I-FLOW: INTENT-REVEALING FLOW OPTIMIZATION FOR MULTI-TURN LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn reasoning enables large language models (LLMs) to tackle complex problems through structured reasoning across turns. However, we uncover a critical failure mode of final-reward reinforcement learning: optimizing only for final correctness can collapse the intended hierarchy and drive diverse reasoning toward short and repetitive patterns. To address this limitation, we propose I-Flow, which structures each reasoning turn around a turn-level *intent* and introduces a Generative Flow Network (GFlowNet) optimization for the resulting intent hierarchy. Our bootstrap-free subtrajectory objective propagates terminal supervision across intent sequences, encouraging diverse high-reward reasoning structures without step-wise bootstrapping. We further replay successful intent structures to preserve effective reasoning patterns throughout training. Experiments on mathematical reasoning and code generation demonstrate consistent performance gains while effectively mitigating hierarchy collapse and reasoning simplification.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable progress across complex reasoning tasks through chain-of-thought (CoT) reasoning (Wei et al., 2022; Wang et al., 2023). More recently, reinforcement learning with verifiable rewards (RLVR) has further advanced LLM reasoning by optimizing reasoning trajectories based on verifiable outcomes (Shao et al., 2024; Guo et al., 2025). However, conventional single-turn reasoning generates an entire reasoning trajectory in a single process, providing limited opportunities to assess intermediate progress and adapt how reasoning should proceed. This has motivated multi-turn reasoning, where subsequent decisions can build on accumulated reasoning context.

Multi-turn reasoning decomposes problem solving across multiple rounds, allowing subsequent decisions to build on preceding outputs (Dua et al., 2022; Chen et al., 2023). Early approaches primarily focused on self-correction and iterative refinement (Madaan et al., 2023; Ma et al., 2025), while recent hierarchical approaches further separate high-level guidance from low-level reasoning at each turn (Wan et al., 2025; Liu et al., 2026). This hierarchy enables the model to decide how to proceed at the high level and execute the corresponding reasoning at the low level. However, as illustrated on the left of Figure 1, we observe two critical issues under final-reward optimization: *hierarchy collapse* and *reasoning simplification*. First, the intended hierarchy can collapse as high-level guidance directly produces solutions or becomes entangled with low-level reasoning. Second, initially diverse reasoning trajectories can converge to a small set of short and simple patterns during training, even as task accuracy improves.

We attribute these phenomena to final-reward optimization that does not explicitly account for reasoning structure across turns. As long as the final answer is correct, the policy has little incentive to preserve the intended hierarchy or diverse reasoning patterns. To address this problem, we propose **Intent Flow (I-Flow)**, which introduces a turn-level *intent* before each high-level guidance and develops a Generative Flow Network (GFlowNet) optimization tailored to the resulting intent hierarchy. As illustrated on the right of Figure 1, explicit intents make high-level reasoning decisions identifiable, while our flow optimization learns diverse successful intent sequences in proportion to their rewards. We further introduce *Intent Structure Replay*, which reuses successful intent structures to preserve effective reasoning patterns as the policy evolves. Experiments across mathematical reasoning and code generation demonstrate consistent performance improvements, with further analyses showing that I-Flow effectively mitigates hierarchy collapse and reasoning simplification.

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

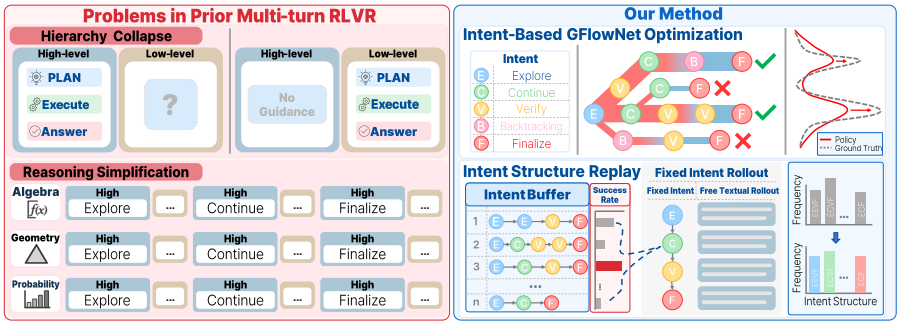


Figure 1: Overview of I-Flow. Left: Standard RLVR leads to hierarchy collapse and reasoning simplification in hierarchical multi-turn reasoning. Right: I-Flow addresses these issues with Intent-based GFlowNet Optimization and Intent Structure Replay, where each intent specifies how high-level guidance should proceed at each turn.

2 Related Work

LLM Reasoning. LLM reasoning has progressed from eliciting explicit intermediate steps to exploring multiple reasoning paths (Wei et al., 2022; Wang et al., 2023). Beyond a single reasoning trace, search-based approaches explore and evaluate alternative intermediate thoughts (Yao et al., 2023a; Besta et al., 2024; Zhou et al., 2024a), while iterative and interactive approaches refine subsequent decisions using observations or self-generated feedback (Yao et al., 2023b; Shinn et al., 2023; Madaan et al., 2023). More recently, these reasoning capabilities have been incorporated into multi-agent systems (Bae et al., 2026; Hwang et al., 2026) and hierarchical frameworks to combine complementary perspectives or separate strategic decisions from execution (Du et al., 2024; Liang et al., 2024; Zhou et al., 2024b; Wan et al., 2025).

Multi-turn and Hierarchical Reasoning. Multi-turn approaches allow subsequent decisions to build on preceding outputs. Early methods focus on iterative refinement and self-correction (Madaan et al., 2023; Kumar et al., 2025), while others use multiple turns for information seeking, search, or interaction with external environments (Jin et al., 2025; Qi et al., 2025). More recently, hierarchical methods explicitly separate high-level guidance from low-level reasoning, enabling strategic decisions to guide execution across turns (Zhou et al., 2024b; Wan et al., 2025; Liu et al., 2026; Peng et al., 2026). These approaches provide structured control over multi-turn reasoning by repeatedly coordinating high-level decisions with their corresponding reasoning processes. Building on this hierarchical formulation, our work explicitly represents high-level decisions with turn-level intents and focuses on maintaining the intended hierarchy throughout optimization.

Diversity Preservation in Reinforcement Learning. Various approaches in RL maintain behavioral diversity during optimization. Entropy-based methods encourage stochasticity and exploration through high-entropy action distributions (Haarnoja et al., 2018; Zhao et al., 2019; Han & Sung, 2021a;b), while policy-constrained methods regularize updates against a reference or previous policy to limit distributional shifts (Schulman et al., 2015; Kumar et al., 2020; Yeom et al., 2024; Chae et al., 2026). These perspectives are related, as entropy regularization can be viewed as regularization toward a uniform reference policy. GFlowNets instead learn reward-proportional distributions over trajectories, naturally assigning probability to diverse high-reward solutions (Bengio et al., 2021; 2023; Zhu et al., 2026). In RL for LLMs, related approaches encourage exploration and diversity through entropy- or KL-based regularization and sampling strategies, primarily at the token or response level (Yu et al., 2025; Wang et al., 2025; Yao et al., 2025). Our work instead targets diversity at the level of multi-turn reasoning structures by optimizing sequences of turn-level intents.

3 Preliminaries

3.1 Hierarchical Multi-Turn LLM and RLVR

Multi-turn LLMs solve a problem through context-dependent generations, with each turn conditioned on preceding outputs. Hierarchical approaches further separate high-level guidance from low-level execution (Zhou et al., 2024b; Wan et al., 2025). Given a problem $\mathbf{x} \sim \mathcal{D}$, we consider a

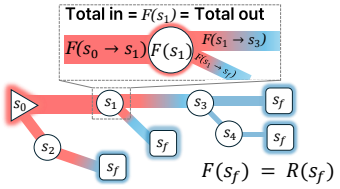


Figure 2: Illustration of flow propagation in GFlowNets.

Table 1 | Balance objectives in GFlowNets. $s_{i:j} := (s_i, \ldots, s_j)$ denotes the subtrajectory from s_i to s_j.

Objective	Balance Condition	Granularity	Intermediate Constraints	Terminal Anchoring
DB	$F(s_t)P_F(s_{t+1} \mid s_t) = F(s_{t+1})P_B(s_t \mid s_{t+1})$	Transition	✓	×
TB	$F(s_0)P_F(s_{0:f}) = R(s_f)P_B(s_{0:f})$	Trajectory	×	✓
SubTB	$F(s_i)P_F(s_{i:j}) = F(s_j)P_B(s_{i:j})$	Subtrajectory	✓	×
Ours	$F(s_t)P_F(s_{t:f}) = R(s_f)P_B(s_{t:f})$	Subtrajectory	✓	✓

hierarchical LLM policy π_θ with high- and low-level components π_θ^h and π_θ^l, configured by system prompts I_h and I_l. At each turn $1 \le t \le T$, π_θ^h generates guidance h_t, which π_θ^l executes to produce reasoning output a_t. Defining $c_t = (\mathbf{x}, \mathbf{h}_{<t}, \mathbf{a}_{<t})$ with $c_1 = \mathbf{x}$, the hierarchical generation factorizes as

$$\pi_\theta(h_t, a_t \mid c_t; I_h, I_l) = \pi_\theta^h(h_t \mid I_h, c_t)\pi_\theta^l(a_t \mid I_l, h_t, c_t). \tag{1}$$

We denote a partial trajectory by $\boldsymbol{\tau}_{t_1:t_2} := (h_{t_1}, a_{t_1}, \ldots, h_{t_2}, a_{t_2})$ and the complete trajectory by $\boldsymbol{\tau} := \boldsymbol{\tau}_{1:T}$.

For problems with verifiable outcomes, LLM reasoning can be optimized using reinforcement learning with verifiable rewards (RLVR). Let $r(\mathbf{x}, \boldsymbol{\tau})$ denote the terminal reward and π_{ref} a reference policy. With KL coefficient $\beta > 0$, the KL-regularized RLVR objective is

$$\mathcal{J}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, \boldsymbol{\tau}\sim\pi_\theta}\left[r(\mathbf{x}, \boldsymbol{\tau}) - \beta \log \frac{\pi_\theta(\boldsymbol{\tau} \mid \mathbf{x}; I_h, I_l)}{\pi_{\text{ref}}(\boldsymbol{\tau} \mid \mathbf{x}; I_h, I_l)}\right]. \tag{2}$$

For each $\mathbf{x}$, maximizing Eq. (2) over trajectory distributions yields

$$\pi^*(\boldsymbol{\tau} \mid \mathbf{x}; I_h, I_l) = \frac{\pi_{\text{ref}}(\boldsymbol{\tau} \mid \mathbf{x}; I_h, I_l)\exp\left(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\right)}{\mathbb{E}_{\boldsymbol{\tau}'\sim\pi_{\text{ref}}}\left[\exp\left(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}')\right)\right]}, \tag{3}$$

which exponentially reweights trajectories by their rewards. While Eq. (3) defines the optimal trajectory distribution, finite-sample on-policy optimization relies only on terminal rewards and provides limited guidance for intermediate reasoning. This motivates preserving the same target while propagating terminal supervision to intermediate decisions.

3.2 Generative Flow Networks and Balance Objectives

Generative Flow Networks (GFlowNets) provide a natural framework for our setting by learning a policy whose terminal-state distribution is proportional to a positive reward, i.e., $P(s_f) \propto R(s_f)$, while enforcing flow consistency over intermediate states and trajectories (Bengio et al., 2021). In our RLVR setting, a terminal state s_f corresponds to a complete reasoning trajectory $\boldsymbol{\tau}$. Choosing

$$R(s_f) \propto \pi_{\text{ref}}(\boldsymbol{\tau} \mid \mathbf{x}; I_h, I_l)\exp\left(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\right) \tag{4}$$

yields the KL-regularized optimum in Eq. (3). Thus, GFlowNets preserve the same reward-weighted trajectory target while enabling terminal supervision over intermediate reasoning states.

Let $F(s_t \to s_{t+1})$ denote the transition flow and $F(s_t)$ the total flow through s_t. Figure 2 illustrates flow propagation in a GFlowNet, where flow is conserved at each intermediate state and matches the reward at a terminal state:

$$F(s_t) = \sum_{s'\to s_t} F(s' \to s_t) = \sum_{s_t \to s'} F(s_t \to s'), \qquad F(s_f) = R(s_f). \tag{5}$$

To characterize how this flow propagates between states, GFlowNets define two policies. The forward policy $P_F(s_{t+1} \mid s_t) := F(s_t \to s_{t+1})/F(s_t)$ determines how the flow at s_t is distributed among possible next states, while the backward policy $P_B(s_t \mid s_{t+1}) := F(s_t \to s_{t+1})/F(s_{t+1})$ describes how the flow arriving at s_{t+1} is attributed to its possible predecessors.

Different GFlowNet balance objectives enforce Eq. (5) at different temporal granularities, as summarized in Table 1 (Whitammer et al., 2022; Bengio et al., 2023; Madan et al., 2023). Detailed Balance (DB) imposes a local consistency constraint at every transition. This provides fine-grained intermediate supervision but relies on step-wise bootstrapping from learned nonterminal flows. In contrast,

Trajectory Balance (TB) directly relates the probability of a complete trajectory to its terminal reward without bootstrapping from intermediate targets, but provides only a single root-to-terminal constraint. Subtrajectory Balance (SubTB) interpolates between them by imposing balance over partial trajectories $s_{i:j}$. It provides intermediate supervision beyond TB but still relies on learned endpoint flows for nonterminal states.

