The Two-Stage Decision-Sampling Hypothesis: Understanding the Emergence of Self-Reflection in RL-Trained LLMs

Anonymous ACL submission

Abstract

Self-reflection capabilities emerge in Large Language Models after RL post-training, with multi-turn RL achieving substantial gains over SFT counterparts. Yet the mechanism of how a unified optimization objective gives rise to functionally distinct capabilities of generating solutions and evaluating when to revise them remains opaque. To address this question, we introduce the Gradient Attribution Property to characterize how reward gradients distribute across policy components, formalized through the Two-Stage Decision-Sampling (DS) Hypothesis, which decomposes the policy into sampling (π_{sample}) for generation and decision (π_d) for verification. We prove that surrogate rewards exhibit Balanced Gradient Attribution, while SFT and KL penalties exhibit Unbalanced Gradient Attribution, with length-weighting creating asymmetric regularization that constrains π_{sample} while leaving π_d under-optimized, providing an theoretical explanation of why RL succeeds where SFT fails. We also empirically validate our theoretical predictions on arithmetic reasoning demonstrates that RL's superior generalization stems primarily from improved decision-making (π_d) rather than sampling capabilities, providing a first-principles mechanistic explanation for self-correction in thinking models.

1 Introduction

Self-reflection capabilities—the ability to verify reasoning, detect errors, and revise incorrect answers—emerge spontaneously in large language models after RL post-training (DeepSeek-AI et al., 2025; OpenAI et al., 2024; Bercovich et al., 2025; Zhao et al., 2024). This emergent behavior correlates strongly with substantial performance improvements, particularly in mathematical reasoning. Yet despite widely documented gains, the mechanism by which RL produces these qualitatively different capabilities remains theoretically opaque. It is unclear how a unified reinforcement learning objective gives rise to functionally distinct abilities—generating candidate solutions versus evaluating when to accept or revise them—and why RL succeeds where supervised fine-tuning (SFT) consistently fails.

While existing literature extensively records what occurs—the emergence of verification and revision behaviors—it lacks an anatomical explanation of how the training objective fundamentally alters the model's policy. Specifically, it remains unclear which optimization processes drive the functional separation required for self-correction. To bridge this gap, we must examine the underlying gradient dynamics that govern this behavioral bifurcation.

In this paper, we view learning to self-reflect as learning to multitask with a shared policy function. Under this framework, we introduce the Gradient Attribution Property to characterize how reward gradients distribute across policy components. We formalize this through the Two-Stage Decision-Sampling (DS) Hypothesis, which decomposes the model's unified policy into a sampling policy π_{sample} for content generation and a decision policy π_d for verification and stopping. When a model develops self-reflection, it primarily learns to improve π_d(judgment about when to trust versus revise outputs) rather than (or in addition to) improving π_{sample}. Transforming the question about "emergent self-reflection" into a question about gradient flow: how does the RL objective differentially update these policy components?

We prove that different reward structures induce fundamentally different learning dynamics. Surrogate rewards exhibit balanced gradient attribution: variations in the reward signal map cleanly to whichever policy component was responsible, creating symmetric learning pressure on both π_{sample} and π_d. In contrast, KL-divergence penalties exhibit imbalanced gradient attribution:

length-weighting in token-level calculations creates asymmetric regularization that heavily constrains π_{sample} while leaving π_d relatively unconstrained. This explains why RL succeeds—its objective mathematically favors learning better π_d, while SFT, which resembling a KL-divergence objective without countervailing reward, systematically fails to develop genuine self-reflection.

We make several contributions. First, we formalize the DS-Hypothesis and mathematically characterize when gradient signals can be cleanly attributed to specific policy components. Second, we provide a first-principles mechanistic account of why RL algorithms like GRPO induce self-reflection while SFT does not. Third, we empirically validate the framework on arithmetic reasoning tasks, demonstrating that: (i) the framework accurately predicts model performance; (ii) RL improves π_d more than π_{sample}; and (iii) out-of-distribution generalization is primarily limited by π_{sample}. Finally, we explain documented phenomena including the "echoing effect"[1] and why reflection-rich SFT data improves first-answer accuracy without enabling genuine self-correction.

The remainder proceeds as follows. Section 2 reviews policy gradient methods, mathematical reasoning generalization, and the evolution of self-correction. Section 3 develops the theoretical framework. Section 4 presents empirical validation. Section 5 applies insights to explain SFT's limitations. Section 6 concludes.

2 Related Work

Our work lies at the intersection of three research streams: the control-theoretic foundations of policy gradients, mathematical generalization in Transformers, and the evolution of self-correction from prompting to learned RL behaviors. To develop the Gradient Attribution Property framework: we borrow analytical tools for decomposing reward gradients from policy gradient theory; we adopt arithmetic tasks as a controlled testbed from the mathematical reasoning literature. Then we take the empirical puzzle from the self-correction literature to motivate the setting of the DS-Hypothesis and showed that our theoretical framework can theoretically explain these puzzles. Our contribution is showing that gradient attribution properties of training objectives determine whether models learn genuine decision-making or merely imitate reflective patterns. We review each literature stream below.

2.1 Foundations and Practices of Policy Optimization Methods

The transition from supervised imitation to reinforcement learning in reasoning models rests fundamentally on policy gradient theory. (Sutton et al., 1999a) introduced *options*—temporally extended actions within semi-Markov decision processes—providing theoretical justification for treating multi-token Chain of Thought as optimizable sequences rather than mere predictions; (Agarwal et al., 2020) reviews the theory and convergence properties.[2] Policy gradient methods have been widely applied in LLM post-training (Ouyang et al., 2022; Stiennon et al., 2020), with prominent algorithms including TRPO (Schulman et al., 2017a), PPO (Schulman et al., 2017b), and DPO (Rafailov et al., 2024).[3] Most relevant to our setting, (Shao et al., 2024) introduced GRPO, which eliminates value networks by normalizing rewards within group samples, enabling thinking models with emergent self-reflection (DeepSeek-AI et al., 2025).[4]

2.2 Mathematical Reasoning and Generalization

Arithmetic reasoning serves as a rigorous testbed for Transformer generalization. (Nogueira et al., 2021) and (Anil et al., 2022) established that standard Transformers, despite scale, exhibit catastrophic length extrapolation failures, suggesting they learn surface heuristics rather than robust algorithms. (Lee et al., 2023) challenged the necessity of scale, showing small models can master arithmetic via data formatting (scratchpads, reverse-order generation), with sharp phase transitions indicative of "grokking." (Xu et al., 2025) formalized these mechanics through a unified framework linking generalization to architecture-task symmetry alignment. Beyond simple arithmetic, benchmarks like GSM8K, MATH, and MetaMathQA are widely

[1] (Kang et al., 2025) found that SFT-trained thinking models never discriminate between correct and incorrect answers; post-reflection answers largely echo pre-reflection ones.

[2] We do not address convergence here; this motivates our single-dimension demonstration.

[3] TRPO constrains updates via KL-divergence; PPO simplifies this through clipping and has become standard for RLHF.

[4] Our theoretical analysis abstracts away clipping and trust-region constraints to preserve tractability. Although GRPO is closest to our setting, our framework generalizes across policy-based algorithms rather than formalizing any specific one.

used to measure LLM reasoning ability (Cobbe et al., 2021; Hendrycks et al., 2021; Yu et al., 2024). We adopt mathematical reasoning for its clarity in evaluating learning effects.[5]

2.3 RL-Reflection and Self-Correction

Self-correction capabilities have evolved from prompt engineering to learned behaviors. While (Wei et al., 2022) and (Feng et al., 2023) established the computational necessity of intermediate reasoning steps, critical surveys by (Kamoi et al., 2024) and (Huang et al., 2024) revealed that "intrinsic self-correction" in SFT models is often illusory or Oracle-dependent. (Kumar et al., 2024) (SCoRe) identified SFT's distribution mismatch—training on others' mistakes versus correcting one's own—and proposed multi-turn RL on self-generated traces. Subsequent frameworks including Self-Rewarding Correction (Xiong et al., 2025) and PAG (Jiang et al., 2025) unify solver and critic into a single policy. Most relevant, (Zhao et al., 2025) and (Ma et al., 2025) explicitly decompose policies into answer generation and verification, demonstrating performance gains from this breakdown. We focus instead on the *implicit* policy decomposition driving emergent self-reflection in models like DeepSeek-R1.

3 Formalization of The Decision-Sampling Hypothesis

We develop a formal framework explaining why RL produces self-reflection while SFT fails. The key concept is the **gradient attribution property**, characterizing how reward gradients distribute across policy components. When a model performs multiple functions through a single policy, balanced gradient attribution enables learning all functions effectively, while imbalanced attribution causes some functions to be learned ineffectively. We model an LLM solving query Q as a sequential decision process where at each iteration $k \geq 1$, the model: (i) samples a candidate answer A_k with reasoning T_k, then (ii) decides whether to STOP or RESAMPLE. We prove that surrogate rewards[6] exhibit balanced attribution while KL-penalties exhibit imbalanced attribution—explaining why SFT-trained models lack genuine self-reflection.

[5] Ideally, our framework would apply more generally; developing valid measurements in other domains remains challenging.

[6] We analyze simple surrogate rewards for tractability; practice uses clipped variants for stability.

Section 3.1 introduces our formal setting; Section 3.2 defines gradient attribution; Section 3.3 analyzes both reward components. Section 5 applies these results to SFT behavior.

3.1 Settings and Preliminary

State Space The state at step k is $s_k = (Q, A_k, T_k)$, encoding the query and the model's current candidate solution.

Policy Decomposition The overall policy π_θ decomposes into two components:

- Sampling policy $\pi_{\text{sample}}(\cdot|s_{k-1};\theta)$: distribution over (A_k, T_k) given context
- Decision policy $\pi_d(\cdot|s_k;\theta)$: distribution over STOP, RESAMPLE given current state

Trajectory Probability Factorization: A trajectory τ of length T consists of a sequence of samples and decisions:

$$\tau = (A_1, T_1, \text{RESAMPLE}, \ldots, A_T, T_T, \text{STOP})$$

Lemma 1. *The probability of trajectory τ under policy π_θ factorizes as:*

$$P(\tau|Q;\theta) = \left[\prod_{k=1}^{T} \pi_{sample}(A_k, T_k|s_{k-1};\theta)\right] \cdot \left[\prod_{k=1}^{T-1} \pi_d(R|s_k;\theta)\right] \cdot \pi_d(S|s_T;\theta)$$

Corollary 1. *The log-probability gradient separates into sampling and decision components:*

$$\nabla_\theta \log P(\tau|Q;\theta) = \sum_{k=1}^{T} \nabla_\theta \log \pi_{sample}(A_k, T_k|s_{k-1}) + \sum_{k=0}^{T} \nabla_\theta \log \pi_d(a_k|s_k)$$

where $a_k \in \{RESAMPLE, STOP\}$ denotes the decision at step k.

Reward Function: For query Q with ground truth answer A_Q^*, the reward of trajectory τ ending at step T is $R(\tau) = \mathbb{I}(A_T = A_Q^*)$. Thus, for a given policy π_θ and reward function R, define:

$$Q_R^\pi(s,a) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=t}^{\infty} \gamma^{k-t} R_k \,\middle|\, s_t = s, a_t = a\right]$$

where $\gamma \in (0, 1]$ is the discount factor.

Lemma 2. *The gradient of the expected return decomposes as:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_{sample}(a'_t|s_{t-1})Q^\pi_{sample} + \nabla_\theta \log \pi_d(a''_t|s_t)Q^\pi_d\right)\right]$$

where $a'_t = (A_t, T_t)$ denotes sampling actions and $a''_t \in \{RESAMPLE, STOP\}$ denotes decision actions.

3.2 Gradient Attribution Property

We formalize gradient attribution through the information structure of Q-values.

Definition 3.1 (Gradient Attribution Property)**.** Consider a reward function R and the induced Q-values $Q^\pi_{\text{sample}}(s, a')$ and $Q^\pi_d(s, a'')$ under policy π_θ.

