) releases the training and evaluation code, the trained Collab-Director weights, the training and test splits of all twelve datasets, and the scripts that regenerate every table and figure.

H ADDITIONAL ANALYSES AND COMPLETE REPORTING TABLES

This appendix lists the per-benchmark scores behind Figure 4 and the token costs behind Figure 5f.

H.1 PER-BENCHMARK SCORES WITH OTHER EXECUTORS

Table H.1 lists the per-benchmark scores behind Figure 4, together with DeepSeek V4, which was evaluated on the six IID benchmarks only and is therefore absent from the figure.

(a) In-Distribution (IID) benchmarks

Executor	HotpotQA	TriviaQA	AIME 2026	MedXpertQA	ALFWorld	MBPP+	Avg. (IID)	
	EM	EM	Acc.	Acc.	SR	Pass@1	Direct	CollabFlow
GPT-5.6-Luna	55.47 (+33.59)	88.28 (+35.16)	93.33 (+16.67)	100.00 (+62.50)	89.06 (+16.41)	92.97 (+5.47)	58.22	86.52 (+28.30)
DeepSeek V4	58.59 (+28.91)	90.63 (+14.84)	83.33 (+33.33)	100.00 (+21.09)	81.25 (+65.63)	89.84 (+3.91)	55.99	83.94 (+27.95)
GLM	65.63 (+38.28)	87.50 (+53.13)	56.67 (+13.33)	100.00 (+10.16)	96.09 (+7.03)	84.38 (+19.53)	58.13	81.71 (+23.58)
DeepSeek V4.1	64.06 (+24.22)	89.06 (+23.44)	83.33 (+13.33)	100.00 (+18.75)	94.53 (+36.72)	88.28 (+7.03)	65.96	86.55 (+20.58)
Grok-4.5	69.53 (+10.94)	91.41 (+5.47)	96.67 (+10.00)	100.00 (+9.38)	97.66 (+13.28)	90.63 (+21.09)	79.29	90.98 (+11.69)
MiniMax M2.7	61.72 (+9.38)	89.84 (+7.03)	70.00 (+20.00)	95.31 (+22.66)	96.09 (+10.16)	89.06 (+14.06)	69.79	83.67 (+13.88)
Claude Haiku 4.5	66.41 (+6.25)	89.84 (+3.91)	73.33 (+20.00)	100.00 (+11.72)	93.75 (+3.91)	88.28 (+5.47)	76.73	85.27 (+8.54)

(b) Out-of-Distribution (OOD) benchmarks

Executor	MuSiQue	NQ-Open	MATH-Hard	GPQA-D	HumanEval	WebShop	Avg. (OOD)	
	EM	EM	Acc.	Acc.	Pass@1	SR	Direct	CollabFlow
GPT-5.6-Luna	78.91 (+24.22)	77.34 (+14.06)	95.31 (+5.47)	85.94 (+10.94)	96.88 (+7.81)	82.81 (+25.78)	71.48	86.20 (+14.71)
GLM	82.03 (+13.28)	72.66 (+10.94)	96.88 (+10.94)	81.25 (+14.06)	97.66 (+5.47)	72.66 (+5.47)	73.83	83.85 (+10.03)
DeepSeek V4.1	85.94 (+14.06)	64.06 (+11.72)	93.75 (+14.06)	81.25 (+17.19)	95.31 (+2.34)	77.34 (+8.59)	71.61	82.94 (+11.33)
Grok-4.5	81.25 (+20.31)	71.09 (+11.72)	96.09 (+7.03)	72.66 (+14.84)	93.75 (+7.03)	81.25 (+34.38)	66.80	82.68 (+15.89)
MiniMax M2.7	71.09 (+10.16)	77.34 (+9.38)	95.31 (+2.34)	85.16 (+17.19)	96.88 (+4.69)	57.81 (+7.81)	72.01	80.60 (+8.59)
Claude Haiku 4.5	78.13 (+32.03)	72.66 (+13.28)	96.88 (+6.25)	81.25 (+24.22)	95.31 (+0.00)	77.34 (+14.06)	68.62	83.59 (+14.97)

Table H.1: CollabFlow with other frozen executors. Each cell gives the score with CollabFlow and, in pink, its gain over direct inference by the same executor; averages cover each part’s six benchmarks.

H.2 TOKEN COST PER ITEM

Table H.2 lists the tokens used per item by CollabFlow and the four comparison systems, counting the input and output tokens of every model call in an episode.

I LIMITATIONS

CollabFlow is a bounded form of recursive self-improvement: its answer evaluator, executor, Agent population, and update rules stay fixed across rounds, and only the director (with its backward kernel

Method	Mean per item			Spread of the total			CV ↓
	Total ↓	Input ↓	Output ↓	P90 ↓	Max ↓	Std ↓	
Single agent	12,010.0	10,889.8	1,120.2	27,507.4	39,279	13,892.5	1.157
Fixed team	10,404.4	9,119.6	1,284.8	13,091.0	14,437	2,373.8	0.228
Ensemble	24,081.8	20,894.2	3,187.6	37,974.8	47,942	12,557.5	0.521
SkillFlow	10,451.6	9,396.0	1,055.6	16,639.4	18,039	5,063.7	0.484
CollabFlow	9,676.6	8,645.8	1,030.8	11,490.0	11,986	1,903.2	0.197

Table H.2: Tokens per item. Single agent: one tool-using Agent; Fixed team: the original hand-built team; Ensemble: independent Agents whose answers are combined. CV is the standard deviation divided by the mean total.

and normalizer), the skill library, and the reliability records change. The representation is limited by its fixed Agent population, protocols, output modes, and resource caps. Construction coverage is relative to this admissible space, and does not imply a capable Agent exists for every task. Additional organization choices can also increase sampling cost without improving task performance.

The single-writer design prevents conflicts between different internal writers under enforced permissions and ordered commits, but does not guarantee correct plans, fresh observations, safe action semantics, or consistency with external participants. Diagnostic traces are observational and may associate a message with an improvement that also depends on extra computation or other changes.

The conditional update compares tool and retrieval support with fixed rules, so authentic but incomplete or misleading evidence can still favor an incorrect candidate. Fabricated event records are excluded under the authentication contract; accepting them constitutes a violation of that contract. Reliability estimates from sparse or nonstationary executions need not be calibrated, and finite training does not ensure the exact terminal distribution proved under fixed conditions.

Runtime likelihood surrogates, incomplete budget accounting, and unimplemented adapters or extensions limit which claims can be supported by a particular evaluated system.

J FUTURE WORK

Further work can investigate controlled message interventions that distinguish peer information from additional reasoning, while retaining explicit resource and permission accounting.

Another direction is open-ended recursion: learning the update rule itself, or parts of the evaluator, jointly with the team, rather than fixing them in advance.

Larger Agent populations, better-calibrated reliability estimates, and mid-episode advice in more environments also merit evaluation under comparable total costs.

K APPLICABILITY AND BROADER IMPACT

The framework targets tasks for which candidate Agents, permissible communication, execution interfaces, and factual evaluators can be specified. Its organization/execution separation permits explicit inspection of selected roles, messages, and resource use.

Potential applications nevertheless require domain-specific validation: benchmark success, including on medical reasoning questions, does not authorize unsupervised clinical decisions or unsafe execution in a real environment.

Repository and tool access should use limited permissions, isolated execution, and appropriate human approval for consequential actions. Trace releases should respect confidentiality and avoid distributing sample-specific sensitive information.