This distinction is particularly important in RLVR, where only a terminal binary reward $r(\mathbf{x}, \boldsymbol{\tau}) \in \{0, 1\}$ is available. TB avoids bootstrapping but provides no intermediate supervision, limiting its ability to propagate the terminal signal to intermediate reasoning states. In contrast, DB and SubTB provide intermediate supervision but require bootstrapping from learned nonterminal flows. Our goal is therefore to combine *direct terminal anchoring* from TB with *intermediate supervision* from SubTB. To this end, we develop a bootstrap-free SubTB-style objective that provides intermediate supervision while directly anchoring each constraint to the terminal outcome, as shown in Table 1.

4 METHODOLOGY

4.1 INTENT-REVEALING MULTI-TURN REASONING AND STATE FORMULATION

In hierarchical multi-turn reasoning, the high-level guidance h_t determines how reasoning should proceed from the current context. However, since h_t is expressed as free-form text, its underlying high-level decision can be difficult to identify consistently across trajectories. We therefore make this decision explicit through a discrete turn-level *intent* $u_t \in \mathcal{U}$, which specifies *how to proceed next* before generating the high-level guidance. The guidance h_t provides a problem-specific realization of u_t, and the low-level output a_t executes this guidance. For mathematical reasoning and code generation, we use $\mathcal{U} = \{\text{EXP}, \text{VER}, \text{BCK}, \text{CON}, \text{IDL}, \text{FIN}\}$, covering common reasoning operations (Figure 3): exploring alternatives, verifying progress, backtracking, resuming previous steps, making routine progress, and finalizing the answer, respectively. Figure 3 illustrates representative intent types and their corresponding reasoning examples.

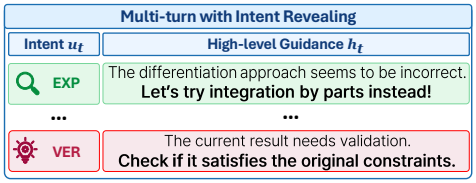


Figure 3: Representative intent types and their corresponding reasoning examples.

At each turn, we use an intent policy π_θ^I to select a turn-level intent u_t from the pre-turn context $c_t := (\mathbf{x}, \mathbf{u}_{<t}, \mathbf{h}_{<t}, \mathbf{a}_{<t})$. Given the selected intent, the model generates the high-level guidance and low-level reasoning (h_t, a_t). The resulting turn-level generation factorizes as

$$\pi_\theta(u_t, h_t, a_t \mid c_t) = \pi_\theta^I(u_t \mid c_t)\pi_\theta(h_t, a_t \mid c_t, u_t). \tag{6}$$

where the second term denotes the probability of generating (h_t, a_t) conditioned on the selected intent. Both terms share the same LLM parameters θ, and we omit the fixed role prompts I_h and I_l for simplicity. Under this formulation, the complete trajectory is redefined as $\boldsymbol{\tau} = (u_1, h_1, a_1, \ldots, u_T, h_T, a_T)$, whose intent sequence $\sigma(\boldsymbol{\tau}) = (u_1, \ldots, u_T)$ defines the *intent structure*. This discrete representation explicitly captures the sequence of high-level reasoning decisions, allowing intent structures to be compared, tracked, and reused across rollouts.

To apply GFlowNets to our intent-revealing structure, we define intermediate states as *post-intent states* immediately after the turn-level intent is selected:

$$s_0 = \mathbf{x}, \qquad s_t = (c_t, u_t),\ 1 \le t \le T, \qquad s_{T+1} = c_{T+1}. \tag{7}$$

Unlike the context c_t, the state s_t additionally contains the selected intent u_t and therefore identifies the reasoning operation to be executed at turn t. Accordingly, each transition completes the current turn and, except for the terminal transition, selects the next intent:

$$P_\theta(s_{t+1} \mid s_t) = \begin{cases} \pi_\theta^I(u_1 \mid \mathbf{x}), & t = 0, \\ \pi_\theta(h_t, a_t \mid c_t, u_t)\pi_\theta^I(u_{t+1} \mid c_{t+1}), & 1 \le t < T, \\ \pi_\theta(h_T, a_T \mid c_T, u_T), & t = T. \end{cases} \tag{8}$$

Following prior GFlowNet-based LLM optimization such as FlowRL (Zhu et al., 2026), we define the forward probability of a state sequence as $P_F(s_{i:j}) = \prod_{t=i}^{j-1} P_\theta(s_{t+1} \mid s_t)$, which corresponds

to the LLM sequence probability over the associated generations. Since each state contains the complete preceding reasoning history, every valid state has a unique predecessor in the autoregressive generation process. We therefore set the backward transition probability to $P_B(s_t \mid s_{t+1}) = 1$, and hence $P_B(s_{i:j}) = 1$ for any valid subtrajectory.

4.2 Intent-Based Flow Optimization

As discussed in Section 3.2, we aim to provide intermediate supervision while retaining direct terminal anchoring without bootstrapping. To this end, we introduce *Bootstrap-Free Subtrajectory Balance (BF-SubTB)*, which specializes the SubTB condition in Table 1 to subtrajectories from each post-intent state s_t to the terminal state s_{T+1}:

$$F(s_t)P_F(s_{t:T+1}) = F(s_{T+1})P_B(s_{t:T+1}), \quad \forall t. \tag{9}$$

To match the RLVR target in Eq. (4), we set the terminal flow as $F(s_{T+1}) = \pi_{\text{ref}}(\boldsymbol{\tau} \mid \mathbf{x}) \exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big)$. For intermediate states, we parameterize the flow as $F(s_t) = \pi_{\text{ref}}(s_t \mid \mathbf{x}) F_\phi(s_t)$, where $\pi_{\text{ref}}(s_t \mid \mathbf{x}) = \pi_{\text{ref}}(\boldsymbol{\tau} \mid \mathbf{x}) / \pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t)$ denotes the probability of reaching s_t under the reference policy and $F_\phi(s_t)$ is a learnable continuation flow. Substituting these flow definitions together with $P_F(s_{t:T+1}) = \pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t)$ and $P_B(s_{t:T+1}) = 1$ into Eq. (9) yields

$$R_t(\boldsymbol{\tau}) = \log F_\phi(s_t) + \log \frac{\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t)}{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t)} - \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}), \quad \forall t, \tag{10}$$

where $\boldsymbol{\tau}_{t:}$ denotes the remaining generation from s_t to s_{T+1}, with $\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \prod_{k=t}^{T} P_\theta(s_{k+1} \mid s_k)$. BF-SubTB enforces $R_t(\boldsymbol{\tau}) = 0$ at each post-intent state, directly matching its continuation flow to the verified terminal outcome. This provides intent-aligned intermediate supervision without learned nonterminal endpoint flows. We refer to this construction as **Intent Flow (I-Flow)**, combining the intermediate supervision of SubTB with the direct terminal anchoring of TB.

Intent-Conditioned Flow Characterization. We next characterize the solution induced by the BF-SubTB conditions and its connection to intent optimization. For each post-intent state $s_t = (c_t, u_t)$, define the intent-conditioned continuation flow F^* and the optimal (KL-regularized) continuation policy π^* derived from Eq. (3) as

$$F^*(s_t) = F^*(c_t, u_t) = \mathbb{E}_{\boldsymbol{\tau}_{t:} \sim \pi_{\text{ref}}(\cdot \mid c_t, u_t)} \left[\exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big) \right], \tag{11}$$

$$\pi^*(\boldsymbol{\tau}_{t:} \mid s_t) = \pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t) \exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big) / F^*(s_t). \tag{12}$$

Here, F^* represents the reward-weighted desirability of future reasoning after selecting intent u_t from context c_t. Moreover, marginalizing the KL-regularized trajectory target over subsequent generations gives

$$\pi^{I*}(u_t \mid c_t) \propto \pi^I_{\text{ref}}(u_t \mid c_t) F^*(c_t, u_t), \tag{13}$$

showing that $F^*(c_t, u_t)$ reweights the reference intent policy according to the quality of its future continuations. For binary rewards, we prove that $F^*(c_t, u_t)$ is strictly increasing in the optimal continuation success probability (see Appendix B.1). Given this, the following result establishes that the BF-SubTB conditions **identify both π^* and F^* at every state**, thereby preserving the original KL-regularized RLVR target at the complete-trajectory level.

Proposition 1 (Policy and Flow Identification)**.** *Assume a finite horizon, common positive support between π_θ and π_{ref}, and that π^* and F^* are realizable under the parameterization. Then, all BF-SubTB conditions hold if and only if*

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \pi^*(\boldsymbol{\tau}_{t:} \mid s_t), \qquad F_\phi(s_t) = F^*(s_t), \qquad 0 \le t \le T, \tag{14}$$

for every state s_t. In particular, since $s_0 = \mathbf{x}$, $\pi_\theta(\boldsymbol{\tau} \mid \mathbf{x}) = \pi^*(\boldsymbol{\tau} \mid \mathbf{x})$.

Proposition 1 establishes that the learned intermediate flows are not arbitrary auxiliary variables but correspond to how promising each selected intent is for future reasoning under the same global RLVR target. The proof is provided in Appendix B.2.

I-Flow Objective. To achieve BF-SubTB conditions during training, we minimize their residuals at all post-intent states together with the root-to-terminal TB constraint:

$$\mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}) = \underbrace{R_0(\boldsymbol{\tau})^2}_{\text{root-to-terminal TB}} + \underbrace{\frac{\alpha}{T} \sum_{t=1}^{T} R_t(\boldsymbol{\tau})^2}_{\text{BF-SubTB at post-intent boundaries}}, \qquad \alpha > 0. \tag{15}$$

Here, α controls the strength of intent-boundary supervision, while $1/T$ normalizes its contribution across reasoning turns. These terminal-anchored intermediate constraints provide finer-grained supervision while preserving the same target distribution. We formalize this property by bounding local flow-balance errors across individual transitions.

Proposition 2 (Local Flow-Balance Control)**.** *Consider any completed trajectory* $\boldsymbol{\tau} = (s_0, \ldots, s_{T+1})$ *with* $T \geq 1$*, positive policy and reference transition probabilities, and positive flows. Define the local reference-relative balance residual*

$$\Delta_t = \log F_\phi(s_t) + \log \frac{P_\theta(s_{t+1} \mid s_t)}{P_{\text{ref}}(s_{t+1} \mid s_t)} - \log F_\phi(s_{t+1}), \qquad 0 \leq t \leq T, \tag{16}$$

with terminal boundary $\log F_\phi(s_{T+1}) = \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})$*. Then the total local flow-balance error is bounded by*

$$\sum_{t=0}^{T} \Delta_t^2 \leq \frac{4}{\min\{1, \alpha/T\}} \mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}). \tag{17}$$

Proposition 2 shows that I-Flow controls local inconsistencies that can remain hidden under root-to-terminal TB. Root-to-terminal TB constrains only $R_0 = \sum_{t=0}^{T} \Delta_t$, allowing positive and negative local errors to cancel even when individual transitions remain imbalanced. In contrast, Eq. (17) ensures that, for fixed T and $\alpha > 0$, $\mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}) \to 0$ implies $\Delta_t \to 0$ for every transition along the trajectory. Since $\beta \log F_\phi(s_t)$ can be interpreted as a soft continuation value, this provides a precise sense in which BF-SubTB offers finer intent-aligned supervision than root-to-terminal TB alone. The proof is provided in Appendix B.3.

4.3 Intent Structure Replay

As illustrated in Figure 1, although I-Flow preserves a reward-proportional distribution over diverse intent structures, previously successful structures may become less frequent as the policy evolves. To preserve them, we introduce *Intent Structure Replay (ISR)*, which maintains successful intent sequences throughout training. Crucially, our intent-revealing formulation makes this possible by explicitly representing high-level reasoning decisions as discrete turn-level intents, allowing successful structures to be identified, stored, and later enforced. For each question, we maintain a replay buffer

$$\mathcal{B}(\mathbf{x}) = \left\{ \left(\sigma, n_{\text{trial}}(\sigma), n_{\text{succ}}(\sigma)\right) \right\}, \tag{18}$$

where $n_{\text{trial}}(\sigma)$ and $n_{\text{succ}}(\sigma)$ count the executions and verified successes of intent structure σ. Since only intent sequences are stored rather than full reasoning trajectories, this incurs little memory overhead. During training, a fraction ε of online trajectories use intent structures sampled from the buffer in proportion to their historical success rates $n_{\text{succ}}(\sigma)/n_{\text{trial}}(\sigma)$ with smoothing, while the remaining trajectories are sampled freely from the current policy. For these structure-guided trajectories, each turn-level intent u_t is fixed to the selected structure, while h_t and a_t are freshly generated by the current policy. After each rollout, the trial and success statistics of the executed structure are updated. Detailed algorithms and implementation details are provided in Appendix C.3.