A reward R exhibits **balanced gradient attribution**[7] if the Q-values admit a decomposition:

$$Q^\pi_{\text{sample}}(s_{k-1}, a'_k) = f_{\text{sample}}(s_{k-1}, a'_k, \Phi(s_k))$$

$$Q^\pi_d(s_k, a''_k) = f_d(s_k, a''_k, \Phi(s_{k+1}))$$

where $\Phi : \mathcal{S} \to \mathbb{R}$ is a scale-invariant sufficient statistic and the weighting functions f_{sample}, f_d are of comparable magnitude: $f_{\text{sample}} = \Theta(f_d)$

A reward exhibits **unbalanced gradient attribution**[8] if the Q-values decompose as:

$$Q^\pi_{\text{sample}}(s_{k-1}, a'_k) = r_k^{\text{sample}} + \gamma \cdot V^\pi(s_k)$$

$$Q^\pi_d(s_k, a''_k) = r_k^{\text{decision}} + \gamma \cdot V^\pi(s_{k+1})$$

where the immediate reward components satisfy $|r_k^{\text{sample}}| = \omega(|r_k^{\text{decision}}|)$ systematically (i.e., scale-separated by more than a constant factor across typical trajectories).

Remark (balanced attribution). In the case of balanced attribution, we can identify $\Phi(s) = V^\pi(s)$ as the state value function. This is the standard Bellman decomposition:

$$Q^\pi(s, a) = \mathbb{E}[R(s, a) + \gamma V^\pi(s')|s, a]$$

The key is that the *same* V^π appears in the decomposition for both policy components.

Remark (When unbalanced attribution arise?). Consider a reward structure where:

- The immediate reward for sampling depends on sequence length: $r_{\text{sample}} \sim O(L_k)$
- The immediate reward for decisions is scalar: $r_d \sim O(1)$
- Future value recursively depends on these asymmetric immediate rewards

Then Φ_{sample} must encode "accumulated future sampling costs" while Φ_d encodes "accumulated future decision costs", which are fundamentally different scales. This is precisely what happens with the KL penalty, as we show in the section 3.3.

Remark. Operational Identification of Decision Actions he decomposition into π_{sample} and π_d is an analytical abstraction—the model architecturally remains a single next-token predictor. However, decision actions can be operationalized through identifiable proxy tokens that signal verification or revision intent. In our experiments, we detect decision boundaries via lexical markers including "I'll go back and check", "Let me recompute", "I made a mistake" and similar verification phrases. The token sequence following such markers until the next candidate answer constitutes a decision action, while extended generation of reasoning and answers constitutes sampling actions. This operationalization enables the calibration reported in Section 4.2: we estimate $p_d|C$ and $p_d|W$ by observing stopping and resampling behavior conditional on answer correctness, where "resampling" is identified by the presence of revision markers followed by a new solution attempt

3.3 Gradient Attribution of Surrogate Reward and KL Penalty

In this subsection, we analyze the gradient attribution properties of two fundamental reward functions used in RLM training.

[7] *Interpretation: Balanced attribution means both Q-values can be expressed in terms of the same information about future rewards (the sufficient statistic Φ), just evaluated at different states along the trajectory. This allows the unified network to learn a consistent representation of "future value" that both π_{sample} and π_d can use.*

[8] *Unbalanced attribution means the Q-values require different information about the future (Φ_{sample} vs Φ_d). The unified network cannot learn a single coherent representation of future value—attempting to do so leads to "mismeshing" the gradient signals, where updates intended for one conceptual function interfere with learning the other.*

3.3.1 Simple Surrogate Reward Has Balanced Gradient Attribution

The simple surrogate reward with advantage A_i for trajectory τ_i is:

$$L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\text{old}}} \left[\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)} A_i \right]$$

where A_i is the group-relative advantage measuring whether trajectory τ_i is better or worse than the average trajectory for query Q_i.

Theorem 3.1. *For the surrogate reward objective with trajectory-level advantage A_i, the Q-values satisfy:*

$$Q^{\pi,\textit{reward}}_{\textit{sample}}(s_{k-1}, a'_k) = \gamma^{\sum_{j=k}^{T} \textit{len}(A_j,T_j)} \cdot A_i$$

$$Q^{\pi,\textit{reward}}_{d}(s_k, a''_k) = \gamma^{\sum_{j=k}^{T} \textit{len}(A_j,T_j)} \cdot A_i$$

*The sufficient statistic $\Phi(s_k) = \gamma^{\sum_{j=k}^{T} \textit{len}(A_j,T_j)} A_i$ is **scale-invariant**: it enters both Q-values as an identical multiplicative factor. Consequently, the advantage A_i weights gradient contributions to $\pi_{\textit{sample}}$ and π_d symmetrically.*

Proof Sketch. The advantage A_i is a trajectory-level scalar measuring overall quality. From the policy gradient theorem, the gradient is:

$$\nabla_\theta \mathcal{L}_{\text{reward}} = \mathbb{E}_{\tau_i} \left[A_i \sum_{k=1}^{T} \nabla_\theta \log \pi_{\text{sample}}(\cdot) + A_i \sum_{k=0}^{T} \nabla_\theta \log \pi_d(\cdot) \right]$$

The advantage multiplies both gradient components equally. Since both π_{sample} and π_d contributed to producing the trajectory that received advantage A_i, and the advantage reflects the *coupled* outcome (correct answer + appropriate stopping), both Q-values equal the length-discounted advantage. The key observation is that the advantage A_i is a trajectory-level scalar that does not decompose into action-specific components. From the policy gradient theorem, the advantage multiplies both gradient sums identically. There is no action-specific immediate reward that could create magnitude asymmetry—both policy components receive gradient signal proportional to their log-probability scores weighted by the *same* scalar A_i. This is the formal sense in which attribution is "balanced." See Appendix A.2 for complete proof. □

Interpretation. The sufficient statistic $\Phi = \gamma^{\sum \text{len}(\cdot)} A_i$ encodes: "this trajectory will yield advantage A_i at the end, discounted by the temporal distance." Both π_{sample} and π_d use this *same* information when evaluating their actions. This enables coherent learning in the unified network.

3.3.2 KL Penalty: Unbalanced Gradient Attribution

In RLHF and GRPO, the KL divergence penalty regularizes the learned policy π_θ against a reference policy π_{ref} (typically the SFT model). Following standard practice, we consider the token-level KL penalty added to the reward:[9]

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) - w \sum_{t=1}^{|\tau|} \log \frac{\pi_\theta(a_t|s_t)}{\pi_{ref}(a_t|s_t)} \right]$$

where $w \geq 0$ controls the regularization strength.

Under our policy decomposition, the per-trajectory KL penalty decomposes as:

$$D_{KL}^{\text{token}}(\tau) = \underbrace{\sum_{k=1}^{T} \sum_{j=1}^{L_k} \log \frac{\pi_{\text{sample},\theta}(\text{token}_{k,j} \mid \cdot)}{\pi_{\text{sample,ref}}(\text{token}_{k,j} \mid \cdot)}}_{\text{Sampling KL: } \sum_k L_k \text{ terms}} + \underbrace{\sum_{k=1}^{T} \log \frac{\pi_{d,\theta}(a_k \mid s_k)}{\pi_{d,\text{ref}}(a_k \mid s_k)}}_{\text{Decision KL: } T \text{ terms}}$$

The structural asymmetry is immediate: sampling actions contribute $\sum_{k=1}^{T} L_k$ terms while decision actions contribute only T terms. Since typical reasoning traces have $L_k \gg 1$ (hundreds of tokens per attempt), the sampling component dominates.

Define the immediate KL penalties for each action type:

$$d_k^{\text{sample}} = \sum_{j=1}^{L_k} \log \frac{\pi_{\text{sample},\theta}(\text{token}_{k,j} \mid \cdot)}{\pi_{\text{sample,ref}}(\text{token}_{k,j} \mid \cdot)} \sim O(L_k)$$

$$d_k^{\text{decision}} = \log \frac{\pi_{d,\theta}(a_k \mid s_k)}{\pi_{d,\text{ref}}(a_k \mid s_k)} \sim O(1)$$

Theorem 3.2. *For the token-level KL penalty, the Q-values satisfy the Bellman recursions:*

$$Q^{\pi,KL}_{\textit{sample}}(s_{k-1}, a'_k) = d_k^{\textit{sample}} + \gamma \cdot \mathbb{E}_{\pi_\theta}[Q_d^{KL}(s_k, a''_k)]$$
$$Q_d^{\pi,KL}(s_k, a''_k) = d_k^{\textit{decision}} + \gamma \cdot \mathbb{E}_{\pi_\theta}[Q^{KL}_{\textit{sample}}(s_k, a'_{k+1})]$$

where the immediate penalties satisfy:

$$d_k^{\textit{sample}} = \sum_{j=1}^{L_k} \log \frac{\pi_{\textit{sample},\theta}(\textit{token}_{k,j} \mid \cdot)}{\pi_{\textit{sample,ref}}(\textit{token}_{k,j} \mid \cdot)} \sim O(L_k)$$

$$d_k^{\textit{decision}} = \log \frac{\pi_{d,\theta}(a_k \mid s_k)}{\pi_{d,\textit{ref}}(a_k \mid s_k)} \sim O(1)$$

The scale separation $|d_k^{\textit{sample}}|/|d_k^{\textit{decision}}| \approx L_k$ creates systematic gradient magnitude asymmetry.

[9] This corresponds to the "KL in reward" placement used in GRPO and PPO-based RLHF implementations.

Proof Sketch. The key observation is that d_k^{sample} sums over L_k token-level divergences while $d_k^{decision}$ is a single scalar, yielding $|d_k^{sample}| \approx L_k \cdot \delta$ versus $|d_k^{decision}| \approx \delta$ for token-level divergence magnitude δ. Working backwards: at step T, $Q_d^{KL}(s_T, \text{STOP}) = d_T^{decision} \sim O(1)$, while $Q_{sample}^{KL}(s_{T-1}, a'_T) = d_T^{sample} + \gamma \cdot Q_d^{KL}(s_T, \text{STOP}) \approx O(L_T)$. This asymmetry propagates: Q_{sample}^{KL} is dominated by immediate $O(L_k)$ penalties, while Q_d^{KL} has small immediate terms. No unified sufficient statistic Φ exists since satisfying both $\Phi(s_k) \approx O(L_k) + \gamma\Phi(s_{k+1})$ (sampling) and $\Phi(s_k) \approx O(1) + \gamma\Phi(s_{k+1})$ (decision) is inconsistent when $L_k \gg 1$. See Appendix A.5. □

Interpretation. The KL penalty creates asymmetric regularization: changes to π_{sample} incur immediate penalties proportional to L_k, while changes to π_d incur $O(1)$ penalties. Combined with the surrogate reward's symmetric push (Theorem 3.1, where $Q_{sample}^{reward} = Q_d^{reward} = \gamma^{\sum len(\cdot)} A_i$), the net effect is differential learning: π_d receives sustained gradients with minimal KL constraint, while π_{sample} is heavily regularized.[10]

4 Experiment and Calibration

The theoretical framework predicts that RL's balanced gradient attribution enables coherent learning of both π_{sample} and π_d, while SFT's unbalanced attribution leads to miss-meshed gradients that fail to develop an effective decision policy. We design experiments to validate these predictions and to decompose observed performance gains into their constituent policy components. The experiment set-up

4.1 Experimental Setup and Results

Table 1 presents model performance across multiplication tasks of increasing difficulty. The 3×3 and 3×4 tasks constitute the training distribution; all others are out-of-distribution (OOD). Both RL and SFT (reflection) achieve near-ceiling in-distribution performance (96–98% on 3×3, 91–92% on 3×4) and remain comparable on 3×5. The critical divergence emerges at 3×6: RL maintains 90% accuracy while SFT (reflection) drops to 49%. This gap widens dramatically further OOD—on 3×9, RL achieves 34% versus SFT (reflection)'s 0%.

SFT (reflection) substantially outperforms both Base and SFT (no reflection) across all task difficulties, indicating that reflection-rich training data does improve performance. However, the steep degradation pattern suggests this improvement reflects memorization of in-distribution patterns rather than learned generalization. Later in the section, we develop a statistical approach to estimate the performance of π_{sample} and π_d, decomposing the performance gap between RL and SFT (reflection) into contributions from each policy component.