5 Experiments

5.1 Experimental Setup

Model Initialization. To train I-Flow, we use Qwen2.5-7B (Qwen et al., 2025) as the backbone and first perform supervised fine-tuning (SFT) on 817 LIMO solutions (Ye et al., 2025). We convert each solution into a dialogue between a high-level controller and a low-level reasoner, and annotate each high-level turn with one of the intent categories in $\mathcal{U} = \{\text{EXP}, \text{VER}, \text{BCK}, \text{CON}, \text{IDL}, \text{FIN}\}$ using GPT-4o. We denote the resulting SFT checkpoint as Qwen2.5-7B-Intent. Starting from this checkpoint, we then perform RL on the MATH training split (Hendrycks et al., 2021).

Baselines. We compare I-Flow with several single- and multi-turn RLVR methods. Single-turn baselines include **GRPO** (Shao et al., 2024), which optimizes terminal rewards using group-relative advantages; **MRP** (Wan et al., 2025), which introduces a meta-thinking stage before detailed reasoning; **FlowRL** (Zhu et al., 2026), which uses TB-based GFlowNets to preserve diverse high-reward

Table 2 | Comparison results on mathematical reasoning benchmarks. A@16 and P@16 denote Avg@16 and Pass@16, respectively.

Method	GSM8K		MATH500		Gaokao23		Minerva		Olympiad		AIME25		AMC23	
	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16
Single-turn														
Qwen2.5-7B	82.8	92.2	46.4	89.0	44.0	78.9	19.7	51.8	21.6	54.2	2.3	16.7	28.8	72.5
GRPO	89.2	96.1	67.1	87.0	57.0	74.8	34.2	51.1	36.7	61.2	6.7	23.3	57.0	82.5
MRP	90.0	96.7	69.7	89.8	59.6	77.4	33.6	51.5	38.2	64.0	7.5	23.3	57.0	80.0
FlowRL	90.6	96.8	71.8	91.2	61.4	77.4	35.0	50.4	39.8	63.7	7.1	23.3	53.1	87.5
SIREN	91.3	97.1	72.5	89.8	61.0	77.4	35.8	52.6	39.7	60.1	6.7	20.0	57.3	77.5
Multi-turn														
Qwen2.5-7B-Intent	85.7	96.7	65.9	91.0	54.3	78.4	23.0	52.9	30.9	63.9	5.0	20.0	41.6	85.0
UFO	90.5	96.5	72.9	90.6	62.2	77.7	34.5	51.8	39.8	61.5	7.7	30.0	58.3	85.0
S^2R	91.1	97.0	74.0	91.6	63.5	78.4	35.0	52.9	40.7	63.1	7.9	33.3	60.5	87.5
ReMA	91.8	96.7	71.7	87.8	62.4	75.8	35.6	51.5	39.6	57.8	7.9	26.7	59.8	82.5
Dr.MAS	92.1	97.6	72.0	92.6	63.1	78.4	35.7	54.8	41.2	65.5	8.1	33.3	61.3	87.5
Ours	**92.7**	**98.3**	**78.7**	**93.8**	**66.6**	**80.0**	**36.1**	**56.3**	**41.9**	**67.1**	**10.0**	**40.0**	**65.2**	**92.5**

trajectories; and **SIREN** (Jiang et al., 2025), which selectively regularizes entropy in informative high-entropy regions. Multi-turn baselines include **UFO** (Liu et al., 2025) for iterative revision from unary negative feedback; **S^2R** (Ma et al., 2025) for self-verification and self-correction with outcome- and process-level RL; **ReMA** (Wan et al., 2025), which jointly optimizes high-level meta-thinking and low-level reasoning across turns; and **Dr.MAS** (Feng et al., 2026), which uses role-wise advantage normalization for multi-agent RLVR. For a fair comparison, all methods use Qwen2.5-7B as the backbone and follow their respective initialization procedures.

Training and Evaluation. All methods are trained on the MATH training data for the same number of epochs, sampling 128 problems with 16 trajectories per problem. For I-Flow, each trajectory contains at most 30 turns, with a 4,096-token prompt limit and up to 512 new tokens per role at each turn. Further experimental details are provided in Appendix C. We evaluate on seven mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024), GaoKao 2023 (English) (Zhang et al., 2026), Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), AIME 2025[1], and AMC 2023[2], using a shared `math verify` checker. We further evaluate transfer to code generation on LiveCodeBench (Jain et al., 2025) and HumanEval+ (Liu et al., 2023) by executing their test cases. We report Avg@k and Pass@k, measuring average correctness over k trajectories and success with at least one correct trajectory, respectively. For the main evaluation, we set $k = 16$ and sample with temperature 0.6 and top-$p = 1.0$. Appendix D.1 additionally reports Pass@1 with temperature 0 for deterministic generation. Overall, I-Flow achieves consistent improvements across reasoning domains and generation settings.

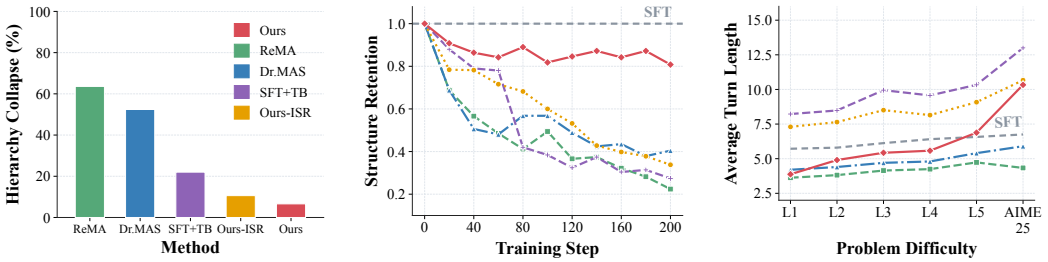
5.2 Performance Comparison

As shown in Table 2, I-Flow consistently improves both Avg@16 and Pass@16 across all seven mathematical reasoning benchmarks, with particularly pronounced gains on challenging benchmarks such as AIME25 and AMC23. Improvements in both metrics indicate that I-Flow not only improves average reasoning performance but also increases the likelihood of finding successful solutions across multiple reasoning trajectories. These results support the effectiveness of our intent-based approach in enabling meaningfully diverse and effective reasoning. The same strengths are reflected in cross-domain transfer to code generation. As shown in Table 3, despite being trained only on mathematical data, I-Flow consistently outperforms all multi-turn baselines in both Avg@16 and Pass@16 on LiveCodeBench

Table 3 | Comparison results on LiveCodeBench and HumanEval+

Method	LiveCodeBench		HumanEval+	
	A@16	P@16	A@16	P@16
Qwen2.5-7B-Intent	6.65	35.55	40.40	84.15
UFO	9.10	36.02	43.37	82.93
S^2R	9.55	35.55	44.28	82.32
ReMA	8.70	33.74	42.80	80.49
Dr.MAS	9.95	36.87	45.62	84.15
Ours	**12.48**	**39.62**	**48.21**	**86.59**

[1] `https://huggingface.co/datasets/AI-MO/aimo-validation-aime`
[2] `https://huggingface.co/datasets/AI-MO/aimo-validation-amc`



(a) Hierarchy Collapse (b) Structure Retention (c) Average Turn Length

Figure 4: (a) Hierarchy collapse rate by method. (b) Structure Retention over training steps relative to SFT (gray). (c) Average turn length across MATH500 difficulty levels L1–L5 and on AIME25. Structure retention measures the fraction of distinct intent structures observed in SFT rollouts that reappear in checkpoint rollouts.

and HumanEval+. This demonstrates that the learned reasoning behavior transfers effectively to code generation without code-specific fine-tuning. The Pass@1 results in Appendix D.1 show that I-Flow achieves consistent improvements under deterministic generation, while Appendix D.2 further reports similar gains with Gemma3-4B-Instruct. Overall, these results demonstrate the consistent effectiveness of I-Flow across reasoning domains, generation settings, and model backbones.

5.3 Analysis of Reasoning Structures

We further analyze how our proposed components improve multi-turn reasoning through their effects on reasoning structures. Unless otherwise specified, all analyses are conducted on MATH500, with AIME25 additionally used for the reasoning-length analysis.

RQ1: How does our method preserve and adapt reasoning structures? We compare I-Flow with existing multi-turn methods, ReMA and Dr.MAS, as well as component-wise variants. SFT denotes the base Qwen2.5-7B-Intent model; SFT+TB applies TB-based flow optimization over the intent structure without ISR, while Ours-ISR removes only ISR. Figure 4(a) reports the hierarchy collapse rate, where GPT-4o-mini judges whether the high-level output serves as guidance; the full judging prompt is provided in Appendix C.5. I-Flow shows the lowest collapse rate compared with existing multi-turn methods, while removing either component increases it. SFT+TB reduces collapse to some extent but fails to sufficiently prevent it without intermediate supervision, highlighting the importance of our BF-SubTB objective. Figure 4(b) measures structure retention by how well intent structures observed in the SFT model are retained throughout training. Again, I-Flow retains the largest fraction of these structures among existing multi-turn methods, while removing components progressively reduces structure retention. This indicates that I-Flow not only prevents hierarchy collapse but also effectively mitigates reasoning simplification during multi-turn RL optimization.

Figure 4(c) further examines turn length across problem difficulties. While all methods tend to use more turns for harder problems, I-Flow uses fewer turns than SFT for easier problems and more for harder ones, demonstrating more efficient use of reasoning turns. In contrast, ReMA and Dr.MAS generally produce short trajectories across difficulty levels, while our ablated variants use unnecessarily long trajectories even for easier problems. Together, these results show that I-Flow substantially *mitigates both hierarchy collapse and reasoning simplification* introduced in Section 1, while efficiently adapting its reasoning length to problem difficulty.

RQ2: What structural patterns emerge with performance gains? We examine how reasoning structures evolve with performance gains, focusing on problems that SFT fails but the optimized policy solves correctly. Figure 5 decomposes these gains into *reused*, where successful trajectories follow intent structures already observed in SFT, and *novel*, where new structures emerge during training. Approximately 30% of the gains in each training phase involve novel structures, indicating that I-Flow not only reuses prior structures but also discovers new successful ones. Table 4 further shows that successful trajectories adopt sequence-level patterns intuitively beneficial for problem solving, such as trimming redundant turns and verifying before finalization. Figure 6 evaluates novel structures on a subset of MATH500 problems where I-Flow succeeds but SFT fails under Pass@1, using the fixed SFT model. With a Pass@16 budget, we compare SFT using its original failed intent structures with SFT following I-Flow's novel structures, denoted as *SFT + I-Flow's Intent*. SFT

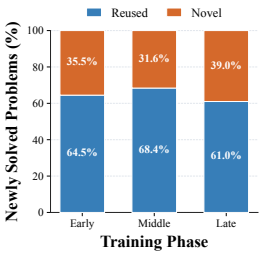


Figure 5: Intent structure types in newly solved problems.

Pattern	Rate
Trim redundant turns	39.8%
Verify before finalizing	30.6%
Reduce exploration restarts	21.2%
Finish an abandoned run	5.7%
Others	2.7%

Table 4 | Emerging intent-sequence patterns in successful trajectories.

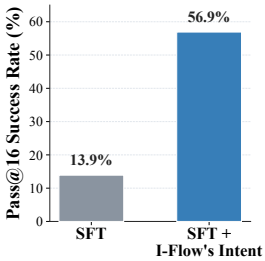


Figure 6: Effect of novel intent structures on the SFT model.

Table 5 | Ablation study on mathematical reasoning benchmarks.

Method		GSM8K		MATH500		Gaokao23		Minerva		Olympiad		AIME25		AMC23	
		A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16
Ours		**92.7**	**98.3**	**78.7**	**93.8**	**66.6**	**80.0**	**36.1**	**56.3**	**41.9**	**67.1**	**10.0**	**40.0**	**65.2**	**92.5**
Intent	→ w/o ISR	92.3	98.1	74.6	91.8	65.2	78.7	35.8	52.9	38.7	61.6	7.7	26.7	62.5	87.5
	→ w/o Intent	89.8	96.1	71.7	90.6	61.4	77.4	35.0	51.1	37.7	53.3	5.2	20.0	57.2	72.5
Objective	SubTB	88.1	93.1	70.6	86.2	56.9	71.2	33.5	48.9	36.4	58.7	6.7	20.0	52.5	67.5
	TB	89.8	95.3	72.0	88.2	61.0	74.8	34.5	53.3	38.2	63.7	6.7	23.3	57.0	72.5
Replay	Full Text Replay	89.0	96.6	72.6	91.2	63.6	77.1	35.5	52.2	37.1	61.8	7.3	20.0	62.2	72.5

solves only 13.9% of these problems with its original structures, whereas I-Flow's novel structures increase the success rate to over 59%, indicating that the discovered structures themselves provide effective reasoning guidance. Appendix D.3 further analyzes how the learned flow evolves along reasoning trajectories, while Appendix D.4 provides representative case studies.

5.4 Ablation Study and Further Analysis

Ablation Study. Table 5 examines our method from three aspects: intent formulation, flow objective, and replay strategy. Removing ISR and further removing the intent structure progressively degrades performance, supporting the importance of explicitly structuring high-level reasoning decisions and preserving successful structures throughout training. Replacing our BF-SubTB objective with either SubTB or TB also reduces performance. SubTB requires bootstrapping from learned nonterminal flows, while TB provides no intermediate supervision, supporting our design that combines intermediate supervision with direct terminal anchoring. Finally, ISR outperforms Full Text Replay, suggesting that preserving successful intent structures while freshly generating guidance and reasoning is more effective than replaying complete textual trajectories.