4.2 A Simple Calibration of the Model

To decompose the performance gap between RL and SFT into contributions from π_{sample} and π_d, we construct a calibration model that abstracts the LLM's behavior into the two-stage decision-sampling process.

Sampling Accuracy (π_{sample}). We model the sampling policy by a single probability $p_s = P(A_k = A_Q^*)$, representing the likelihood that any given sample is correct.

Decision Policy (π_d). We model the decision policy as a classifier with two parameters:

$$p_{d|C} = P(\texttt{STOP} \mid A_k = A_Q^*),$$
$$p_{d|W} = P(\texttt{RESAMPLE} \mid A_k \neq A_Q^*).$$

The parameter $p_{d|C}$ captures the probability of correctly accepting a correct answer; $p_{d|W}$ captures the probability of correctly rejecting an incorrect answer. An effective decision policy requires both to be high. The framework predicts that RL develops high $p_{d|W}$ through the surrogate reward's balanced gradient attribution[11].

Model Accuracy. Under the two-stage model, overall accuracy is:

$$\text{Model Acc} = \frac{p_s \cdot p_{d|C}}{1-(p_s \cdot (1-p_{d|C}) + (1-p_s) \cdot p_{d|W})}.$$

We estimate $(p_s, p_{d|C}, p_{d|W})$ from model outputs: p_s from first-attempt accuracy, and the decision parameters from observed stopping and resampling behavior conditional on answer correctness.

Figure 1a presents calibrated parameters for the RL-trained model. The close agreement between

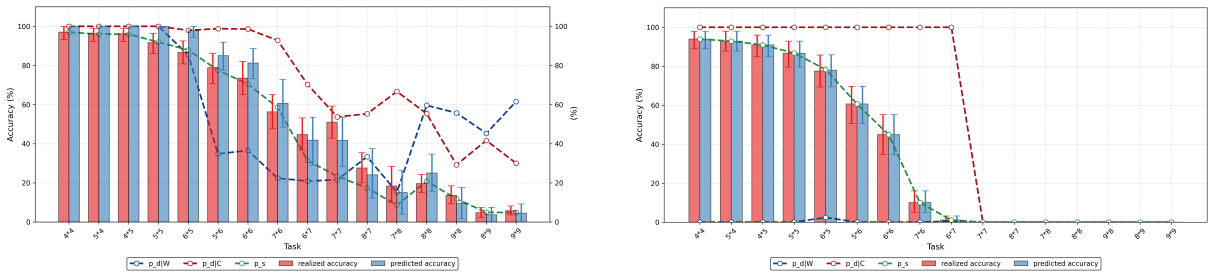
[10] This "unbalanced attribution" concerns gradient magnitude distribution, not existence of a unified V^π—which exists by standard theory. The distinction is whether immediate rewards differ by $O(1)$ (balanced) or $O(L)$ (unbalanced) factors.

[11] Also it's reasonable to conjecture that negative sampling and negative reward would tend to develops high $p_{d|C}$.

Table 1: MODEL PERFORMANCE ON ARITHMETIC TASKS

	3×3 (1)	3×4 (2)	3×5 (3)	3×6 (4)	3×7 (5)	3×8 (6)	3×9 (7)
A. Baselines							
Base	74.0 (65.4, 82.6)	34.0 (24.7, 43.3)	5.0 (0.7, 9.3)	1.0 (-1.0, 3.0)	4.0 (0.2, 7.8)	3.0 (-0.3, 6.3)	1.0 (-1.0, 3.0)
SFT (no reflection)	75.0 (66.5, 83.5)	32.0 (22.9, 41.1)	6.0 (1.3, 10.7)	2.0 (-0.7, 4.7)	1.0 (-1.0, 3.0)	0.0 (0.0, 0.0)	1.0 (-1.0, 3.0)
B. Reflecting Methods							
SFT (reflection)	96.0 (92.2, 99.8)	92.0 (86.7, 97.3)	94.0 (89.3, 98.7)	49.0 (39.2, 58.8)	4.0 (0.2, 7.8)	1.0 (-1.0, 3.0)	0.0 (0.0, 0.0)
RL	98.0 (95.3, 100.7)	91.0 (85.4, 96.6)	92.0 (86.7, 97.3)	90.0 (84.1, 95.9)	72.0 (63.2, 80.8)	53.0 (43.2, 62.8)	34.0 (24.7, 43.3)

NOTE: Columns (1)–(7) report accuracy percentages and 95% confidence intervals ($n = 100$) for arithmetic tasks of increasing complexity. Panel A displays results for baselines trained without retry patterns. Panel B displays results for models trained on trajectories with explicit reflection or reinforcement learning. All models are based on `Qwen2.5-7B-Instruct` (Qwen et al., 2025).



(a) RL Thinking Model

(b) SFT Thinking Model

Figure 1: **Calibrated Policy Parameters Across Arithmetic Task Sizes.** Comparison of the RL model (a) versus the SFT model (b). p_s: sampling accuracy (approximated by 1st try accuracy); $p_{d|C}$: probability of stopping conditional on getting a correct answer; $p_{d|W}$: probability of resampling conditional on getting an incorrect answer. Note that while $p_{d|C}$ remains high for both, the SFT model fails to maintain $p_{d|W}$ on harder tasks.

predicted and observed accuracy validates the two-stage decomposition. Three patterns emerge: (1) sampling accuracy p_s degrades monotonically with task difficulty, from approximately 80% in-distribution to below 20% on the most challenging OOD tasks; (2) decision parameters exhibit remarkable stability, with $p_{d|C}$ near ceiling and $p_{d|W}$ sustaining 40 60% even far out-of-distribution; (3) the calibrated model accurately predicts observed accuracy across the full difficulty range.

Figure 1b reveals a starkly different pattern for SFT. While $p_{d|C}$ similarly remains high, $p_{d|W}$ collapses toward zero on OOD tasks—the model fails to reject incorrect answers, instead echoing first attempts. Sampling accuracy p_s also degrades more precipitously than RL. The critical distinction: RL maintains discriminative ability ($p_{d|W} > 0$) where SFT does not, confirming that RL's superior generalization stems from learning an effective π_d that enables error correction even when π_{sample} falters.

5 Towards Understanding the Insufficiency of SFT, and the Power of RL

5.1 SFT Echo Chamber

Recent systematic analysis reveals that SFT consistently fails to develop self-correction despite extensive efforts to engineer reflection-rich training data. (Kang et al., 2025) constructed SFT datasets with varying reflection steps and corrective $F \to T$ transition proportions (0% to 100%). Models trained on maximum-reflection data outperformed minimal-reflection variants by 4.05%,

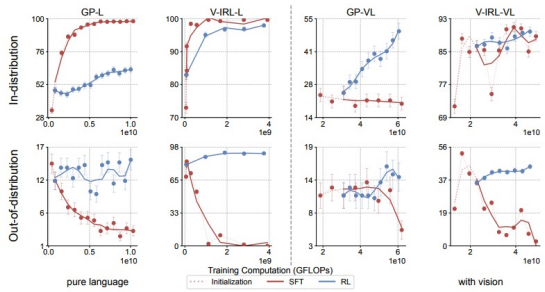


Figure 2: (Chu et al., 2025): ID/OOD performance comparison between SFT/RL

yet decomposition reveals first-candidate accuracy accounts for 3.75% while reflection contributes only 0.3%. More critically, $p(F \to T)$ showed no meaningful improvement across any configuration Table 2.

Table 2: (Kang et al., 2025): FT Training Cannot Improve Self-Correction

F→T Ratio in Data	p(F→T) Llama3.1-8B	p(F→T) Qwen2.5-7B
100%	0.053	0.036
50%	0.058	0.045
0%	0.050	0.041

Note: $p(F \to T)$ remains flat regardless of corrective reflection exposure.

The gradient attribution framework explains this pattern. SFT operates similarly to a KL-divergence between π_θ and π_{data}, maximizing likelihood of entire trajectories without decomposing credit between "good decision to RESAMPLE" versus "good sampling." Even with abundant $F \to T$ patterns in training data, SFT teaches π_{sample} to produce reflection-like text but provides misattributed gradient signal for π_d to learn when to trigger correction, creating the 'SFT echo chamber'.

5.2 Memorization vs. Generalization.

If SFT primarily improves π_{sample} without developing π_d, it is natural to predict SFT models will memorize training distributions rather than learn generalizable decision rules. (Chu et al., 2025) test this directly across four task variants. The results are stark: SFT degrades OOD performance by 8–80% while RL improves it by 3–61%. Crucially, this pattern persists even when SFT uses sub-optimal trajectories containing errors—memorization stems from the training objective's structure, not data quality.

5.3 Why do Dynamic Fine-tuning Works Better

(Wu et al., 2025) prove that SFT's gradient is equivalent to policy gradient with an implicit reward inversely proportional to model confidence ($\frac{1}{\pi_\theta}$). This $\frac{1}{\pi_\theta}$ weighting creates policy-entangled rewards where Q-values depend on future policy probabilities rather than outcomes, precluding a joint Q-function for π_{sample} and π_d[12]. DFT rescales the objective by π_θ, mitigating this entanglement[13] so that Q-values depend more on trajectory outcomes. Their empirical gains are substantial: on Qwen2.5-Math-7B, standard SFT degrades AIME24 from 6.68 to 2.48, while DFT improves it to 8.56.

6 Conclusion

We develop the Two-Stage Decision-Sampling Hypothesis as a mechanistic framework explaining why RL post-training produces self-reflection while SFT fails. We introduce the gradient attribution property—a novel analytical tool characterizing how reward gradients distribute between multitasking policy components under the DS-Hypothesis. We prove that standard surrogate rewards exhibit balanced gradient attribution, where both policy components receive proportional learning signals through a unified sufficient statistic. In contrast, KL-divergence penalties exhibit imbalanced gradient attribution: length-weighting in token-level objectives creates asymmetric regularization that heavily constrains π_{sample} while leaving π_d under-optimized. This asymmetry explains the fundamental mechanistic gap between RL and SFT.

Empirical validation on arithmetic reasoning confirms the framework's predictions. The calibrated two-stage decomposition reveals that RL's superior out-of-distribution generalization stems primarily from learning an effective decision policy rather than improved sampling. We further demonstrate how the gradient attribution framework explains multiple empirical phenomena: SFT's echoing effect, memorization versus generalization patterns, DFT's effectiveness.

[12] Refer to Appendix A and B for proof of unbalanced gradient attribution.

[13] Though not fully canceling it due to length asymmetry; we provide rigorous proof in Appendix D.

Limitations

Our analysis abstracts away clipping, trust-region constraints, and other stabilization mechanisms standard in practical implementations (e.g., PPO, GRPO) to preserve analytical tractability and result generality. These mechanisms sacrifice analytical properties in favor of training stability. Extending the gradient attribution framework to exactly match specific algorithmic implementations—characterizing how clipping interacts with the reward structure or how trust regions modify the effective gradient flow—requires additional theoretical work.

We adopt mathematical reasoning as our empirical context for its clarity, well-defined correctness criteria, and controllable difficulty. Although the theory is not domain specific, to empirically validate the theory prediction in other domains such as code generation, open-ended reasoning, or creative tasks remains requires careful methodological innovation on developing valid decomposition measures and correctness criteria.

We analyze binary correctness rewards $R(\tau) = \mathbb{I}(A_T = A_Q^*)$. Many practical applications employ more nuanced reward functions, including partial credit, process-based rewards, or learned reward models. The gradient attribution properties under these alternative reward structures may differ and warrant separate investigation.

Acknowledgments

AI assistants were used for editing and proofreading text, and LaTeX formatting. All technical content, theoretical derivations, experimental design, and scientific conclusions are the authors' own work. AI-generated content was reviewed and verified by the authors.

References

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. 2020. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Preprint*, arXiv:1908.00261.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 117 others. 2025. Llama-nemotron: Efficient reasoning models. *Preprint*, arXiv:2505.00949.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *Preprint*, arXiv:2501.17161.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI, Daya Guo, and 1 others. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: A theoretical perspective. *Preprint*, arXiv:2305.15408.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Jie Huang, Xinyun Gu, Leyang Shen, Yeganeh Kordi, Bailin Wu, Kaiyan Zhang, Dacheng Li, Qianyu Li, Pengcheng Zhang, Yiming Liu, and 1 others. 2024. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*.

Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. 2025. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier. *Preprint*, arXiv:2506.10406.

Ryo Kamoi, Olga Golovneva, Po-Yao Hsu, Asli Celikyilmaz, Yejin Kim, Yejin Choi, and Markus Freitag. 2024. A critical analysis of self-correction capabilities in large language models. *Transactions of the Association for Computational Linguistics*, 12.

Liwei Kang, Yue Deng, Yao Xiao, Zhanfeng Mo, Wee Sun Lee, and Lidong Bing. 2025. First try matters: Revisiting the role of reflection in reasoning models. *Preprint*, arXiv:2510.08308.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, Seyed Mehran Kazemi, and Sergey Levine. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. S^2r: Teaching llms to self-verify and self-correct via reinforcement learning. *Preprint*, arXiv:2502.12853.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. *Preprint*, arXiv:1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 1999a. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12.

Richard S. Sutton, Doina Precup, and Satinder Singh. 1999b. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. 2025. On the generalization of sft: A reinforcement learning perspective with reward rectification. *Preprint*, arXiv:2508.05629.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. 2025. Self-rewarding correction for mathematical reasoning. *Preprint*, arXiv:2502.19613.

Xingcheng Xu, Zibo Zhao, Haipeng Zhang, and Yanqing Yang. 2025. Principled understanding of generalization for generative transformer models in arithmetic reasoning tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*. To appear.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Xutong Zhao, Tengyu Xu, Xuewei Wang, Zhengxing Chen, Di Jin, Liang Tan, Yen-Ting, Zishun Yu, Zhuokai Zhao, Yun He, Sinong Wang, Han Fang, Sarath Chandar, and Chen Zhu. 2025. Boosting llm reasoning via spontaneous self-correction. *Preprint*, arXiv:2506.06923.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *Preprint*, arXiv:2411.14405.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Proofs for Gradient Attribution Property Framework

This appendix provides complete mathematical derivations for all results in Section 3.

A.1 Proof of Lemma 3.1 (Trajectory Factorization)

Proof. We prove this by applying the chain rule of probability to the sequential generation process. A trajectory τ of length T is:

$$\tau = (A_1, T_1, a_1, A_2, T_2, a_2, \ldots, A_T, T_T, a_T)$$

where $a_k \in \{\text{RESAMPLE}, \text{STOP}\}$ for $k = 1, \ldots, T-1$ and $a_T = \text{STOP}$ (by definition, the trajectory ends with STOP).

By the chain rule:

$$\begin{aligned} P(\tau|Q;\theta) = P(A_1, T_1|Q;\theta) \\ \cdot P(a_1|A_1, T_1, Q;\theta) \\ \cdot P(A_2, T_2|a_1, A_1, T_1, Q;\theta) \cdots \end{aligned}$$

Sequentially we write it to be:

$$\begin{aligned} P(\tau|Q;\theta) = \pi_{\text{sample}}(A_1, T_1|s_0;\theta) \\ \cdot \pi_d(\text{RESAMPLE}|s_1;\theta) \\ \cdot \pi_{\text{sample}}(A_2, T_2|s_1;\theta) \cdots \\ \cdot \pi_d(\text{RESAMPLE}|s_{T-1};\theta) \\ \cdot \pi_{\text{sample}}(A_T, T_T|s_{T-1};\theta) \\ \cdot \pi_d(\text{STOP}|s_T;\theta) \end{aligned}$$

Thus:

$$\begin{aligned} P(\tau|Q;\theta) = \left[\prod_{k=1}^{T} \pi_{\text{sample}}(A_k, T_k|s_{k-1};\theta)\right] \\ \cdot \left[\prod_{k=1}^{T-1} \pi_d(\text{R}|s_k;\theta)\right] \cdot \pi_d(\text{S}|s_T;\theta) \end{aligned}$$

□

A.2 Proof of Corollary 3.1 (Gradient Decomposition)

Proof. From Lemma 3.1:

$$\begin{aligned} \log P(\tau|Q;\theta) = \sum_{k=1}^{T} \log \pi_{\text{sample}}(A_k, T_k|s_{k-1};\theta) \\ + \sum_{k=1}^{T-1} \log \pi_d(\text{RESAMPLE}|s_k;\theta) \\ + \log \pi_d(\text{STOP}|s_T;\theta) \end{aligned}$$

Taking the gradient with respect to θ:

$$\begin{aligned} \nabla_\theta \log P(\tau|Q;\theta) = \sum_{k=1}^{T} \nabla_\theta \log \pi_{\text{sample}}(A_k, T_k|s_{k-1};\theta) \\ + \sum_{k=1}^{T-1} \nabla_\theta \log \pi_d(\text{RESAMPLE}|s_k;\theta) \\ + \nabla_\theta \log \pi_d(\text{STOP}|s_T;\theta) \end{aligned}$$

Rewrite the decision gradient more compactly by defining:

- $a_k = \text{RESAMPLE}$ for $k = 1, \ldots, T-1$
- $a_T = \text{STOP}$
- $a_0 = \text{START}$ (initial action, could be included for notational completeness)

Then:

$$\begin{aligned} \nabla_\theta \log P(\tau|Q;\theta) = \sum_{k=1}^{T} \nabla_\theta \log \pi_{\text{sample}}(A_k, T_k|s_{k-1};\theta) \\ + \sum_{k=0}^{T} \nabla_\theta \log \pi_d(a_k|s_k;\theta) \end{aligned}$$

where we interpret $\pi_d(a_0|s_0) = 1$ (deterministic start) and thus $\nabla_\theta \log \pi_d(a_0|s_0) = 0$. This completes the proof. □

A.3 Proof of Lemma 3.2 (Policy Gradient Theorem - Applied)

Proof. We adapt the standard policy gradient theorem to our factorized policy structure.

Standard Policy Gradient Theorem The standard policy gradient theorem (Sutton et al., 1999b) states that for a policy π_θ and expected return $J(\theta) = \mathbb{E}_{\tau\sim\pi\theta}[R(\tau)]$:

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(a|s;\theta) \cdot Q^\pi(s,a)\, da\, ds$$

where $\rho^\pi(s)$ is the discounted state visitation measure.

A.3.1 State Visitation Distribution in Our Setting

Define $\rho_t^\pi(s)$ as the probability of visiting state $s = (Q, A, T)$ at step t:

$$\begin{aligned} \rho_t^\pi(s_t) = \sum_{s_{t-1}} \rho_{t-1}^\pi(s_{t-1}) \\ \cdot \pi_d(\text{RESAMPLE}|s_{t-1}) \cdot \pi_{\text{sample}}(A_t, T_t|s_{t-1}) \end{aligned}$$

This says: to reach state s_t at time t, we must have been at some state s_{t-1} at time $t-1$, chosen to RESAMPLE, and then sampled (A_t, T_t).

Discounted state visitation:

$$\rho^\pi(s) = \sum_{t=1}^{\infty} \gamma^t \rho_t^\pi(s)$$

At each state $s_k = (Q, A_k, T_k)$, there are two types of actions:

- Sampling actions $a' = (A, T)$ drawn from $\pi_{\text{sample}}(\cdot|s_{k-1})$
- Decision actions $a'' \in \{\text{STOP}, \text{RESAMPLE}\}$ drawn from $\pi_d(\cdot|s_k)$

However, these occur at different points in the trajectory:

- After state s_{k-1}, we take sampling action $a'_k = (A_k, T_k)$ to reach state s_k
- At state s_k, we take decision action $a''_k \in \{\text{STOP}, \text{RESAMPLE}\}$

For sampling actions at state s_{k-1}:

$$\int_{\mathcal{A}_{\text{sample}}} \nabla_\theta \pi_{\text{sample},\theta}(a'|s_{k-1}) \cdot Q^\pi(s_{k-1}, a')\, da' = \mathbb{E}_{a' \sim \pi_{\text{sample}}} \left[\nabla_\theta \log \pi_{\text{sample},\theta}(a'|s_{k-1}) \cdot Q^\pi_{\text{sample}}(s_{k-1}, a')\right]$$

For decision actions at state s_k:

$$\sum_{a'' \in \{\text{STOP,RESAMPLE}\}} \nabla_\theta \pi_{d,\theta}(a''|s_k) \cdot Q^\pi(s_k, a'') = \mathbb{E}_{a'' \sim \pi_d} \left[\nabla_\theta \log \pi_{d,\theta}(a''|s_k) \cdot Q^\pi_d(s_k, a'')\right]$$

Combining over all states in a trajectory of length T:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_{\text{sample},\theta}(a'_t|s_{t-1}) \cdot Q^\pi_{\text{sample}}(s_{t-1}, a'_t)\right] + \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_{d,\theta}(a''_t|s_t) \cdot Q^\pi_d(s_t, a''_t)\right]$$

We can combine these into a single expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \left(\nabla_\theta \log \pi_{\text{sample}}(a'_t|s_{t-1}) \cdot Q^\pi_{\text{sample}}(s_{t-1}, a'_t) + \nabla_\theta \log \pi_d(a''_t|s_t) \cdot Q^\pi_d(s_t, a''_t)\right)\right]$$

where we use the convention that terms with $t = 0$ correspond to the initial action. □

A.4 Proof of Theorem 3.1 (Balanced Attribution of Surrogate Reward)

The surrogate reward for trajectory τ_i is:

$$L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\text{old}}} \left[\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)} A_i\right]$$

where:

- A_i is the advantage for trajectory τ_i, computed using Group Relative Advantage Estimation (GRAE)
- For query Q_i with multiple sampled trajectories $\{\tau_1, \ldots, \tau_G\}$ (the "group"):

$$A_i = \frac{R(\tau_i) - \text{mean}(R(\tau_1), \ldots, R(\tau_G))}{\text{std}(R(\tau_1), \ldots, R(\tau_G))}$$

- The reward function is $R(\tau) = \mathbb{I}(A_T = A^*_Q)$

Taking the gradient with respect to θ:

$$\nabla_\theta L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\text{old}}} \left[\nabla_\theta \left(\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)}\right) \cdot A_i\right]$$

Using the log-derivative trick:

$$\nabla_\theta \left(\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)}\right) = \frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)} \cdot \nabla_\theta \log \pi_\theta(\tau_i|Q_i)$$

From Corollary 3.1:

$$\nabla_\theta \log \pi_\theta(\tau_i|Q_i) = \sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\text{sample},\theta}(A_k, T_k|s_{k-1}) + \sum_{k=0}^{T_i} \nabla_\theta \log \pi_{d,\theta}(a_k|s_k)$$

Therefore, when sampling from π_{old} and using importance weighting:

$$\nabla_\theta L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_{\text{old}}} \left[\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{\text{old}}(\tau_i|Q_i)} \cdot A_i \cdot \left(\sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\text{sample},\theta}(\cdot) + \sum_{k=0}^{T_i} \nabla_\theta \log \pi_{d,\theta}(\cdot)\right)\right]$$

By importance sampling, this equals:

$$\nabla_\theta L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_\theta} \left[A_i \left(\sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\text{sample},\theta}(A_k, T_k|s_{k-1}) + \sum_{k=0}^{T_i} \nabla_\theta \log \pi_{d,\theta}(a_k|s_k)\right)\right]$$

Comparing with the policy gradient decomposition from Lemma 3.2, we identify the Q-values by noting that the gradient must have the form:

$$\nabla_\theta L_{\text{reward}}(\theta) = \mathbb{E}_{\tau_i \sim \pi_\theta}\Bigg[\sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\text{sample},\theta}(\cdot) \cdot Q^{\pi,\text{reward}}_{\text{sample}}(\cdot) + \sum_{k=0}^{T_i} \nabla_\theta \log \pi_{d,\theta}(\cdot) \cdot Q^{\pi,\text{reward}}_{d}(\cdot)\Bigg]$$

For the surrogate reward, both Q-values equal the length-discounted advantage:

$$Q^{\pi,\text{reward}}_{\text{sample}}(s_{k-1}, (A_k, T_k)) = \gamma^{\sum_{j=k}^{T_i} \text{len}(A_j,T_j)} A_i$$

$$Q^{\pi,\text{reward}}_{d}(s_k, a_k) = \gamma^{\sum_{j=k}^{T_i} \text{len}(A_j,T_j)} A_i$$

Define:

$$\Phi(s_k) = \gamma^{\sum_{j=k}^{T_i} \text{len}(A_j,T_j)} A_i$$

This is the sufficient statistic of future rewards at state s_k. It encodes: "the trajectory will receive advantage A_i at the end, which is at temporal distance $\sum_{j=k}^{T_i} \text{len}(A_j, T_j)$ from now." Then:

$$Q^{\pi,\text{reward}}_{\text{sample}}(s_{k-1}, a'_k) = \Phi(s_k)$$

$$Q^{\pi,\text{reward}}_{d}(s_k, a''_k) = \Phi(s_k)$$

Both Q-values can be expressed using the same sufficient statistic Φ, evaluated at the appropriate state. This satisfies Definition 3.1 for Balanced Gradient Attribution. □

A.5 Proof of Theorem 3.2 (Unbalanced Attribution of KL Penalty)

We analyze the gradient attribution properties of the token-level KL penalty.