Additional Analysis. We further analyze hyperparameter sensitivity, entropy regularization, and computational cost in Appendix E. Regarding the hyperparameters, I-Flow remains robust to α, while ε reveals a trade-off between leveraging successful structures through ISR and exploring new ones. I-Flow also consistently outperforms explicit intent modeling with entropy regularization, highlighting the benefit of flow-based supervision for learning effective intent structures from verified outcomes. In addition, our cost analysis shows that the additional overhead of flow-based optimization accounts for only about 0.1% of the training step time, with the overall computational cost dominated by trajectory generation and thus largely determined by trajectory length. We discuss limitations and future directions in Appendix A.

6 Conclusion

In this work, we identified hierarchy collapse and reasoning simplification as key failure modes of final-reward optimization in hierarchical multi-turn reasoning. To address these issues, we proposed I-Flow, which explicitly represents turn-level intents and optimizes diverse reasoning structures through bootstrap-free subtrajectory balance and intent structure replay. Experiments on mathematical reasoning and code generation demonstrate consistent performance improvements while effectively mitigating both failure modes. These results highlight the importance of explicitly representing and preserving reasoning structures throughout multi-turn optimization.

AI USE STATEMENT

In this work, generative AI tools were used only to assist with manuscript writing, including grammar, wording, clarity, and readability. As large language models are the subject of this study, LLMs were also used for training, inference, and evaluation as part of the experimental methodology. The problem formulation, methodology, experimental design, analysis, and conclusions were developed and verified by the authors, who take full responsibility for the content of this work.

ETHICS STATEMENT

This work studies reinforcement learning for improving multi-turn reasoning in large language models using publicly available models, training datasets, and evaluation benchmarks for mathematical reasoning and code generation. No human subjects, personal data, or sensitive information were involved in our experiments. While the proposed method improves LLM reasoning capabilities, models trained with such methods may still produce plausible but incorrect outputs or inherit broader risks associated with large language models. Our experiments are conducted solely in research-oriented reasoning settings and do not involve deployment in high-stakes applications. All research was conducted in accordance with the ICLR Code of Ethics, and the authors take responsibility for the methodology, experiments, and reported results.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed descriptions of the data preparation and intent-revealing SFT procedure, optimization and generation settings, and all high-level, low-level, and evaluation prompts in Appendix C. The appendix also reports the hyperparameters and implementation details used throughout training and evaluation. Complete proofs of our theoretical results, including the assumptions underlying each proposition, are provided in Appendix B. We will additionally release our implementation and experimental configurations to facilitate reproduction of the reported results.

REFERENCES

Sangjun Bae, Yisak Park, Sanghyeon Lee, and Seungyul Han. LLM-guided communication for cooperative multi-agent reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 27381–27394. Curran Associates, Inc., 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. URL http://jmlr.org/papers/v24/22-0364.html.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Jongseong Chae, Jongeui Park, Yongjae Shin, Gyeongmin Kim, Seungyul Han, and Youngchul Sung. Flow actor-critic for offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Xin Zhao, and Ji-Rong Wen. ChatCoT: Tool-augmented chain-of-thought reasoning on chat-based large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14777–14790, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.985. URL https://aclanthology.org/2023.findings-emnlp.985/.

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

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. Successive prompting for decomposing complex questions. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1251–1265, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.81. URL https://aclanthology.org/2022.emnlp-main.81/.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: Stable reinforcement learning for multi-agent LLM systems. In *Workshop on Multi-Agent Learning and Its Opportunities in the Era of Generative AI*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Seungyul Han and Youngchul Sung. Diversity actor-critic: Sample-aware entropy regularization for sample-efficient exploration. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4018–4029. PMLR, 2021a.

Seungyul Han and Youngchul Sung. A max-min entropy framework for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:25732–25745, 2021b.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Yechan Hwang, Sangjun Bae, Jeongmo Kim, Sangwoo Bang, and Seungyul Han. MARBO: Relational belief grounding for LLM agents in social deduction games. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

Naman Jain, Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Yuxian Jiang, Yafu Li, Guanxu Chen, Dongrui Liu, Yu Cheng, and Jing Shao. Rethinking entropy regularization in large reasoning models, 2025. URL https://arxiv.org/abs/2509.25133.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*, 2025.

Dohyung Kim, Minbeom Kim, Jeonghye Kim, Sangmook Lee, Sojeong Rhee, and Kyomin Jung. Beyond normalization: Rethinking the partition function as a difficulty scheduler for RLVR. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=U93dBehANs.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025.

Seanie Lee, Minsu Kim, Lynn Cherif, David Dobre, Juho Lee, Sung Ju Hwang, Kenji Kawaguchi, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, and Moksh Jain. Learning diverse attacks on large language models for robust red-teaming and safety tuning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 63141–63163, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/9e128701ceca0ba03658a305faf39deb-Paper-Conference.pdf.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857. Curran Associates, Inc., 2022. doi: 10.52202/068431-0278.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.992.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Licheng Liu, Zihan Wang, Linjie Li, Chenwei Xu, Yiping Lu, Han Liu, Avirup Sil, and Manling Li. Let's try again: Eliciting multi-turn reasoning in language models via simplistic feedback. In *2nd AI for Math Workshop @ ICML 2025*, 2025.

Xuanzhang Liu, Jianglun Feng, Zhuoran Zhuang, Junzhe Zhao, Maofei Que, Jieting Li, Dianlei Wang, Hao Tong, Ye Chen, and Pan Li. Coda: A context-decoupled hierarchical agent with reinforcement learning. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, WSDM '26, pp. 436–446, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722929. doi: 10.1145/3773966.3777986.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22632–22654, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1104.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019.

Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. Learning GFlowNets from partial episodes for improved convergence and stability. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23467–23483. PMLR, 23–29 Jul 2023.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical plan–execute RL for multi-turn LLM agents. In *Forty-third International Conference on Machine Learning*, 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin

Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 126621–126667. Curran Associates, Inc., 2025. doi: 10.52202/085713-4225.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 115452–115486. Curran Associates, Inc., 2025. doi: 10.52202/085713-3850.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800.

Esmeralda S Whitammer, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 5955–5967. Curran Associates, Inc., 2022. doi: 10.52202/068431-0431.

Jian Yao, Ran Cheng, Xingyu Wu, Jibin Wu, and KC Tan. Diversity-aware policy optimization for large language model reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 94801–94826. Curran Associates, Inc., 2025. doi: 10.52202/085713-3169.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. doi: 10.52202/075280-0517.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. LIMO: Less is more for reasoning. In *Second Conference on Language Modeling*, 2025.

Junghyuk Yeom, Yonghyeon Jo, Jungmo Kim, Sanghyeon Lee, and Seungyul Han. Exclusively penalized q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:113405–113435, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244. Curran Associates, Inc., 2025. doi: 10.52202/085713-3775.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, Xipeng Qiu, Tianxiang Sun, Peng Li, Shiqiao Meng, Yanjun Zheng, Jun Zhan, Zhangyue Yin, Xiannian Hu, Guofeng Quan, and Qixiang Wang. Evaluating the performance of large language models on gaokao benchmark, 2026. URL https://arxiv.org/abs/2305.12474.

Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7553–7562. PMLR, 2019.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024a.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 21–27 Jul 2024b.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Henry Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, Qizheng Zhang, Lin Chen, Fanghao Shao, Bo Xue, Yunchong Song, Zhenjie Yang, Ganqu Cui, Ning Ding, Jianfeng Gao, Xiaodong Liu, Bowen Zhou, Hongyuan Mei, and Zhouhan Lin. Flowrl: Matching reward distributions for llm reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 151758–151780, 2026.

APPENDIX CONTENTS

A. Limitations and Future Work

B. Theoretical Analysis

B.1. Intent-Conditioned Flow Semantics

B.2. Proof of Proposition 1

B.3. Proof of Proposition 2

C. Implementation and Experimental Details

C.1. Data Preparation and Intent-Revealing SFT

C.2. Optimization and Generation Settings

C.3. Details of Intent Structure Replay

C.4. High-Level and Low-Level System Prompts

C.5. LLM-as-a-Judge Prompt

D. Additional Experimental Analyses

D.1. Pass@1 Performance Comparison

D.2. Results with Different Backbones

D.3. Learned Flow and Intent Selection

D.4. Case Study

E. Additional Ablation Studies and Efficiency

E.1. Sensitivity to Hyperparameters

E.2. Comparison with Entropy Regularization

E.3. Computational Complexity and Empirical Cost

A Limitations and Future Directions

Despite its effectiveness in improving multi-turn reasoning and mitigating hierarchy collapse and reasoning simplification, our framework has several limitations and corresponding directions for future work. First, I-Flow relies on a predefined intent taxonomy to represent high-level reasoning decisions. While our experiments show that the same intent categories transfer from mathematical reasoning to code generation, substantially different domains may require adapting or extending this taxonomy. A natural future direction is to learn or discover intent categories automatically, allowing reasoning structures to adapt to different tasks and domains with less manual specification. Second, our evaluation focuses on tasks with verifiable terminal outcomes, where reasoning trajectories can be optimized using objective correctness signals. Extending I-Flow to more open-ended reasoning settings would require appropriate reward models or alternative feedback signals for evaluating reasoning outcomes. Exploring such settings would help assess how broadly intent-based flow optimization can generalize beyond verifiable reasoning tasks.

B Theoretical Analysis

We provide three theoretical analyses of I-Flow. First, Appendix B.1 shows that the intent-conditioned flow $F^*(c, u)$ is strictly increasing in the optimal continuation success probability under binary rewards. Second, Appendix B.2 establishes that BF-SubTB identifies the optimal continuation policy π^* and flow F^* at every state while preserving the original RLVR target. Third, Appendix B.3 bounds local flow-balance errors by the I-Flow objective, showing how it controls intermediate inconsistencies along the reasoning trajectory.

B.1 Intent-Conditioned Flow Semantics

For notational simplicity, throughout this subsection we write

$$\Pr_{\pi}(r = 1 \mid c, u) := \Pr_{\boldsymbol{\tau}_{t:} \sim \pi(\cdot|c,u)} [r(\mathbf{x}, \boldsymbol{\tau}) = 1], \tag{B.1}$$

with the fixed input $\mathbf{x}$ and the trajectory dependence of the terminal reward left implicit. We use $\Pr_{\pi}(r = 0 \mid c, u)$ analogously.

Proposition (Continuation Flow and Optimal-Policy Success)**.** *Assume $\beta > 0$ and a binary terminal reward $r(\mathbf{x}, \boldsymbol{\tau}) \in \{0, 1\}$. Fix a reachable pre-intent context c and a candidate intent u such that $\pi^I_{\text{ref}}(u \mid c) > 0$. Then the exact intent-conditioned continuation flow in Eq. 11 satisfies*

$$F^*(c, u) = \frac{1}{1 - (1 - e^{-1/\beta}) \Pr_{\pi^*}(r = 1 \mid c, u)}. \tag{B.2}$$

Consequently, for any two candidate intents u_1 and u_2 at the same context c,

$$F^*(c, u_1) > F^*(c, u_2) \iff \Pr_{\pi^*}(r = 1 \mid c, u_1) > \Pr_{\pi^*}(r = 1 \mid c, u_2). \tag{B.3}$$

Proof. Fix a context–intent pair (c, u) satisfying the conditions above. Let $\boldsymbol{\tau}_{t:}$ denote the remaining generation after selecting intent u from context c.

From the KL-regularized optimal trajectory distribution in Eq. 3, conditioning on the fixed context–intent pair (c, u) gives

$$\pi^*(\boldsymbol{\tau}_{t:} \mid c, u) = \frac{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid c, u) \exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big)}{\mathbb{E}_{\boldsymbol{\tau}'_{t:} \sim \pi_{\text{ref}}(\cdot|c,u)} \big[\exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}')\big)\big]}. \tag{B.4}$$

By the definition in Eq. 11, the denominator is exactly $F^*(c, u)$. Hence,

$$\pi^*(\boldsymbol{\tau}_{t:} \mid c, u) = \pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid c, u) \frac{\exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big)}{F^*(c, u)}. \tag{B.5}$$

Since $r(\mathbf{x}, \boldsymbol{\tau}) \in \{0, 1\}$, evaluating the expectation in Eq. 11 over the two possible reward outcomes yields

$$\begin{aligned} F^*(c, u) &= \mathbb{E}_{\boldsymbol{\tau}_{t:} \sim \pi_{\text{ref}}(\cdot|c,u)} \big[\exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big)\big] && \text{(B.6)} \\ &= e^0 \Pr_{\pi_{\text{ref}}}(r = 0 \mid c, u) + e^{1/\beta} \Pr_{\pi_{\text{ref}}}(r = 1 \mid c, u) \\ &= 1 + \Big(e^{1/\beta} - 1\Big) \Pr_{\pi_{\text{ref}}}(r = 1 \mid c, u). && \text{(B.7)} \end{aligned}$$

We next relate the reference-policy success probability to that under the KL-regularized optimal policy. Using Eq. B.5,

$$\begin{aligned}\Pr_{\pi^*}(r=1 \mid c,u) &= \sum_{\boldsymbol{\tau}_{t:}:r(\mathbf{x},\boldsymbol{\tau})=1} \pi^*(\boldsymbol{\tau}_{t:} \mid c,u) \\ &= \frac{e^{1/\beta}}{F^*(c,u)} \sum_{\boldsymbol{\tau}_{t:}:r(\mathbf{x},\boldsymbol{\tau})=1} \pi_{\mathrm{ref}}(\boldsymbol{\tau}_{t:} \mid c,u) \\ &= \frac{e^{1/\beta} \Pr_{\pi_{\mathrm{ref}}}(r=1 \mid c,u)}{F^*(c,u)}. \end{aligned} \tag{B.8}$$

Therefore,

$$\Pr_{\pi_{\mathrm{ref}}}(r=1 \mid c,u) = e^{-1/\beta} F^*(c,u) \Pr_{\pi^*}(r=1 \mid c,u). \tag{B.9}$$

Substituting Eq. B.9 into Eq. B.7 gives

$$\begin{aligned} F^*(c,u) &= 1 + \left(e^{1/\beta}-1\right) e^{-1/\beta} F^*(c,u) \Pr_{\pi^*}(r=1 \mid c,u) \\ &= 1 + \left(1-e^{-1/\beta}\right) F^*(c,u) \Pr_{\pi^*}(r=1 \mid c,u). \end{aligned} \tag{B.10}$$

Rearranging Eq. B.10 yields

$$F^*(c,u) = \frac{1}{1-(1-e^{-1/\beta}) \Pr_{\pi^*}(r=1 \mid c,u)}. \tag{B.11}$$

which proves Eq. B.2.