KL Penalty Formulation. The standard token-level KL penalty, computed on trajectories sampled from the current policy, is:

$$D^{token}_{KL}(\tau) = \sum_{t=1}^{|\tau|} \log \frac{\pi_\theta(a_t|s_t)}{\pi_{ref}(a_t|s_t)}$$

where the sum runs over all tokens in trajectory τ. The RL objective includes this as a penalty:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\Big[R(\tau) - w \cdot D^{token}_{KL}(\tau)\Big]$$

Decomposition Under the Two-Stage Framework. A trajectory τ of length T (i.e., T sampling-decision cycles) consists of:

- T sampling actions, where the k-th sampling action (A_k, T_k) comprises L_k tokens
- T decision actions $a_k \in \{\text{RESAMPLE}, \text{STOP}\}$, each a single token

The total number of tokens is $|\tau| = \sum_{k=1}^{T} L_k + T$. The KL penalty decomposes as:

$$D^{\text{token}}_{KL}(\tau) = \underbrace{\sum_{k=1}^{T}\sum_{j=1}^{L_k} \log \frac{\pi_{\text{sample},\theta}(\text{token}_{k,j} \mid \text{context})}{\pi_{\text{sample,ref}}(\text{token}_{k,j} \mid \text{context})}}_{\text{Sampling component}} + \underbrace{\sum_{k=1}^{T} \log \frac{\pi_{d,\theta}(a_k \mid s_k)}{\pi_{d,\text{ref}}(a_k \mid s_k)}}_{\text{Decision component}}$$

Immediate Penalties. Define the immediate KL penalty attributed to each action:

For sampling actions:

$$d^{sample}_k = \sum_{j=1}^{L_k} \log \frac{\pi_{sample,\theta}(token_{k,j}|context)}{\pi_{sample,ref}(token_{k,j}|context)}$$

This is a sum over L_k terms. If individual token-level log-ratios have typical magnitude δ (which may be positive or negative depending on whether the current policy assigns higher or lower probability than the reference), then:

$$|d^{sample}_k| \leq \sum_{j=1}^{L_k} \left|\log \frac{\pi_{sample,\theta}(token_{k,j}|\cdot)}{\pi_{sample,ref}(token_{k,j}|\cdot)}\right| \sim O(L_k \cdot \delta)$$

For decision actions:

$$d^{decision}_k = \log \frac{\pi_{d,\theta}(a_k|s_k)}{\pi_{d,ref}(a_k|s_k)}$$

This is a single scalar:

$$|d^{decision}_k| \sim O(\delta)$$

The ratio of magnitudes is approximately $L_k : 1$. For typical reasoning traces with $L_k \approx 100$–500 tokens, this represents a two-orders-of-magnitude asymmetry.

Q-Value Recursions.

Proof. The policy gradient for the KL penalty takes the standard form:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[D_{\text{KL}}^{\text{token}}(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot Q^{\text{KL}}(s_t, a_t)\right]$$

where $Q^{KL}(s_t, a_t)$ is the expected cumulative KL penalty from taking action a_t at state s_t.

We derive the Q-values by working backwards from the terminal state.

Terminal state (STOP at step T):

$$Q_d^{KL}(s_T, \text{STOP}) = d_T^{decision}$$

There are no future actions, so the Q-value equals the immediate penalty. Magnitude: $O(1)$.

Final sampling action (step T):

$$Q_{sample}^{KL}(s_{T-1}, a'_T) = d_T^{sample} + \gamma \cdot Q_d^{KL}(s_T, \text{STOP})$$

Substituting:

$$\begin{aligned} Q_{sample}^{KL}(s_{T-1}, a'_T) &= d_T^{sample} + \gamma \cdot d_T^{decision} \\ &\sim O(L_T) + O(1) \approx O(L_T) \end{aligned}$$

The immediate sampling penalty dominates.

Penultimate decision (RESAMPLE at step $T-1$):

$$Q_d^{KL}(s_{T-1}, \text{RESAMPLE}) = d_{T-1}^{decision} + \gamma \cdot \mathbb{E}_{\pi_\theta}\left[Q_{sample}^{KL}(s_{T-1}, a'_T)\right]$$

Magnitude:

$$Q_d^{KL}(s_{T-1}, \text{RESAMPLE}) \sim O(1) + \gamma \cdot O(L_T) \approx O(\gamma L_T)$$

The future sampling penalty dominates, but note: the *immediate* contribution is only $O(1)$.

General recursion (for $k < T$):

$$\begin{aligned} Q_{\text{sample}}^{\text{KL}}(s_{k-1}, a'_k) &= d_k^{\text{sample}} + \gamma \cdot \mathbb{E}_{\pi_\theta}\left[Q_d^{\text{KL}}(s_k, a''_k)\right] \\ Q_d^{\text{KL}}(s_k, \text{RESAMPLE}) &= d_k^{\text{decision}} + \gamma \cdot \mathbb{E}_{\pi_\theta}\left[Q_{\text{sample}}^{\text{KL}}(s_k, a'_{k+1})\right] \end{aligned}$$

Scale separation analysis. Define the accumulated future KL from state s_k:

$$V^{KL}(s_k) = \mathbb{E}_{\pi_\theta}\left[\sum_{t\geq k} \gamma^{t-k} d_t \,\middle|\, s_k\right]$$

For the sampling policy at step k:

$$Q_{sample}^{KL}(s_{k-1}, a'_k) = \underbrace{d_k^{sample}}_{\sim O(L_k)} + \gamma \cdot V^{KL}(s_k)$$

For the decision policy at step k:

$$Q_d^{KL}(s_k, a''_k) = \underbrace{d_k^{decision}}_{\sim O(1)} + \gamma \cdot V^{KL}(s_{k+1})$$

Although $V^{KL}(s_k)$ and $V^{KL}(s_{k+1})$ are similar in magnitude (both accumulate future penalties), the *immediate* contributions differ by a factor of L_k.

Impossibility of unified sufficient statistic. Suppose a single sufficient statistic $\Phi : \mathcal{S} \to \mathbb{R}$ exists such that:

$$\begin{aligned} Q_{sample}^{KL}(s_{k-1}, a'_k) &= f_{sample}(s_{k-1}, a'_k, \Phi(s_k)) \\ Q_d^{KL}(s_k, a''_k) &= f_d(s_k, a''_k, \Phi(s_{k+1})) \end{aligned}$$

From the Bellman-style recursions, Φ must satisfy:

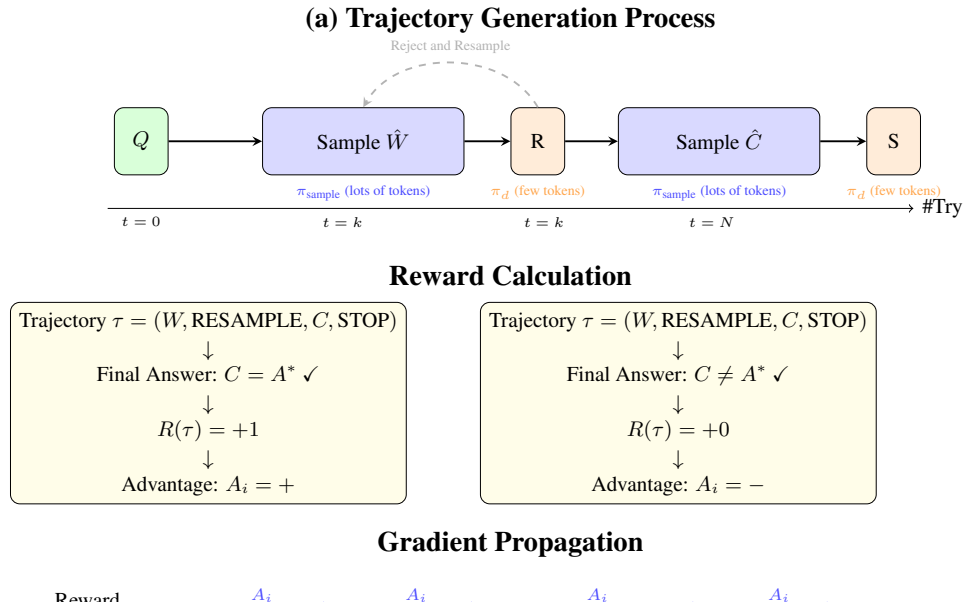
$$\begin{aligned} &\Phi(s_k) = O(L_k) + \gamma\Phi(s_{k+1}) \quad \text{(from sampling Q-value)} \\ &\Phi(s_k) = O(1) + \gamma\Phi(s_{k+1}) \quad \text{(from decision Q-value)} \end{aligned}$$

Subtracting: $0 = O(L_k) - O(1)$, which is a contradiction when $L_k \gg 1$.

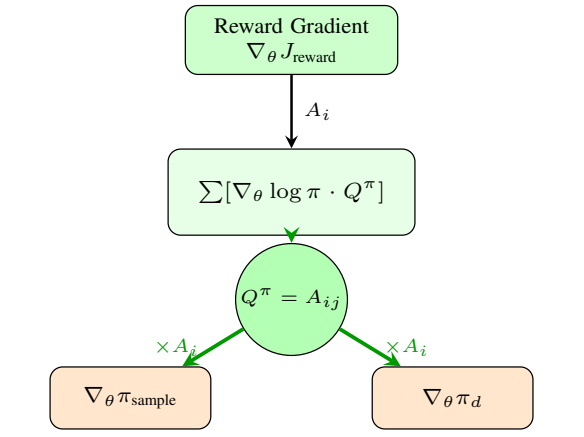
Therefore, no single sufficient statistic Φ exists. The Q-values require distinct information structures:

$$\begin{aligned} \Phi_{sample}(s_k) &= d_k^{sample} + \gamma\Phi_d(s_{k+1}) \sim O(L_k) + \gamma(\cdot) \\ \Phi_d(s_k) &= d_k^{decision} + \gamma\Phi_{sample}(s_k) \sim O(1) + \gamma(\cdot) \end{aligned}$$

This establishes Unbalanced Gradient Attribution. □



(b) Surrogate Reward (Balanced)

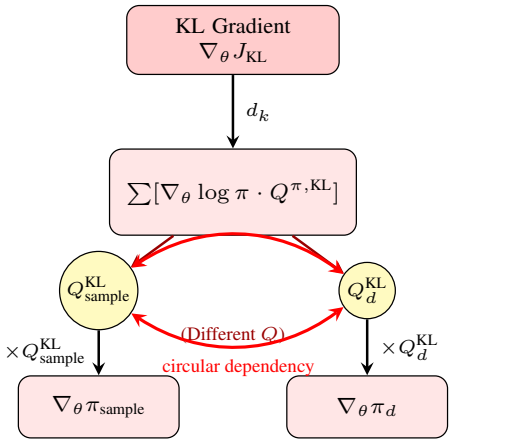


$$Q^\pi_{\text{sample}} \approx Q^\pi_d \approx A_{ij} \text{ (length discounted } A_i)$$

$$\nabla_\theta J \approx \sum_k \nabla_\theta \log \pi_{\text{sample}}(A_k|s_{k-1})\cdot A_{ik} + \sum_k \nabla_\theta \log \pi_d(a_k|s_k)\cdot A_{ik}$$

balanced gradient attribution

(c) KL Penalty (Unbalanced)



$$Q^{\text{KL}}_{\text{sample}} = d^{\text{sample}} + \mathbb{E}_{\pi_d}[\gamma\cdot Q^{\text{KL}}_d]$$

$$Q^{\text{KL}}_d = d^{\text{decision}} + \mathbb{E}_{\pi_{\text{sample}}}[\gamma\cdot Q^{\text{KL}}_{\text{sample}}]$$