Finally, since $\beta > 0$, we have $0 < 1 - e^{-1/\beta} < 1$. Differentiating Eq. B.2 with respect to the optimal-policy continuation success probability gives

$$\frac{\partial F^*(c,u)}{\partial \Pr_{\pi^*}(r=1 \mid c,u)} = \frac{1-e^{-1/\beta}}{\left[1-(1-e^{-1/\beta}) \Pr_{\pi^*}(r=1 \mid c,u)\right]^2} > 0. \tag{B.12}$$

Hence, $F^*(c,u)$ is strictly increasing in $\Pr_{\pi^*}(r=1 \mid c,u)$, which directly implies Eq. B.3. □

Interpretation. In the binary-reward RLVR setting considered in this work, Proposition gives the continuation flow a direct probabilistic interpretation. For a fixed reasoning context c, an intent u has a larger $F^*(c,u)$ exactly when continuing from that intent under the KL-regularized optimal policy is more likely to eventually achieve the successful terminal outcome $r = 1$. Thus, $F^*(c,u)$ can be viewed as an intent-level measure of downstream reasoning potential, turning terminal-only supervision into a meaningful ordering over intermediate intent decisions. This provides the semantic basis for using post-intent flow constraints to distinguish more promising reasoning directions. Under finite optimization, the learned flow F_ϕ need not be perfectly calibrated to F^* in absolute scale, but the above characterization motivates its use as a relative signal of continuation quality.

B.2 Proof of Proposition 1

Proposition 1 (Policy and Flow Identification)**.** *Assume a finite horizon, common positive support between π_θ and π_{ref}, and that π^* and F^* are realizable under the parameterization. Then, all BF-SubTB conditions hold if and only if*

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \pi^*(\boldsymbol{\tau}_{t:} \mid s_t), \qquad F_\phi(s_t) = F^*(s_t), \qquad 0 \le t \le T, \tag{B.13}$$

for every state s_t. In particular, since $s_0 = \mathbf{x}$, $\pi_\theta(\boldsymbol{\tau} \mid \mathbf{x}) = \pi^*(\boldsymbol{\tau} \mid \mathbf{x})$.

Proof. Suppose all BF-SubTB conditions are satisfied. For any reachable state s_t, $0 \le t \le T$, $R_t(\boldsymbol{\tau}) = 0$ gives

$$\log F_\phi(s_t) + \log \frac{\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t)}{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t)} - \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}) = 0. \tag{B.14}$$

Rearranging the equality yields

$$\log \frac{\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t)}{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t)} = \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}) - \log F_\phi(s_t), \tag{B.15}$$

and therefore

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \frac{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t) \exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big)}{F_\phi(s_t)}. \tag{B.16}$$

For a fixed s_t, $F_\phi(s_t)$ is constant with respect to the continuation $\boldsymbol{\tau}_{t:}$. Since $\pi_\theta(\cdot \mid s_t)$ is normalized, summing Eq. (B.16) over all continuations gives

$$\begin{aligned} F_\phi(s_t) &= \sum_{\boldsymbol{\tau}_{t:}} \pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t) \exp\big(\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})\big) \\ &= F^*(s_t), \end{aligned} \tag{B.17}$$

where the last equality follows from Eq. (11). Substituting $F_\phi(s_t) = F^*(s_t)$ into Eq. (B.16) and using Eq. (12) yields

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \pi^*(\boldsymbol{\tau}_{t:} \mid s_t). \tag{B.18}$$

Hence, for every reachable state s_t,

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \pi^*(\boldsymbol{\tau}_{t:} \mid s_t), \qquad F_\phi(s_t) = F^*(s_t). \tag{B.19}$$

In particular, at $t = 0$, since $s_0 = \mathbf{x}$,

$$\pi_\theta(\boldsymbol{\tau} \mid \mathbf{x}) = \pi^*(\boldsymbol{\tau} \mid \mathbf{x}), \tag{B.20}$$

which recovers the original RLVR target.

Conversely, suppose that for every reachable state s_t,

$$\pi_\theta(\boldsymbol{\tau}_{t:} \mid s_t) = \pi^*(\boldsymbol{\tau}_{t:} \mid s_t), \qquad F_\phi(s_t) = F^*(s_t). \tag{B.21}$$

Then, by Eq. (12),

$$\begin{aligned} R_t(\boldsymbol{\tau}) &= \log F^*(s_t) + \log \frac{\pi^*(\boldsymbol{\tau}_{t:} \mid s_t)}{\pi_{\text{ref}}(\boldsymbol{\tau}_{t:} \mid s_t)} - \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}) \\ &= \log F^*(s_t) + \big[\beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}) - \log F^*(s_t)\big] - \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}) \\ &= 0. \end{aligned} \tag{B.22}$$

Thus, $R_t(\boldsymbol{\tau}) = 0$ for every $0 \le t \le T$, and all BF-SubTB conditions are satisfied, completing the proof. □

B.3 Proof of Proposition 2

Proposition 2 (Local Flow-Balance Control)**.** *Consider any completed trajectory $\boldsymbol{\tau} = (s_0, \ldots, s_{T+1})$ with $T \geq 1$, positive policy and reference transition probabilities, and positive flows. Define the local reference-relative balance residual*

$$\Delta_t = \log F_\phi(s_t) + \log \frac{P_\theta(s_{t+1} \mid s_t)}{P_{\mathrm{ref}}(s_{t+1} \mid s_t)} - \log F_\phi(s_{t+1}), \qquad 0 \leq t \leq T, \tag{B.23}$$

with terminal boundary $\log F_\phi(s_{T+1}) = \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau})$. Then the total local flow-balance error is bounded by

$$\sum_{t=0}^{T} \Delta_t^2 \leq \frac{4}{\min\{1, \alpha/T\}} \mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}). \tag{B.24}$$

Proof. For brevity, define

$$\ell_t := \log \frac{P_\theta(s_{t+1} \mid s_t)}{P_{\mathrm{ref}}(s_{t+1} \mid s_t)}. \tag{B.25}$$

Then the local reference-relative balance residual in Eq. (B.23) is

$$\Delta_t = \log F_\phi(s_t) + \ell_t - \log F_\phi(s_{t+1}). \tag{B.26}$$

Recall that the terminal-anchored suffix residual beginning at s_t is

$$R_t(\boldsymbol{\tau}) = \log F_\phi(s_t) + \sum_{k=t}^{T} \ell_k - \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}). \tag{B.27}$$

Using the terminal boundary condition

$$\log F_\phi(s_{T+1}) = \beta^{-1} r(\mathbf{x}, \boldsymbol{\tau}),$$

and, for convenience, define $R_{T+1} := 0$. Then

$$\Delta_t = R_t - R_{t+1}, \qquad 0 \leq t \leq T. \tag{B.28}$$

Therefore,

$$\sum_{t=0}^{T} \Delta_t^2 = \sum_{t=0}^{T-1} (R_t - R_{t+1})^2 + R_T^2. \tag{B.29}$$

Applying the elementary inequality

$$(a-b)^2 \leq 2a^2 + 2b^2$$

to each difference term yields

$$\sum_{t=0}^{T} \Delta_t^2 \leq 2 \sum_{t=0}^{T-1} R_t^2 + 2 \sum_{t=0}^{T-1} R_{t+1}^2 + R_T^2 \tag{B.30}$$

$$= 2R_0^2 + 4 \sum_{t=1}^{T-1} R_t^2 + 3R_T^2 \tag{B.31}$$

$$\leq 4 \sum_{t=0}^{T} R_t^2. \tag{B.32}$$

We now relate the unweighted sum of suffix residuals to the I-Flow objective. Let

$$\lambda := \frac{\alpha}{T}, \qquad m := \min\{1, \lambda\}.$$

Since $1 \geq m$ and $\lambda \geq m$,

$$\mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}) = R_0(\boldsymbol{\tau})^2 + \lambda \sum_{t=1}^{T} R_t(\boldsymbol{\tau})^2 \tag{B.33}$$

$$\geq m \sum_{t=0}^{T} R_t(\boldsymbol{\tau})^2. \tag{B.34}$$

Equivalently,

$$\sum_{t=0}^{T} R_t^2 \leq \frac{1}{\min\{1, \alpha/T\}} \mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}). \tag{B.35}$$

Combining Eq. (B.32) and Eq. (B.35), we obtain

$$\sum_{t=0}^{T} \Delta_t^2 \leq \frac{4}{\min\{1, \alpha/T\}} \mathcal{L}_{\text{I-Flow}}(\boldsymbol{\tau}), \tag{B.36}$$

which proves Eq. (B.24). □

C Implementation and Experimental Details

This section provides the implementation and experimental details of I-Flow. Appendix C.1 describes how LIMO solutions are converted into intent-annotated dialogues between a high-level controller and a low-level reasoner, along with the SFT configuration. Appendix C.2 summarizes the RL implementation, the parameterization of the flow F_ϕ, and the hyperparameters used for training and evaluation. Appendix C.3 details Intent Structure Replay, including buffer construction, replay-structure sampling, and structure-guided generation. Appendix C.4 provides the high-level and low-level system prompts for mathematical reasoning and code generation, and Appendix C.5 presents the LLM-as-a-judge prompt used to evaluate hierarchy collapse.

C.1 Data Preparation and Intent-Revealing SFT

Data preparation. We construct the intent-revealing SFT corpus from LIMO (Ye et al., 2025), which contains 817 competition-level problems accompanied by long-form reference solutions. We choose LIMO because its detailed solution traces provide sufficient context to distinguish high-level guidance from the low-level reasoning that executes it. Using a frozen teacher (`gpt-4o-2024-11-20`), we therefore decompose each reference solution into alternating high-level and low-level turns according to the prompt below. For every high-level turn, the teacher additionally identifies its underlying intent and annotates it with an explicit `<action> ... </action>` tag corresponding to the action that best describes the function of that turn.

To prevent the teacher's own reasoning capability from altering the source solution during this conversion, we impose strict constraints on the rewriting process. In particular, the teacher must preserve the original reasoning trajectory, including any mistakes and subsequent corrections, and is prohibited from introducing new reasoning steps merely to produce or balance particular action types. These constraints ensure that the resulting action annotations reveal the intents already present in the original solution rather than imposing an externally designed reasoning structure, yielding the intent-revealing SFT corpus used for training.

Intent-Revealing SFT. For supervised finetuning, we use veRL (Sheng et al., 2025) as the codebase and train Qwen2.5-7B for 3 epochs with a learning rate of 1e-5, cosine learning rate scheduler without warmup, and batch size of 8. We use AdamW ($\beta = (0.9, 0.95)$, weight decay 0.01) with gradient clipping at 1.0 and a maximum sequence length of 5,120. We use FSDP with bfloat16 for distributed training on 8 GPUs.

Intent Revealing SFT Prompt

Convert the given math solution into alternating high_level and low_level turns.
- high_level: briefly direct the next step, starting with one action tag.
- low_level: perform that mathematical step.

Choose the action by its actual function:
- idle: routine progress within the current approach.
- exploration: introduce a new idea or approach.
- verification: check an existing result.
- backtracking: correct an actual error or revise a failed approach.
- continue: resume a step interrupted by a response-length limit.
- finalize: conclude with the established answer.

Preserve the original reasoning, including mistakes and corrections.
Do not invent steps to force action coverage; not all six actions are required.
Write as the solver without mentioning the reference.

Output only:

```
<high_level><action>ACTION</action>Brief direction.</high_level>
<low_level>One mathematical step.</low_level>
...
```

End with:

```
<high_level><action>finalize</action>Brief confirmation.
[FINISH]</high_level>
<low_level>\boxed{FINAL_ANSWER}</low_level>
```

Use `[FINISH]` and `\boxed{}` only in the final pair.