$Q^{\text{KL}}_{\text{sample}} \not\approx Q^{\text{KL}}_d$ **with circular dependency**

Imbalanced gradient attribution

B Alternative Proof of Theorem 3.1 and Theorem 3.2

The probability of a trajectory τ under the combined policy can be expressed as:

$$P(\tau \mid Q;\theta) = \left(\prod_{k=1}^{T} \pi_{\text{sample}}(A_k, T_k \mid s_{k-1};\theta)\right) \cdot \left(\prod_{k=1}^{T-1} \pi_d(\text{RESAMPLE} \mid s_k;\theta)\right) \cdot \pi_d(\text{STOP} \mid s_T;\theta) \tag{1}$$

Taking the logarithm:

$$\log P(\tau \mid Q;\theta) = \sum_{k=1}^{T} \log \pi_{\text{sample}}(A_k, T_k \mid s_{k-1};\theta) + \sum_{k=1}^{T-1} \log \pi_d(\text{RESAMPLE} \mid s_k;\theta) + \log \pi_d(\text{STOP} \mid s_T;\theta) \tag{2}$$

Important: The sampling log-probability $\log \pi_{sample}(A_k, T_k | s_{k-1})$ decomposes at the token level as:

$$\log \pi_{\text{sample}}(A_k, T_k \mid s_{k-1}) = \sum_{j=1}^{L_k} \log \pi_{\text{sample}}(\text{token}_{k,j} \mid s_{k-1}, \text{token}_{k,1}, \ldots, \text{token}_{k,j-1}) \tag{3}$$

The gradient with respect to θ:

$$\nabla_\theta \log P(\tau \mid Q;\theta) = \sum_{k=1}^{T} \nabla_\theta \log \pi_{\text{sample}}(A_k, T_k \mid s_{k-1};\theta) + \sum_{k=1}^{T-1} \nabla_\theta \log \pi_d(\text{RESAMPLE} \mid s_k;\theta) + \nabla_\theta \log \pi_d(\text{STOP} \mid s_T;\theta) \tag{4}$$

For convenience, we denote $A_0, T_0 = \emptyset$.

B.1 Policy Gradient Decomposition

Using the policy gradient theorem adapted to our factorized policy structure (see Appendix A.3):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_{\text{sample},\theta}(a'_t \mid s_t)\cdot Q^\pi(s_t, a'_t) + \nabla_\theta \log \pi_{d,\theta}(a''_t \mid s_t)\cdot Q^\pi(s_t, a''_t)\right)\right] \tag{5}$$

B.2 Attribution Analysis for Surrogate Reward

Surrogate Reward Term.

$$L_{clip}(\theta) = \mathbb{E}_{\tau_i\sim\pi_{old}}\left[\frac{\pi_\theta(\tau_i|Q_i)}{\pi_{old}(\tau_i|Q_i)}A_i\right] \tag{6}$$

where A_i is the group-relative advantage computed via GRAE:

$$A_i = \frac{R(\tau_i) - \text{mean}(R(\tau_1), \ldots, R(\tau_G))}{\text{std}(R(\tau_1), \ldots, R(\tau_G))} \tag{7}$$

The probability ratio factorizes as:

$$\frac{\pi_\theta(\tau_i \mid Q_i)}{\pi_{\text{old}}(\tau_i \mid Q_i)} = \underbrace{\left(\prod_{k=1}^{T_i} \frac{\pi_{\theta,\text{sample}}(A_k, T_k \mid s_{k-1})}{\pi_{\text{old,sample}}(A_k, T_k \mid s_{k-1})}\right)}_{\text{Sampling ratio}} \cdot \underbrace{\left(\prod_{k=1}^{T_i-1} \frac{\pi_{\theta,d}(\text{RESAMPLE} \mid s_k)}{\pi_{\text{old},d}(\text{RESAMPLE} \mid s_k)}\right)}_{\text{Resampling ratio}} \cdot \underbrace{\frac{\pi_{\theta,d}(\text{STOP} \mid s_{T_i})}{\pi_{\text{old},d}(\text{STOP} \mid s_{T_i})}}_{\text{Stopping ratio}} \tag{8}$$

Neglecting the clipping for analytical tractability, the gradient is:

$$\nabla_\theta L_{\text{reward}} \propto A_i\left[\sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\theta,\text{sample}}(a'_k \mid s_{k-1}) + \sum_{k=0}^{T_i} \nabla_\theta \log \pi_{\theta,d}(a''_k \mid s_k)\right] \tag{9}$$

Balanced attribution: The advantage A_i multiplies both sampling and decision gradients equally. Both Q-values reduce to the same sufficient statistic:

$$Q_d^{reward} = Q_{sample}^{reward} \sim \gamma^{\sum_{j=0}^{T_i-k} len(A_{k+j}, T_{k+j})} A_i \tag{10}$$

B.3 Attribution Analysis for KL Penalty

Token-Level KL Decomposition. The KL penalty decomposes at the token level:

$$D_{KL}^{\text{token}}(\tau_i) = \sum_{k=1}^{T_i} \underbrace{\sum_{j=1}^{L_k} \log \frac{\pi_{\text{sample},\theta}(\text{token}_{k,j} \mid \cdot)}{\pi_{\text{sample,ref}}(\text{token}_{k,j} \mid \cdot)}}_{d_k^{\text{sample}}} + \sum_{k=1}^{T_i} \underbrace{\log \frac{\pi_{d,\theta}(a_k \mid s_k)}{\pi_{d,\text{ref}}(a_k \mid s_k)}}_{d_k^{\text{decision}}} \tag{11}$$

The immediate penalties are:

- **Sampling:** $d_k^{sample} = \sum_{j=1}^{L_k} \log(\pi_\theta/\pi_{ref})$ is a sum of L_k terms $\Rightarrow O(L_k)$
- **Decision:** $d_k^{decision} = \log(\pi_\theta/\pi_{ref})$ is a single term $\Rightarrow O(1)$

Q-Value Recursions. Working backwards from the terminal state:

Value of final STOP action:

$$Q_d^{KL}(s_{T_i}, \text{STOP}) = d_{T_i}^{decision} \sim O(1) \tag{12}$$

Value of preceding sampling action:

$$\begin{aligned} Q_{\text{sample}}^{\text{KL}}(s_{T_i-1}, a'_{T_i}) &= d_{T_i}^{\text{sample}} + \gamma \cdot Q_d^{\text{KL}}(s_{T_i}, \text{STOP}) \\ &\sim O(L_{T_i}) + O(1) \approx O(L_{T_i}) \end{aligned} \tag{13}$$

Value of RESAMPLE action:

$$\begin{aligned} Q_d^{\text{KL}}(s_k, \text{RESAMPLE}) &= d_k^{\text{decision}} + \gamma \cdot \mathbb{E}_{\pi_\theta}[Q_{\text{sample}}^{\text{KL}}(s_k, a'_{k+1})] \\ &\sim O(1) + O(\gamma L_{k+1}) \end{aligned} \tag{14}$$

Asymmetric Gradient Attribution. The gradient contributions are:

$$\begin{aligned} \text{Sampling:} \quad & \sum_{k=1}^{T_i} \nabla_\theta \log \pi_{\text{sample},\theta}(a'_k \mid s_{k-1}) \cdot Q_{\text{sample}}^{\text{KL}}(s_{k-1}, a'_k) \\ \text{Decision:} \quad & \sum_{k=1}^{T_i} \nabla_\theta \log \pi_{d,\theta}(a''_k \mid s_k) \cdot Q_d^{\text{KL}}(s_k, a''_k) \end{aligned} \tag{15}$$

The Q-values have incompatible sufficient statistics:

- Q_{sample}^{KL}: dominated by immediate $O(L_k)$ penalty
- Q_d^{KL}: immediate penalty is $O(1)$, accumulates future sampling penalties

This structural asymmetry arising from the sum over L_k tokens in sampling versus single-token decisions creates **Unbalanced Gradient Attribution**.

C Numerical Illustration of Gradient Attribution

To make the theoretical derivations in the preceding sections concrete, this section presents a simplified numerical example focused on a key scenario: where correct and incorrect answers have the same length. The objective is to trace the flow of gradients from the two primary components of the Group Relative Policy Optimization (GRPO) objective—the surrogate reward[14] and the KL penalty—back to the parameters of the sampling policy (π_{sample}) and the decision policy (π_d).

This exercise will quantitatively demonstrate that the asymmetric regularization effect is not dependent on incorrect answers being more verbose. Instead, it is an architectural consequence of the length-weighted formulation of the KL penalty, which applies a significant penalty to any long generated sequence from π_{sample} while applying a negligible penalty to the single-step actions of π_d.

C.1 Scenario and Policy Parameterization

We establish a minimal yet illustrative scenario to examine the gradient dynamics under the condition of equal answer lengths.

Problem Setup The model is tasked with a simple arithmetic problem: "What is 7 * 8?".

Action Space

- **Sampling Policy (π_{sample}):** Generates an answer. We consider two outcomes: a Correct answer (C), "56", or a Wrong answer (W), "54".
- **Decision Policy (π_d):** Evaluates the generated answer and chooses to STOP or RESAMPLE.

Policy Parameterization Each policy choice is modeled using a logistic sigmoid function, $\sigma(x) = 1/(1+e^{-x})$, applied to a single learnable logit (θ).

- **Sampling Policy (π_{sample}):** Governed by a single parameter, θ_s. The probability of generating a correct answer is $P(C|\theta_s) = \sigma(\theta_s)$. We initialize $\theta_s = 0.4$, yielding $P(C) \approx 0.5987$ and $P(W) = 1 - P(C) \approx 0.4013$.
- **Decision Policy (π_d):** Conditioned on the answer from π_{sample}.

[14] We also simplify the clipped part away

- For a correct answer, the stop probability is $P(\text{STOP}|C, \theta_{d,C}) = \sigma(\theta_{d,C})$. We initialize $\theta_{d,C} = 2.2$, yielding $P(\text{STOP}|C) \approx 0.9002$.
- For a wrong answer, the resample probability is $P(\text{RESAMPLE}|W, \theta_{d,W}) = \sigma(\theta_{d,W})$. We initialize $\theta_{d,W} = 1.4$, yielding $P(\text{RESAMPLE}|W) \approx 0.8022$.

Length Assignment This is the critical modification for this example. We set the lengths of both correct and incorrect answers to be equal and significant.

- Length of Correct answer: $len(C) = 8$ tokens.
- Length of Wrong answer: $len(W) = 8$ tokens.

Reference Policy (π_{orig}) The KL penalty regularizes the current policy (π_θ) against a reference policy (π_{orig}), typically the initial SFT model.

- $\theta_{s,orig} = 0.3 \implies P_{orig}(C) \approx 0.5744, P_{orig}(W) \approx 0.4256$.
- $\theta_{d,C,orig} = 2.0 \implies P_{orig}(\text{STOP}|C) \approx 0.8808$.
- $\theta_{d,W,orig} = 1.2 \implies P_{orig}(\text{RESAMPLE}|W) \approx 0.7685$.

Part 1: Symmetric Gradient Push from the Surrogate Reward

First, we analyze the gradient from the surrogate reward. This calculation is entirely independent of sequence length, demonstrating its symmetric nature.

Trajectory and Advantage We consider a single successful trajectory τ_i: the model initially generates a Wrong answer, correctly chooses to RESAMPLE, then generates a Correct answer and correctly chooses to STOP. GRPO is a "critic-less" algorithm that estimates advantage relative to a group of trajectories. We assume this trajectory is better than the group average and assign it a normalized advantage of $A_i = +0.5$.