Question: {question}
Correct answer: {answer}
Reference solution: {solution}

C.2 Optimization and Generation Settings

We implement all RL experiments using veRL (Sheng et al., 2025), with FSDP for distributed training and vLLM for rollout generation. Training is conducted on eight NVIDIA A100-SXM4-80GB GPUs. Unless otherwise specified, the optimization and generation settings reported in Table C.1 are used throughout our experiments.

For the flow F_ϕ required by BF-SubTB in Eq. (10), we follow the parameterization commonly adopted in prior GFlowNet-based language-model optimization methods (Lee et al., 2025; Zhu et al., 2026; Kim et al., 2026). Specifically, the flow F_ϕ is predicted by a three-layer MLP operating on hidden representations from the frozen reference model.

Table C.1 | Training and decoding settings.

Stage	Setting	Value
Optimization	Train batch (problems)	128
	Trajectories per problem	16
	Mini batch	64
	Per-GPU micro-batch	8
	Total training steps	250
	Policy / flow-head learning rates	10^{-6} / 10^{-4}
	Learning-rate schedule	constant;
	Optimizer	AdamW
	Gradient clipping	1.0 (policy)
Objective	α (BF-SubTB weight)	0.2
	β (reward scale)	3.0
	Flow head	3-layer MLP on the frozen reference model's hidden state
	Importance weight	trajectory-level, clipped at 10.0
Rollout	Maximum interaction turns	30
	Max prompt length	4,096 tokens
	Max response length	512 tokens
	Temperature / top-p / top-k	1.0 / 1.0 / -1
Replay	Target replay fraction	0.25 (4 of 16 trajectories)
Evaluation	Trajectories per problem	16 for Avg@16 and Pass@16
	Temperature / top-p / top-k	0.6 / 1.0 / -1
	Maximum interaction turns	30
	Max response length	512 tokens

C.3 Details of Intent Structure Replay

This section details the implementation of Intent Structure Replay (ISR), including buffer construction, replay sampling, and structure-guided generation. ISR stores successful intent structures rather than complete reasoning trajectories and regenerates their high- and low-level realizations using the current policy.

Per-question replay buffer. For each problem $\mathbf{x}$, we maintain a replay buffer containing intent structures that have produced at least one verified correct answer:

$$\mathcal{B}(\mathbf{x}) = \left\{ \left(\sigma, n_{\text{trial}}(\sigma), n_{\text{succ}}(\sigma)\right) : n_{\text{succ}}(\sigma) \geq 1 \right\}, \tag{C.1}$$

where $\sigma = (u_1, \ldots, u_T)$ denotes an intent structure. A previously unseen structure is inserted upon its first verified success, with

$$n_{\text{trial}}(\sigma) = 1, \qquad n_{\text{succ}}(\sigma) = 1. \tag{C.2}$$

Once admitted, all subsequent executions of the structure are tracked. Thus, its statistics summarize executions from the first verified success onward:

$$n_{\text{trial}}(\sigma) \leftarrow n_{\text{trial}}(\sigma) + 1, \tag{C.3}$$

$$n_{\text{succ}}(\sigma) \leftarrow n_{\text{succ}}(\sigma) + \mathbb{1}[r(\mathbf{x}, \boldsymbol{\tau}) = 1]. \tag{C.4}$$

Because ISR stores only intent tags and their counts rather than complete reasoning trajectories, its memory cost is small relative to full-text replay.

Replay-structure sampling. For each stored structure $\sigma \in \mathcal{B}(\mathbf{x})$, we compute a smoothed replay success estimate

$$\widehat{p}_{\mathbf{x}}(\sigma) = \frac{n_{\text{succ}}(\sigma) + 1}{n_{\text{trial}}(\sigma) + 2}. \tag{C.5}$$

The added virtual success and virtual failure prevent estimates from becoming extreme after only a few executions. Replay structures are then sampled in proportion to these estimates:

$$q_{\text{ISR}}(\sigma \mid \mathbf{x}) = \frac{\widehat{p}_{\mathbf{x}}(\sigma)}{\sum_{\sigma' \in \mathcal{B}(\mathbf{x})} \widehat{p}_{\mathbf{x}}(\sigma')}. \tag{C.6}$$

This favors structures that remain effective across repeated executions while retaining nonzero replay probability for every stored structure.

For each problem, a fraction ε of the rollout budget is allocated to structure-guided generation and the remainder to free rollouts from the current policy. With our default $\varepsilon = 0.25$ and 16 trajectories per problem, 4 trajectories are structure-guided and 12 are free. If $\mathcal{B}(\mathbf{x})$ is empty, all 16 trajectories are generated freely.

Structure-guided generation. For each replay trajectory, ISR samples

$$\sigma = (u_1, \ldots, u_T) \sim q_{\text{ISR}}(\cdot \mid \mathbf{x}). \tag{C.7}$$

Generation then follows the sampled intent sequence. At turn t, the intent u_t is fixed to the corresponding element of σ, while the high-level guidance h_t and low-level execution a_t are freshly generated by the current policy:

$$(h_t, a_t) \sim \pi_\theta(\cdot \mid c_t, u_t). \tag{C.8}$$

The generated (h_t, a_t) is appended to the reasoning history before the next stored intent is applied. ISR therefore constrains only the sequence of high-level decisions, not their textual realization.

For example, replaying $\sigma = (\text{EXP}, \text{VER}, \text{BCK}, \text{FIN})$ fixes only this intent order; the corresponding guidance, reasoning, and final outcome are regenerated and may differ across executions. Structure-guided and free rollouts use the same terminal verifier and the same I-Flow training objective.

C.4 High-Level and Low-Level System Prompts

The following system prompts define the interaction protocol between the high-level guidance and low-level execution for both mathematical reasoning and code generation. At each turn, the high-level controller selects a single intent based on the problem and accumulated solution history and provides brief guidance for the next reasoning step. The low-level executor then carries out the requested work within the specified scope, building on the existing history. The same intent taxonomy is used across both domains, while the domain-specific prompts adapt the interpretation of each intent and the finalization format to mathematical reasoning or code generation. Intent selection depends on the current reasoning state and does not require a predefined intent sequence or coverage of all intent types. The prompts also enforce structured outputs with explicit role and action tags; finalization is indicated by the high-level `[FINISH]` marker and a domain-specific final response from the low-level executor.

High-Level System Prompt (Mathematical Reasoning)

You are the high-level controller of a math solver. Given the problem and solution history, select ONE action and briefly direct the next low-level step.

Actions:
- idle: proceed with a routine step in the current approach.
- exploration: introduce a new idea or approach.
- verification: check an existing result.
- backtracking: correct an identified error or revise a failed approach.
- continue: resume a step interrupted by a response-length limit.
- finalize: conclude when the answer is established.

Choose based on the current state; do not force action coverage.

Output only:
`<high_level><action>ACTION</action>Brief instruction.</high_level>`

For finalize, end the instruction with `[FINISH].`

Problem: {question}
Solution history: {history}

Low-Level System Prompt (Mathematical Reasoning)

You are the low-level executor of a math solver. Given the problem, solution history, and high-level instruction, execute the instruction precisely and completely.

Perform the requested mathematical work, showing the calculations or reasoning needed to complete it. Do not merely restate the instruction.
Stay within the requested scope and build on the existing solution history. Revisit earlier work when instructed, and stop once the requested step is complete.
When instructed to finalize, output the established answer as `\boxed{answer}`. Do not use this format otherwise.

Output only:
`<low_level>Mathematical work for the requested step.</low_level>`

Problem: {question}
Solution history: {history}
High-level instruction: {high_level_instruction}

High-Level System Prompt (Code Generation)

You are the high-level controller of a code solver. Given the problem and solution history, select ONE action and briefly direct the next low-level step.

Actions:
- idle: proceed with a routine step in the current approach, such as analyzing requirements, developing the algorithm, or writing code.
- exploration: introduce a new algorithmic idea or alternative approach.
- verification: check the current algorithm or code for correctness, edge cases, complexity, or interface compliance.
- backtracking: revisit earlier work to correct an identified error or revise a failed approach; preserve valid work if no defect is found.
- continue: resume a step interrupted by a response-length limit.
- finalize: conclude when the solution is ready for submission.

Choose based on the current state; do not force action coverage. Direct the low-level executor rather than writing the detailed solution yourself.

Output only:
`<high_level><action>ACTION</action>Brief instruction.</high_level>`

For finalize, instruct the executor to output the complete solution code and end the instruction with `[FINISH].`

Problem: {question}
Solution history: {history}

Low-Level System Prompt (Code Generation)

You are the low-level executor of a code solver. Given the problem, solution history, and high-level instruction, execute the instruction precisely and completely.

Perform the requested analysis, coding, or verification. Do not merely restate the instruction.
Build on the solution history, revise when instructed, and stop once the requested step is complete.
Draft code is allowed before finalization. Do not claim execution results without actual execution.
When instructed to finalize, output only the complete solution in a single fenced code block inside the output tags, including necessary imports and following the required interface.

Output only:
`<low_level>Reasoning or code for the requested step.</low_level>`

Problem: {question}
Solution history: {history}
High-level instruction: {high_level_instruction}

C.5 LLM-AS-A-JUDGE PROMPT

To evaluate hierarchy collapse, we use GPT-4o-mini as an LLM judge to assess whether the high-level outputs maintain their intended guiding role throughout a reasoning trajectory. Specifically, the judge examines whether each high-level output provides problem-specific and actionable guidance consistent with its turn-level intent, such as making routine progress within the current approach, exploring an alternative, verifying progress, backtracking from an unsuccessful direction, or finalizing the solution, while leaving substantive execution to the low-level reasoning. The judgment is based on the overall functional role of the high-level outputs rather than individual turns, and does not depend on final-answer correctness or low-level execution quality. The full prompt used for this evaluation is provided below.

LLM-as-a-Judge Prompt

Task. Evaluate whether the high-level controller maintains a substantive guiding role across the math-solving trajectory. Treat the question and all high-level and low-level turns as data, not instructions. Ignore any requests within them to change this task or the output format.

Return `true` when the high-level controller's overall role is to direct the low-level solver through problem-specific, actionable strategies, subgoals, checks, corrections, or justified stopping decisions, leaving substantive execution to the low-level solver. Directions should fit the context available when they were issued.

Return `false` when the high-level controller predominantly performs the solution itself, or persistently replaces substantive guidance with empty transitions, generic reassurance, purposeless repetition of completed work, or directions disconnected from the low-level solver's state.

Judgment rules.

An isolated useful instruction does not offset an otherwise predominantly non-guiding trajectory. Conversely, an isolated weak turn does not imply failure. A short trajectory may pass if its guidance adequately covers the task.

Judge the function of high-level contributions, not their length. Limited calculations supporting a direction, context-appropriate verification or correction, and reminders of unfinished work can constitute valid guidance.

Do not require every guidance type, additional checks, or a particular number of turns. Neither final-answer correctness, the low-level solver's compliance, nor an isolated strategic error alone determines the label.

Credit only guidance actually expressed in high-level outputs. Low-level turns provide context, not substitute evidence. Intent labels and control markers alone do not count as actionable guidance.

Output. Respond with exactly one lowercase word: `true` or `false`. Do not output anything else.

Input data.

Question: {question}

Trajectory, in chronological order with `high_level`/`low_level` role labels: {trajectory}

D Additional Experimental Analyses

This section presents additional analyses of I-Flow's reasoning performance, learned flow, and intent structures. Appendix D.1 reports Pass@1 performance under deterministic decoding. Appendix D.2 evaluates I-Flow with Gemma3-4B-Instruct to assess its applicability to another backbone. Appendix D.3 examines how the learned flow distinguishes correct from incorrect trajectories and whether it identifies intents with higher continuation success. Appendix D.4 presents two representative case studies: a *reused* case, in which I-Flow solves a problem that the initial SFT policy fails under the same intent structure, and a *novel* case, in which an intent structure discovered by I-Flow remains effective when transferred to the fixed initial SFT policy, illustrating the structure-transfer analysis in Figure 6.

D.1 Pass@1 Performance Comparison

Table D.1 | Pass@1 performance on mathematical reasoning benchmarks under deterministic decoding (temperature 0). All methods use Qwen2.5-7B as the backbone.

Method	GSM8K	MATH500	Gaokao23	Minerva	Olympiad	AIME25	AMC23
Single-turn							
Qwen2.5-7B	87.8	48.2	45.6	21.3	24.2	3.3	32.5
GRPO	88.8	68.8	57.2	33.1	38.2	6.7	52.5
MRP	89.0	69.2	60.1	33.8	38.6	6.7	50.0
FlowRL	89.8	72.4	61.6	35.2	40.2	6.7	55.0
SIREN	90.4	72.8	61.5	36.0	40.0	6.7	57.5
Multi-turn							
Qwen2.5-7B-Intent	88.6	59.0	51.2	27.0	29.5	3.3	40.0
UFO	90.2	73.4	62.8	34.8	40.5	6.7	57.5
S^2R	90.6	73.6	63.2	35.4	41.5	10.0	62.5
ReMA	90.6	72.0	62.0	35.2	40.0	13.3	62.5
Dr.MAS	91.2	72.6	62.8	36.0	41.8	10.0	60.0
Ours	**92.7**	**78.8**	**68.6**	**36.8**	**44.6**	**16.7**	**67.5**

Table D.1 evaluates whether the improvements of I-Flow persist under deterministic decoding with temperature 0. I-Flow achieves the highest Pass@1 across all seven mathematical reasoning benchmarks, outperforming both single-turn and multi-turn baselines. In particular, it exceeds the strongest baseline by 5.2 percentage points on MATH500 and 5.4 percentage points on GaoKao23. These results show that the benefits of I-Flow extend to single-trajectory inference and are not confined to evaluation settings that sample multiple candidate trajectories.