Gradient Calculation The gradient of the reward objective is $\nabla_\theta J_{Reward}(\theta) = A_i \cdot \nabla_\theta \log P(\tau_i|\theta)$. The log-probability of the trajectory is: The gradient for each parameter is calculated as follows:

- **For θ_s (Sampling Policy):**
$\nabla_{\theta_s} J_{\text{Reward}} = A_i \cdot$
$= 0.5 \cdot [-\sigma(\theta_s) + (1 - \sigma(\theta_s))]$
$= 0.5 \cdot [-0.5987 + 0.4013] = \mathbf{-0.0987}$
- **For $\theta_{d,W}$ (Decision Policy):**
$\nabla_{\theta_{d,W}} J_{\text{Reward}} = A_i \cdot \nabla_{\theta_{d,W}} \log P(\text{RESAMPLE}|W, \theta_{d,W})$
$= 0.5 \cdot (1 - \sigma(\theta_{d,W}))$
$= 0.5 \cdot (1 - 0.8022) = \mathbf{+0.0989}$
- **For $\theta_{d,C}$ (Decision Policy):**
$\nabla_{\theta_{d,C}} J_{\text{Reward}} = A_i \cdot \nabla_{\theta_{d,C}} \log P(\text{STOP}|C, \theta_{d,C})$
$= 0.5 \cdot (1 - \sigma(\theta_{d,C}))$
$= 0.5 \cdot (1 - 0.9002) = \mathbf{+0.0499}$

The advantage A_i is applied as a scalar multiplier to all responsible parameters. The resulting gradients are of a similar order of magnitude, demonstrating a *symmetric push* for improvement.

Part 2: Asymmetric Gradient Drag from the KL Penalty

Next, we analyze the gradient from the KL penalty. The asymmetry arises because the immediate penalty for a sampling action is explicitly weighted by its length, whereas the penalty for a decision action is not.

Calculating KL Penalties and Q-Values We first calculate the immediate KL penalty (d_k) for each step in our trajectory. This measures the "information loss" when the current policy deviates from the reference policy.

- **Step 1 - Sample W:** $d_1^{sample} = 8 \cdot \left(-\log \frac{0.4013}{0.4256}\right) = 8 \cdot (0.0588) \approx +0.4704$
- **Step 1 - Resample:** $d_1^{decision} = -\log \frac{0.8022}{0.7685} \approx -0.0429$
- **Step 2 - Sample C:** $d_2^{sample} = 8 \cdot \left(-\log \frac{0.5987}{0.5744}\right) = 8 \cdot (-0.0415) \approx -0.3320$
- **Step 2 - Stop:** $d_2^{decision} = -\log \frac{0.9002}{0.8808} \approx -0.0218$

Next, we compute the state-action values ($Q^{\pi,KL}$) by working backward from the end of the trajectory (with discount factor $\gamma = 1$).

- **Value of final action (STOP):** $Q_{KL}(s_2, \text{STOP}) = d_2^{decision} = -0.0218$

- **Value of preceding action (Sample C):** $Q_{KL}(s_1, \text{sample } C) = d_2^{sample} + Q_{KL}(s_2, \text{STOP}) = -0.3320 - 0.0218 = -0.3538$

- **Value of preceding action (RESAMPLE):** $Q_{KL}(s_1, \text{RESAMPLE}) = d_1^{decision} + Q_{KL}(s_1, \text{sample } C) = -0.0429 - 0.3538 = -0.3967$

- **Value of first action (Sample W):** $Q_{KL}(s_0, \text{sample } W) = d_1^{sample} + Q_{KL}(s_1, \text{RESAMPLE}) = 0.4704 - 0.3967 = +0.0737$

Gradient Calculation The gradient of the KL penalty objective is proportional to $\sum_t \nabla_\theta \log \pi_t \cdot Q_{KL,t}$. This gradient acts as a "drag" force, pulling the policy back toward the reference.

- **For θ_s (Sampling Policy):**

$$\begin{aligned}\nabla_{\theta_s} J_{\text{KL}} &\propto \nabla_{\theta_s} \log P(W|\theta_s) \cdot Q_{\text{KL}}(s_0, W) \\ &\quad + \nabla_{\theta_s} \log P(C|\theta_s) \cdot Q_{\text{KL}}(s_1, C) \\ &\propto (-\sigma(\theta_s)) \cdot (0.0737) + (1 - \sigma(\theta_s)) \cdot (-0.3538) \\ &\propto (-0.5987) \cdot (0.0737) + (0.4013) \cdot (-0.3538) \\ &= -0.0441 - 0.1420 = \mathbf{-0.1861}\end{aligned}$$

- **For $\theta_{d,W}$ (Decision Policy):**

$$\begin{aligned}\nabla_{\theta_{d,W}} J_{\text{KL}} &\propto \nabla_{\theta_{d,W}} \log P(\text{RESAMPLE}|W) \cdot Q_{\text{KL}}(s_1, \text{RESAMPLE}) \\ &\propto (1 - \sigma(\theta_{d,W})) \cdot (-0.3967) \\ &= (0.1978) \cdot (-0.3967) = \mathbf{-0.0785}\end{aligned}$$

- **For $\theta_{d,C}$ (Decision Policy):**

$$\begin{aligned}\nabla_{\theta_{d,C}} J_{\text{KL}} &\propto \nabla_{\theta_{d,C}} \log P(\text{STOP}|C) \cdot Q_{\text{KL}}(s_2, \text{STOP}) \\ &\propto (1 - \sigma(\theta_{d,C})) \cdot (-0.0218) \\ &= (0.0998) \cdot (-0.0218) = \mathbf{-0.0022}\end{aligned}$$

The results are starkly different from the reward gradients. The gradient magnitude for the sampling policy, $|\nabla_{\theta_s} J_{KL}| \approx 0.186$, is more than double that for the first decision parameter ($|\nabla_{\theta_{d,W}} J_{KL}| \approx 0.079$) and over 80 times larger than for the second ($|\nabla_{\theta_{d,C}} J_{KL}| \approx 0.002$). This is the *asymmetric drag*.

C.2 Synthesis: The Net Parameter Update

The final update to the policy parameters is the sum of the gradients from the surrogate reward (the "push") and the KL penalty (the "drag").

D Gradient Attribution Properties of SFT and DFT

We analyze the gradient attribution properties of Supervised Fine-Tuning (SFT) and Dynamic Fine-Tuning (DFT) within our two-stage decision-sampling framework. We show that SFT exhibits imbalanced gradient attribution due to policy-entangled Q-values, while DFT removes this entanglement, achieving improved (though not perfectly balanced) gradient attribution.

D.1 SFT as Policy Gradient with Implicit Reward

Lemma 3. *The SFT gradient is equivalent to a policy gradient with implicit per-token reward:*

$$\nabla_\theta \mathcal{L}_{SFT} = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\sum_{k=1}^{T} \sum_{j=1}^{L_k} \nabla_\theta \log \pi_{\textit{sample}}(\textit{token}_{k,j} \mid \cdot) + \sum_{k=1}^{T} \nabla_\theta \log \pi_d(a_k \mid s_k) \right] \tag{16}$$

where the implicit reward at each token is $r_t = 1$, but weighted by $1/\pi_\theta$ in the gradient (since $\nabla_\theta \log \pi = \frac{1}{\pi} \nabla_\theta \pi$).

D.2 SFT: Imbalanced gradient attribution via Policy-Entanglement

Theorem D.1 (SFT has Imbalanced gradient attribution)**.** *For SFT, the implicit Q-values are:*

$$\begin{aligned} Q^{SFT}_{\textit{sample}}(s_{k-1}, \textit{token}_{k,j}) &= \underbrace{\sum_{j'=j+1}^{L_k} \frac{1}{\pi_{\textit{sample}}(\textit{token}_{k,j'})}}_{\textit{remaining tokens in step } k} + \underbrace{\frac{1}{\pi_d(a_k \mid s_k)}}_{\textit{decision at step } k} \\ &\quad + \underbrace{\sum_{k'=k+1}^{T} \left(\sum_{j'=1}^{L_{k'}} \frac{1}{\pi_{\textit{sample}}(\textit{token}_{k',j'})} + \frac{1}{\pi_d(a_{k'} \mid s_{k'})} \right)}_{\textit{future steps}} \\ Q^{SFT}_d(s_k, a_k) &= \sum_{k'=k+1}^{T} \left(\sum_{j'=1}^{L_{k'}} \frac{1}{\pi_{\textit{sample}}(\textit{token}_{k',j'})} + \frac{1}{\pi_d(a_{k'} \mid s_{k'})} \right) \end{aligned} \tag{17}$$

*These Q-values exhibit **policy-entanglement**:*

- *$Q^{SFT}_{\textit{sample}}$ depends on π_d through the $1/\pi_d(a_k|s_k)$ and $1/\pi_d(a_{k'}|s_{k'})$ terms.*

- *Q^{SFT}_d depends on $\pi_{\textit{sample}}$ through the $1/\pi_{\textit{sample}}(\textit{token}_{k',j'})$ terms.*

Proof. The SFT gradient can be written in policy gradient form:

$$\nabla_\theta \mathcal{L}_{\text{SFT}} = \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot Q^{\text{SFT}}(s_t, a_t) \tag{18}$$

where $Q^{\text{SFT}}(s_t, a_t)$ represents the "future value" from taking action a_t at state s_t. Since each token contributes gradient $\nabla_\theta \log \pi = \nabla_\theta \pi / \pi$, the implicit reward is effectively $1/\pi$ weighted. Working backwards from the terminal state:

Table 3: Breakdown of KL Penalty Gradient Attribution

Step	Action	d_k (Penalty)	$V_{KL}(s')$ (Future)	Q_{KL} (Total)	$\nabla_\theta \log \pi$	Parameter	Contribution
2	STOP	-0.0218	0	**-0.0218**	$1 - \sigma(\theta_{d,C}) = 0.0998$	$\theta_{d,C}$	-0.0022
2	Sample C	-0.3320	-0.0218	**-0.3538**	$1 - \sigma(\theta_s) = 0.4013$	θ_s	-0.1420
1	RESAMPLE	-0.0429	-0.3538	**-0.3967**	$1 - \sigma(\theta_{d,W}) = 0.1978$	$\theta_{d,W}$	-0.0785
1	Sample W	+0.4704	-0.3967	**+0.0737**	$-\sigma(\theta_s) = -0.5987$	θ_s	-0.0441

Table 4: Summary of Gradient Attribution and Net Update

Parameter	Policy	Reward (Push)	KL (Drag)	Net Gradient	Interpretation
θ_s	π_{sample}	-0.0987	-0.1861	**-0.2848**	Heavily regularized; stable.
$\theta_{d,W}$	π_d	+0.0989	-0.0785	**+0.0204**	Learning signal overcomes drag.
$\theta_{d,C}$	π_d	+0.0499	-0.0022	**+0.0477**	Learning driven by reward.

At the final decision (STOP at step T):

$$Q_d^{\text{SFT}}(s_T, \text{STOP}) = 0 \tag{19}$$

At the final sampling step T, token j:

$$Q_{\text{sample}}^{\text{SFT}}(s_{T-1}, \text{token}_{T,j}) = \sum_{j'=j+1}^{L_T} \frac{1}{\pi_{\text{sample}}(\text{token}_{T,j'})} + \frac{1}{\pi_d(\text{STOP}|s_T)} \tag{20}$$

Recursively, for step $k < T$:

$$\begin{aligned} Q_d^{\text{SFT}}(s_k, \text{RESAMPLE}) &= Q_{\text{sample}}^{\text{SFT}}(s_k, \text{token}_{k+1,1}) \\ Q_{\text{sample}}^{\text{SFT}}(s_{k-1}, \text{token}_{k,j}) &= \sum_{j'=j+1}^{L_k} \frac{1}{\pi_{\text{sample}}(\text{token}_{k,j'})} + \frac{1}{\pi_d(a_k \mid s_k)} + Q_d^{\text{SFT}}(s_k, a_k) \end{aligned} \tag{21}$$

The policy-entanglement is evident: $Q_{\text{sample}}^{\text{SFT}}$ contains $1/\pi_d$ terms, and Q_d^{SFT} contains $1/\pi_{\text{sample}}$ terms through the recursive dependency. These Q-values change as π_θ updates during training, preventing clean credit assignment. □

Corollary 2. *No unified sufficient statistic Φ exists for SFT. Following the logic of Theorem 3.2, define:*

$$\Phi_{sample}^{SFT}(s_k) = f\left(\frac{1}{\pi_{sample}}, \frac{1}{\pi_d}, L\right) \tag{22}$$

$$\Phi_d^{SFT}(s_k) = g\left(\frac{1}{\pi_{sample}}, \frac{1}{\pi_d}, L\right) \tag{23}$$

Both depend on the full policy π_θ, making $\Phi_{sample}^{SFT} \neq \Phi_d^{SFT}$ in general.