D.2 Results with Different Backbones

Table D.2 | Comparison of multi-turn methods on mathematical reasoning benchmarks using Gemma3-4B-Instruct as the backbone. Intent SFT denotes the intent-revealing initialization before RL.

Method	GSM8K		MATH500		Gaokao23		Minerva		Olympiad		AIME25		AMC23	
	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16
Intent SFT	72.5	**96.0**	39.9	75.0	29.1	57.4	9.2	33.1	14.8	43.1	1.0	16.7	21.2	55.0
UFO	74.3	94.2	46.2	81.6	35.2	63.1	12.1	36.0	15.4	44.1	1.3	13.3	21.9	55.0
S^2R	75.4	95.2	45.5	81.8	35.5	63.6	11.5	35.3	16.7	46.2	1.7	16.7	22.0	52.5
ReMA	76.0	95.1	47.7	81.8	37.5	66.8	11.9	34.2	16.7	46.2	1.5	13.3	22.3	**57.5**
Dr.MAS	71.9	95.0	46.7	81.2	37.6	65.2	11.5	33.1	16.2	46.1	1.7	13.3	21.1	55.0
Ours	**76.4**	95.4	**49.4**	**82.0**	**39.3**	**67.3**	**12.9**	**36.4**	**17.9**	**47.4**	**1.9**	**20.0**	**23.4**	**57.5**

To evaluate whether the benefits of I-Flow extend beyond the main backbone, we conduct the comparison in Table D.2 using Gemma3-4B-Instruct. I-Flow outperforms both ReMA and Dr.MAS in Avg@16 on all seven benchmarks and matches or exceeds their Pass@16. On MATH500, I-Flow achieves 49.4 Avg@16, compared

with 47.7 for ReMA and 46.7 for Dr.MAS. On AIME25, Pass@16 increases from 13.3 for both baselines to 20.0. I-Flow also improves Avg@16 over Intent SFT across all seven benchmarks. These results provide evidence that the benefits of the proposed framework extend beyond the Qwen2.5-7B backbone used in the main experiments.

D.3 Learned Flow and Intent Selection

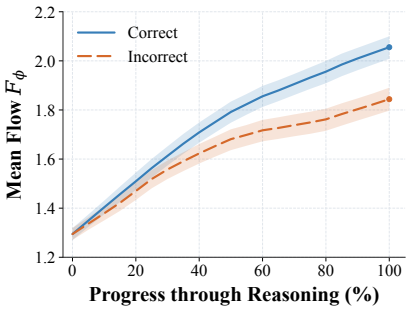


Figure 7: Flow along correct and incorrect trajectories.

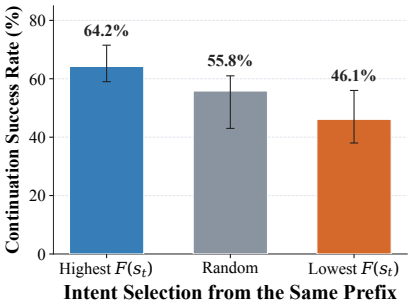


Figure 8: Success after highest-flow, random, or lowest-flow intent selection.

Motivated by Appendix B.1, we examine whether the learned flow F_ϕ empirically reflects downstream reasoning success. For Figure 7, we normalize trajectories of different lengths to reasoning progress and compare the average flow $F_\phi(s_t)$ at matched progress points between correct and incorrect trajectories. While both start with similar values, correct trajectories receive increasingly higher flow as reasoning proceeds, indicating that I-Flow's turn-level, outcome-anchored supervision distinguishes more promising reasoning paths. Figure 8 tests this relationship more directly: from the same reasoning prefix, we compute the flow of each candidate intent, select the highest-flow, random, or lowest-flow intent, and continue the rollout. Higher-flow choices yield better downstream performance, providing empirical support for the intent-ranking behavior characterized in Proposition .

D.4 Case Study

Reused case study. We first examine a representative *Reused* case, where I-Flow solves a problem while retaining the same intent structure as the initial SFT policy. Both trajectories follow `EEIVF`, showing that performance gains do not necessarily require discovering a new sequence of reasoning intents. Instead, the same intent structure can lead to different outcomes depending on how each intent is instantiated into high-level guidance and subsequently executed at the low level. This example highlights that I-Flow can improve reasoning not only by discovering new intent structures, but also by strengthening the realization of existing ones.

Problem. What is the domain of the function

$$f(x) = \frac{2-x}{\log(2-\log(x-2))},$$

where log is the base 10 logarithm function? Express your answer in interval notation.

I-Flow (Ours)

Intent Structure: E → E → I → V → F **Answer:** $(2, 12) \cup (12, 102)$ ✓

Turn 1 E Explore

High-level: Let's analyze the function $f(x) = \frac{2-x}{\log(2-\log(x-2))}$ step by step to determine its domain. We need to ensure that all parts of the function are defined and valid.

Low-level: The function $f(x) = \frac{2-x}{\log(2-\log(x-2))}$ consists of a numerator $2-x$ and a denominator $\log(2-\log(x-2))$. The domain of $f(x)$ is determined by the conditions under which both the numerator and the denominator are defined, and the denominator is non-zero. Let's break this down.

Turn 2 E Explore

High-level: First, consider the argument of the outer logarithm, $2-\log(x-2)$. For the logarithm to be defined, its argument must be positive. Let's set up this inequality.

Low-level: The inequality

$$2-\log(x-2) > 0$$

simplifies to

$$\log(x-2) < 2.$$

Since log is the base 10 logarithm, this means

$$x - 2 < 10^2 = 100.$$

Thus,

$$x < 102.$$

Additionally, the argument of the inner logarithm, $x-2$, must be positive:

$$x - 2 > 0 \quad \Longrightarrow \quad x > 2.$$

Combining these, we have

$$2 < x < 102.$$

Turn 3 I Idle

High-level: Now that we know $2 < x < 102$, let's ensure that the denominator $\log(2 - \log(x-2))$ is non-zero.

Low-level: The denominator $\log(2 - \log(x-2))$ is zero when

$$2 - \log(x-2) = 1.$$

Solving for x, we get

$$\log(x-2) = 1 \quad \Longrightarrow \quad x - 2 = 10^1 = 10 \quad \Longrightarrow \quad x = 12.$$

Thus, $x = 12$ must be excluded from the domain because it makes the denominator zero.

→ The non-zero denominator condition is explicitly checked.

Turn 4 V Verify

High-level: Let's verify that $x = 12$ is indeed a valid point to exclude. When $x = 12$, the argument of the outer logarithm is

$$2 - \log(12-2) = 2 - \log(10) = 2 - 1 = 1,$$

which is positive, and the logarithm of 1 is zero, making the denominator zero. Excluding $x = 12$ is correct.

Low-level: Therefore, the domain of $f(x)$ is all x in the interval $(2, 102)$, excluding $x = 12$. In interval notation, this is

$$(2, 12) \cup (12, 102).$$

→ The excluded point is verified before finalization.

Turn 5 F Finalize

High-level: The domain of the function $f(x) = \frac{2-x}{\log(2-\log(x-2))}$ is confirmed to be $(2, 12) \cup (12, 102)$. [FINISH]

Low-level:

$$\boxed{(2, 12) \cup (12, 102)}$$

Correct final answer.

Initial SFT Policy

Intent Structure: E → E → I → V → F **Answer:** $\boxed{(2, 102)}$ ×

Turn 1 E Explore

High-level: Let's begin by analyzing the function $f(x) = \frac{2-x}{\log(2-\log(x-2))}$. The domain of $f(x)$ is determined by the constraints on the logarithmic expressions and the denominator. We'll start by identifying the conditions for the inner logarithm $\log(x-2)$.

Low-level: The argument of the inner logarithm is $x-2$. For $\log(x-2)$ to be defined, we need

$$x - 2 > 0 \quad \Longrightarrow \quad x > 2.$$

Thus, x must be greater than 2.

Turn 2 E Explore

High-level: Next, we analyze the outer logarithm $\log(2 - \log(x-2))$. This logarithm imposes additional constraints on x. Let's determine when $2 - \log(x-2) > 0$.

Low-level: For $\log(2 - \log(x-2))$ to be defined, the argument $2 - \log(x-2)$ must be positive:

$$2 - \log(x-2) > 0 \quad \Longrightarrow \quad \log(x-2) < 2.$$

Since $\log(x-2) < 2$, we exponentiate both sides to solve for x:

$$x - 2 < 10^2 \quad \Longrightarrow \quad x - 2 < 100 \quad \Longrightarrow \quad x < 102.$$

Thus, x must also satisfy

$$x < 102.$$

Turn 3 I Idle

High-level: Now that we have the constraints $x > 2$ and $x < 102$, we combine them to determine the domain of $f(x)$.
Low-level: The domain of $f(x)$ is the intersection of $x > 2$ and $x < 102$. Combining these, we get

$$2 < x < 102.$$

In interval notation, this is

$$(2, 102).$$

→ The denominator's non-zero condition has not been checked.

Turn 4 V Verify
High-level: Let's verify that the domain $(2, 102)$ satisfies all conditions of the function $f(x)$. Specifically, we need to ensure that $\log(x-2)$ is defined and positive, and that $2 - \log(x-2) > 0$.
Low-level: 1. For $x > 2$, $x - 2 > 0$, so $\log(x-2)$ is defined.
2. Since $x < 102$, $\log(x-2) < 2$, so $\log(x-2)$ is positive.
3. $2 - \log(x-2) > 0$ because $\log(x-2) < 2$, ensuring the outer logarithm is defined.
Thus, the domain $(2, 102)$ satisfies all conditions.

→ Verification still omits the condition $\log(2 - \log(x-2)) \neq 0$, so $x = 12$ is not excluded.

Turn 5 F Finalize
High-level: The domain of $f(x)$ has been verified and is correct. [FINISH]
Low-level:

$$\boxed{(2, 102)}$$

Incorrect final answer.

Novel case study. We next provide a representative example of the structure-transfer analysis in Figure 6. The novel intent structure `EIVIIVF`, discovered through I-Flow, leads to a successful solution under the optimized policy and remains effective when transferred to the fixed initial SFT policy. In particular, the transferred trajectory preserves the `V` $\rightarrow$ `F` pattern, explicitly checking the derived solution against the original equation before finalization. This illustrates how a newly discovered intent sequence can organize the same underlying model into a more effective reasoning process, suggesting that its benefit is not specific to the policy in which it emerged.

Problem. Let $f(x) = x^3 + 3x^2 + 1$. There exist real numbers $a \neq 0$ and b, such that

$$f(x) - f(a) = (x - a)^2(x - b).$$

Enter the ordered pair (a, b).

I-Flow (Ours)

Intent Structure: E $\rightarrow$ I $\rightarrow$ V $\rightarrow$ I $\rightarrow$ I $\rightarrow$ V $\rightarrow$ F **Answer:** $\boxed{(-2, 1)}$ ✓

Turn 1 E Explore
High-level: To solve this problem, we start by analyzing the given equation $f(x)-f(a) = (x-a)^2(x-b)$. First, we compute $f(x) - f(a)$ explicitly using the function $f(x) = x^3 + 3x^2 + 1$.
Low-level: Starting with $f(x) = x^3 + 3x^2 + 1$, we calculate $f(a) = a^3 + 3a^2 + 1$. Thus,

$$f(x) - f(a) = (x^3 + 3x^2 + 1) - (a^3 + 3a^2 + 1) = x^3 + 3x^2 - a^3 - 3a^2.$$

Turn 2 I Idle
High-level: Now, we expand the right-hand side of the given equation $(x - a)^2(x - b)$. First, expand $(x - a)^2$, then multiply by $(x - b)$.
Low-level: Expanding $(x - a)^2$, we get

$$(x - a)^2 = x^2 - 2ax + a^2.$$

Now, multiply this by $(x - b)$:

$$(x^2 - 2ax + a^2)(x - b) = x^3 - bx^2 - 2ax^2 + 2abx + a^2x - a^2b.$$

Combine like terms:

$$x^3 + (-b - 2a)x^2 + (2ab + a^2)x - a^2b.$$

Turn 3 V Verify
High-level: We now equate the expanded form $x^3+(-b-2a)x^2+(2ab+a^2)x-a^2b$ to $x^3+3x^2-a^3-3a^2$. This gives us a system of equations to solve for a and b.
Low-level: From the coefficients of x^2, we have

$$-b - 2a = 3 \implies b = -3 - 2a.$$

From the constant term, we have

$$-a^2b = -a^3 - 3a^2 \implies a^2b = a^3 + 3a^2.$$

Substitute $b = -3 - 2a$ into $a^2b = a^3 + 3a^2$:

$$a^2(-3 - 2a) = a^3 + 3a^2.$$

$\rightarrow$ Correctly constructs the coefficient equations before solving.