D.3 DFT: Removing Policy-Entanglement

Dynamic Fine-Tuning rescales each token's contribution by its probability:

$$\begin{aligned} \mathcal{L}_{\text{DFT}}(\theta) = \mathbb{E}_{\tau\sim\mathcal{D}}\Bigg[&\sum_{k=1}^{T}\sum_{j=1}^{L_k} \pi_{\text{sample}}(\text{token}_{k,j}) \cdot \log \pi_{\text{sample}}(\text{token}_{k,j}) \\ &+ \sum_{k=1}^{T} \pi_d(a_k \mid s_k) \cdot \log \pi_d(a_k \mid s_k)\Bigg] \end{aligned} \tag{24}$$

Lemma 4 (Cancellation of $1/\pi$ weighting)**.** *The DFT gradient satisfies:*

$$\pi \cdot \nabla_\theta \log \pi = \pi \cdot \frac{\nabla_\theta \pi}{\pi} = \nabla_\theta \pi \tag{25}$$

Therefore:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{DFT} = \mathbb{E}_{\tau\sim\mathcal{D}}\Bigg[&\sum_{k=1}^{T}\sum_{j=1}^{L_k} \nabla_\theta \pi_{sample}(token_{k,j}) \\ &+ \sum_{k=1}^{T} \nabla_\theta \pi_d(a_k \mid s_k)\Bigg] \end{aligned} \tag{26}$$

The $1/\pi$ weighting is exactly canceled.

Theorem D.2 (DFT Removes Policy-Entanglement)**.** *For DFT, the implicit Q-values become policy-independent:*

$$\begin{aligned} Q_{sample}^{DFT}(s_{k-1}, token_{k,j}) &= (L_k - j)\cdot c + c + \sum_{k'=k+1}^{T} (L_{k'} + 1)\cdot c \\ Q_d^{DFT}(s_k, a_k) &= \sum_{k'=k+1}^{T} (L_{k'} + 1)\cdot c \end{aligned} \tag{27}$$

where c is a constant independent of π_θ.

Proof. With the $1/\pi$ weighting cancelled, each token contributes constant implicit reward c. Working backwards:

At the final decision:

$$Q_d^{\text{DFT}}(s_T, \text{STOP}) = 0 \tag{28}$$

At step T, token j:

$$Q_{\text{sample}}^{\text{DFT}}(s_{T-1}, \text{token}_{T,j}) = (L_T - j)\cdot c + c \tag{29}$$

where $(L_T - j) \cdot c$ accounts for remaining sampling tokens and c accounts for the final decision. Recursively, for step $k < T$:

$$Q_d^{\text{DFT}}(s_k, \text{RESAMPLE}) = L_{k+1} \cdot c + c + Q_d^{\text{DFT}}(s_{k+1}, a_{k+1}) \tag{30}$$

Expanding the recursion:

$$Q_d^{\text{DFT}}(s_k, a_k) = \sum_{k'=k+1}^{T} L_{k'} \cdot c + (T-k) \cdot c = \sum_{k'=k+1}^{T} (L_{k'} + 1) \cdot c \tag{31}$$

Crucially:

- $Q_{\text{sample}}^{\text{DFT}}$ does not depend on π_d.
- Q_d^{DFT} does not depend on π_{sample}.

The Q-values are now policy-independent. □

D.4 DFT: Remaining Length Asymmetry

Although DFT removes policy-entanglement, length asymmetry persists:

$$Q_d^{\text{DFT}}(s_k, a_k) = c \cdot \sum_{k'=k+1}^{T} (L_{k'} + 1) \approx c \cdot \sum_{k'=k+1}^{T} L_{k'} \tag{32}$$

Since $L_{k'} \gg 1$ (sampling sequences contain hundreds of tokens while decisions are single tokens), we have:

$$Q_d^{\text{DFT}} \sim O\left(\sum_{k'} L_{k'}\right) \gg Q_{\text{sample,per-token}}^{\text{DFT}} \sim O(1) \tag{33}$$

The decision policy gradient is weighted by accumulated future sequence lengths, even though each decision is a single token. This creates asymmetric regularization similar to (but weaker than) the KL penalty analyzed in Theorem 3.2.

D.5 Graphical Illustration

We parameterize a minimal two-stage process with three learnable logits: θ_s governing sampling accuracy, $\theta_{d|C}$ governing stop decisions given correct answers, and $\theta_{d|W}$ governing resample decisions given incorrect answers.

Figure 3[15] displays the training dynamics under the combined GRPO objective. The center panels decompose gradient contributions: the surrogate reward generates comparable gradient magnitudes across all parameters (center-left), while the KL penalty produces gradients for sampling that dominate decision gradients by nearly an order of magnitude (center-right). This confirms the $O(L_k)$ versus $O(1)$ asymmetry established above in the section. The bottom-right panel quantifies this directly—gradient attribution remains symmetric for the surrogate reward but exceeds 2:1 (sampling versus decision) for the KL penalty throughout training.

The net effect (bottom-left) is differential learning: π_d parameters receive sustained positive gradients while π_{sample} is heavily regularized toward the reference.

[15]The cyclical pattern observed in the training process is because of reference model updating.

E Task Description and Experiment Hyperparameters

E.1 Task Description

We evaluate models on multi-digit multiplication tasks, denoted as $m \times n$ where m and n indicate the number of digits in each operand. For example, a 3×4 task involves multiplying a 3-digit number by a 4-digit number (e.g., 123×4567).

Each problem is presented to the model as a natural language query:

> Calculate [operand1] * [operand2].
> Think step by step.

An answer is scored as correct if and only if the final numerical output exactly matches the ground truth product. Intermediate reasoning steps are not evaluated for correctness.

E.2 Dataset Construction

Training Data. We construct 20,000 training examples for the in-distribution tasks (4×5 and 5×4 multiplication). Operands are sampled uniformly at random within the specified digit range: for an m-digit operand, we sample integers from $[10^{m-1}, 10^m - 1]$. The training set is balanced equally between 4×5 and 5×4 tasks.

Test Data. For each difficulty level (3×3 through 3×9), we construct a test set of 100 examples using the same uniform sampling procedure. No filtering or balancing is applied for the results reported in Table 1. For the calibration analysis in Figure **??**, we exclude samples that exceed the 8,192 token generation limit.

SFT Data Variants. We construct two SFT training sets:

- **SFT (no reflection):** Each example consists of a query paired with a chain-of-thought so-

Property	SFT	DFT	RL (Surrogate Reward)
Q_d depends on π_{sample}	Yes (via $1/\pi_{\text{sample}}$)	No	No
Q_{sample} depends on π_d	Yes (via $1/\pi_d$)	No	No
Q_d depends on future lengths $L_{k'}$	Yes (weighted by $1/\pi$)	Yes (constant weight)	No
Sufficient statistic	$\Phi^{\text{SFT}}_{\text{sample}} \neq \Phi^{\text{SFT}}_{d}$	$\Phi^{\text{DFT}}_{\text{sample}} \neq \Phi^{\text{DFT}}_{d}$	$\Phi = \gamma^{\sum len} A_i$ (unified)
Gradient attribution	Imbalanced	Improved	Balanced

Table 5: Comparison of gradient attribution properties.

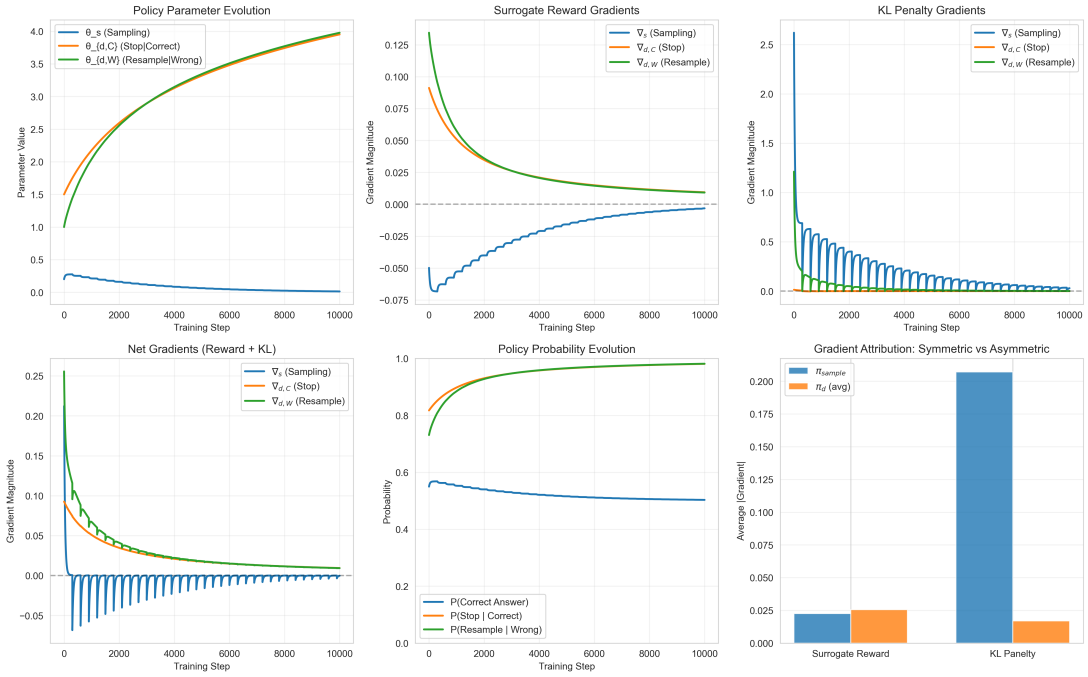


Figure 3: Single Parameter Illustration

lution leading directly to the answer, without retry patterns.

- **SFT (reflection):** Each example consists of a query paired with a trajectory that may include multiple attempts, explicit error detection, and correction steps before arriving at the final answer.

Trajectory generation procedures are provided in the replication repository.

E.3 Model and Training Details

All experiments use Qwen2.5-7B-Instruct (Qwen et al., 2025) as the base model.

SFT Training. Supervised fine-tuning is conducted using LLaMA-Factory (Zheng et al., 2024). Hyperparameters are as follows:

Hyperparameter	Value
Optimizer	Adam
Batch size	128
Learning rate	1×10^{-5}
Epochs	4.0
Training examples	20,000

Table 6: SFT training hyperparameters.

RL Training. Reinforcement learning is conducted using verl (Sheng et al., 2024) with Group Relative Policy Optimization (GRPO). Hyperparameters are as follows:

Hyperparameter	Value
Algorithm	GRPO
Learning rate	1×10^{-6}
Train batch size	512
PPO mini-batch size	128
Rollouts per query (n)	8
KL loss coefficient	0.001
KL loss type	low_var_kl
Max prompt length	1,024 tokens
Max response length	8,192 tokens
Total epochs	2
Gradient checkpointing	Enabled

Table 7: RL training hyperparameters.

E.4 Evaluation Protocol

All models are evaluated with the following settings:

- **Sampling temperature:** 1.0
- **Number of attempts:** 1 (single generation per query)
- **Accuracy metric:** $\frac{\text{Number of correct answers}}{\text{Total number of questions}}$

Confidence Intervals. Confidence intervals for observed accuracy in Table 1 are computed using the normal approximation to the binomial distribution at the 95% level. Confidence intervals for predicted accuracy in Figure 1a and Figure 1b are obtained via bootstrapping: we resample the test set with replacement 100 times, compute the predicted accuracy for each bootstrap sample using the calibrated p_s, $p_{d|C}$, and $p_{d|W}$ parameters, and report the 2.5th and 97.5th percentiles of the resulting distribution.

E.5 Compute Resources

Each experiment is conducted on a cloud server equipped with a single NVIDIA H100 GPU.