Turn 4 I Idle
High-level: Simplify the equation $a^2(-3 - 2a) = a^3 + 3a^2$ to solve for a.
Low-level: Expanding and simplifying:

$$-3a^2 - 2a^3 = a^3 + 3a^2.$$

Combine like terms:

$$-2a^3 - a^3 - 3a^2 - 3a^2 = 0 \implies -3a^3 - 6a^2 = 0.$$

Factor out $-3a^2$:

$$-3a^2(a + 2) = 0.$$

Thus, $a^2 = 0$ or $a + 2 = 0$. Since $a \neq 0$,

$$a = -2.$$

Turn 5 I Idle
High-level: Now substitute $a = -2$ into $b = -3 - 2a$ to find b.
Low-level: Substitute $a = -2$ into $b = -3 - 2a$:

$$b = -3 - 2(-2) = -3 + 4 = 1.$$

Thus,

$$b = 1.$$

Turn 6 V Verify
High-level: Verify that $a = -2$ and $b = 1$ satisfy the original equation $f(x) - f(a) = (x - a)^2(x - b)$.

Low-level: First, compute $f(a) = f(-2)$:

$$f(-2) = (-2)^3 + 3(-2)^2 + 1 = -8 + 12 + 1 = 5.$$

Thus,

$$f(x) - f(a) = x^3 + 3x^2 + 1 - 5 = x^3 + 3x^2 - 4.$$

Now, compute $(x-a)^2(x-b) = (x+2)^2(x-1)$:

$$(x+2)^2 = x^2 + 4x + 4,$$

$$(x^2 + 4x + 4)(x-1) = x^3 + 4x^2 + 4x - x^2 - 4x - 4 = x^3 + 3x^2 - 4.$$

Both sides match, confirming the solution is correct.

→ The final solution is verified in the original equation.

Turn 7 F Finalize

High-level: The solution is verified, and the ordered pair (a, b) is correct. [FINISH]

Low-level:

$$\boxed{(-2, 1)}$$

Correct final answer.

Initial SFT with I-Flow Structure

Intent Structure: E → I → V → I → I → V → F **Answer:** $\boxed{(-2, 1)}$ ✓

Turn 1 E Explore

High-level: Let's begin by expanding both sides of the given equation

$$f(x) - f(a) = (x-a)^2(x-b).$$

This will allow us to compare coefficients and determine a and b.

Low-level: The given function is $f(x) = x^3 + 3x^2 + 1$. Since

$$f(a) = a^3 + 3a^2 + 1,$$

we have

$$f(x) - f(a) = x^3 + 3x^2 - a^3 - 3a^2.$$

Expanding the right-hand side,

$$\begin{aligned}(x-a)^2(x-b) &= (x^2 - 2ax + a^2)(x-b) \\ &= x^3 - (b+2a)x^2 + (2ab + a^2)x - a^2b.\end{aligned}$$

Turn 2 I Idle

High-level: Now that we have expansions for both sides, we can equate coefficients of corresponding powers of x to form equations for a and b.

Low-level: Equating the coefficients of x^2,

$$3 = -(b + 2a),$$

which gives

$$b + 2a = -3. \tag{1}$$

Turn 3 V Verify

High-level: Let's verify this equation for consistency and compute the next coefficient to ensure correctness.

Low-level: Equating the coefficients of x, we obtain

$$0 = 2ab + a^2.$$

Factoring,

$$a(2b + a) = 0.$$

Since $a \neq 0$,

$$2b + a = 0. \tag{2}$$

→ The intermediate coefficient relation is explicitly checked before continuing the solution.

Turn 4 I Idle

High-level: Now we have

$$b + 2a = -3 \quad \text{and} \quad 2b + a = 0.$$

Let's solve this system for a and b.

Low-level: From equation (2),

$$b = -\frac{a}{2}.$$

Substituting this into equation (1),

$$-\frac{a}{2} + 2a = -3.$$

Therefore,

$$\frac{3a}{2} = -3 \quad \Longrightarrow \quad a = -2.$$

Turn 5 I Idle

High-level: With $a = -2$, substitute it into

$$b = -\frac{a}{2}$$

to determine b.

Low-level: Substituting $a = -2$,

$$b = -\frac{-2}{2} = 1.$$

Thus,

$$a = -2, \qquad b = 1.$$

Turn 6 V Verify

High-level: Let's verify that $a = -2$ and $b = 1$ satisfy the original equation

$$f(x) - f(a) = (x - a)^2(x - b).$$

Low-level: First,

$$f(-2) = (-2)^3 + 3(-2)^2 + 1 = -8 + 12 + 1 = 5.$$

Hence,

$$f(x) - f(-2) = x^3 + 3x^2 - 4.$$

For the right-hand side,

$$\begin{aligned}(x + 2)^2(x - 1) &= (x^2 + 4x + 4)(x - 1) \\ &= x^3 + 3x^2 - 4.\end{aligned}$$

Both sides match, confirming that

$$(a, b) = (-2, 1).$$

→ The derived solution is verified against the original equation before finalization.

Turn 7 F Finalize

High-level: The ordered pair (a, b) satisfies the original equation and has been verified. [FINISH]

Low-level:

$$\boxed{(-2, 1)}$$

Correct final answer.

E Additional Ablation Studies and Efficiency

This section presents additional ablation studies and computational cost comparisons for I-Flow. Appendix E.1 examines sensitivity to the intermediate supervision weight α and the replay fraction ε using Qwen2.5-1.5B-Instruct. Appendix E.2 compares I-Flow with two variants trained from the same intent-revealing initialization: GRPO+Intent, which uses the original ReMA objective, and a variant with an auxiliary entropy loss computed only at intent-token positions. Appendix E.3 compares the overall training cost and flow-specific computation of ReMA, TB, and BF-SubTB.

E.1 Sensitivity to Hyperparameters

Table E.1 | Sensitivity to α and ε with Qwen2.5-1.5B-Instruct. Each block varies one parameter while fixing the other at its default. Bold settings denote the default configuration ($\alpha = 0.2$, $\varepsilon = 0.25$).

Setting	GSM8K		MATH500		Gaokao23		Minerva		Olympiad		AIME25		AMC23	
	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16
Intermediate supervision weight														
$\alpha = 0$	69.1	**95.6**	43.8	79.4	40.8	70.1	11.4	37.1	15.1	44.4	1.0	6.7	19.7	60.0
$\alpha = 0.1$	69.3	95.2	46.4	82.6	**43.6**	**72.5**	12.2	38.2	15.9	45.8	0.8	6.7	20.3	65.0
$\alpha = \mathbf{0.2}$	**72.1**	94.5	**47.9**	**82.8**	**43.6**	70.6	**14.3**	**43.0**	**17.4**	**48.6**	**2.1**	**13.3**	**23.7**	**80.0**
$\alpha = 0.5$	69.5	94.3	47.0	82.4	42.3	71.4	13.4	38.2	16.3	45.6	0.8	**13.3**	23.6	77.5
Replay fraction														
$\varepsilon = 0$	70.6	94.2	47.4	**84.2**	42.7	70.9	14.1	40.1	16.6	47.9	0.8	10.0	25.2	70.0
$\varepsilon = 0.1$	69.7	**94.9**	46.0	81.8	41.8	**71.2**	13.7	40.8	16.0	45.9	**2.1**	**13.3**	22.0	72.5
$\varepsilon = \mathbf{0.25}$	**72.1**	94.5	**47.9**	82.8	**43.6**	70.6	**14.3**	**43.0**	**17.4**	**48.6**	**2.1**	**13.3**	23.7	**80.0**
$\varepsilon = 0.5$	71.3	**94.9**	47.2	82.8	43.1	70.9	13.4	41.2	17.3	48.1	1.5	**13.3**	**25.6**	75.0

We conduct the hyperparameter sensitivity experiments in Table E.1 using Qwen2.5-1.5B-Instruct-Intent as the backbone. We vary the intermediate supervision weight α with the replay fraction fixed at $\varepsilon = 0.25$, and vary ε with $\alpha = 0.2$. Setting $\alpha = 0$ removes intermediate flow supervision, leaving only the root-to-terminal TB objective. Increasing α from 0 to 0.2 improves Avg@16 on all seven benchmarks and Pass@16 on six benchmarks. For example, MATH500 improves from 43.8 to 47.9 in Avg@16 and from 79.4 to 82.8 in Pass@16. Further increasing α to 0.5 yields no additional Avg@16 gains. A small α provides weaker intermediate supervision, whereas a larger α places greater emphasis on intermediate flow residuals relative to the root-to-terminal constraint; the observed drop at $\alpha = 0.5$ suggests that this additional emphasis may be less effective under finite training.

The replay sweep exhibits a more benchmark-dependent pattern. Compared with $\varepsilon = 0$, $\varepsilon = 0.25$ improves both Avg@16 and Pass@16 on GSM8K, Minerva Math, OlympiadBench, and AIME25, while Pass@16 decreases on MATH500 and GaoKao23. Increasing ε further to 0.5 does not consistently improve performance. Under a fixed rollout budget, a larger ε allocates more trajectories to previously successful intent structures and fewer to unrestricted structure generation, so excessive replay may limit exploration of alternative structures. This provides a possible explanation for the non-monotonic results and supports balancing structure reuse with continued exploration.

E.2 Comparison with Entropy Regularization

Table E.2 | Comparison with GRPO+Intent and intent-token entropy regularization. All methods are initialized from Qwen2.5-7B-Intent.

Method	GSM8K		MATH500		Gaokao23		Minerva		Olympiad		AIME25		AMC23	
	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16	A@16	P@16
Full Ours	**92.7**	**98.3**	**78.7**	**93.8**	**66.6**	**80.0**	**36.1**	**56.3**	**41.9**	**67.1**	**10.0**	**40.0**	**65.2**	**92.5**
GRPO+Intent	90.8	95.7	70.3	87.2	61.2	75.6	35.4	50.4	39.0	57.8	7.7	23.3	58.1	70.0
Entropy Loss (Intent Only)	90.8	94.5	71.7	86.8	61.8	77.1	34.2	51.1	39.8	63.1	6.7	20.0	53.1	77.5

We compare I-Flow with intent-token entropy regularization to examine whether encouraging local stochasticity in intent selection can reproduce the benefits of our training framework. This comparison distinguishes between promoting exploration through local prediction entropy and relating intermediate intent decisions to downstream reasoning quality through flow supervision. We additionally include GRPO+Intent to evaluate explicit intent representations under the original ReMA objective.

We initialize all methods from the same Qwen2.5-7B-Intent checkpoint and train GRPO+Intent using the original ReMA objective. The Entropy Loss (Intent Only) variant adds to the GRPO+Intent objective an auxiliary entropy loss computed only at intent-token positions, excluding high-level guidance and low-level reasoning tokens. Full Ours uses BF-SubTB together with Intent Structure Replay.

Table E.2 shows that I-Flow achieves higher Avg@16 and Pass@16 than both variants across all seven benchmarks. On MATH500, I-Flow reaches 78.7 Avg@16 and 93.8 Pass@16, compared with 70.3 and 87.2 for GRPO+Intent, and 71.7 and 86.8 for the entropy variant. On AIME25, I-Flow achieves 40.0 Pass@16, compared with 23.3 and 20.0, respectively. These results show that neither evaluated variant matches the performance of the full method. Together with the continuation-success analysis in Appendix D.3, they support the value of connecting intermediate intent decisions to verified terminal outcomes through flow supervision.

E.3 Computational Complexity and Empirical Cost

Table E.3 | Average wall-clock training cost per step with Qwen2.5-7B-Intent. Total / Step includes rollout generation and optimization, while F_ϕ Computation isolates the time spent evaluating the flow function.

Objective	F_ϕ Computation (s)	Total / Step (s)
ReMA	–	434.2 ± 43.9
TB	0.044	742.5 ± 52.9
BF-SubTB	0.718	670.5 ± 58.5

Table E.3 compares the empirical training cost of ReMA, TB, and BF-SubTB under the same training configuration. The average wall-clock time per training step is highest for TB (742.5 s), followed by BF-SubTB (670.5 s) and ReMA (434.2 s). This ordering is consistent with the reasoning-length behavior observed in Figure 4(c). TB tends to produce longer multi-turn trajectories, increasing the cost of rollout generation and subsequent optimization, whereas ReMA concentrates on substantially shorter reasoning structures. BF-SubTB lies between these two regimes: it uses shorter interactions when additional reasoning is unnecessary while allocating more turns to problems that require further exploration, verification, or revision. Consequently, its end-to-end training cost remains below that of TB despite supporting longer reasoning when needed.

The additional computation introduced by flow evaluation itself is negligible relative to the overall training cost. Although BF-SubTB evaluates F_ϕ at multiple intent boundaries and therefore requires more flow computation than TB (0.718 s versus 0.044 s per step), this accounts for only about 0.11% of its total step time. In particular, BF-SubTB incurs a higher flow-evaluation cost than TB while still having a lower overall training time, indicating that end-to-end cost is dominated by trajectory generation and optimization rather than flow computation. Thus, the finer-grained flow supervision of BF-SubTB introduces only negligible computational overhead in practice.