

Interleaved Reasoning for Large Language Models via Reinforcement Learning

Anonymous ACL submission

Abstract

Long chain-of-thought (CoT) significantly enhances the reasoning capabilities of large language models (LLMs). However, extensive reasoning traces lead to inefficiencies and increased time-to-first-token (TTFT). We propose a training paradigm that uses only reinforcement learning (RL) to guide reasoning LLMs to *interleave thinking and answering* for multi-hop questions. We observe that models inherently possess the ability to perform interleaved reasoning, which can be further enhanced through RL. We introduce a simple yet effective reward scheme to incentivize correct intermediate steps, guiding the policy model toward correct reasoning paths by leveraging intermediate signals generated during interleaved reasoning. Extensive experiments across five diverse datasets and three RL algorithms (PPO, GRPO, and REINFORCE++) demonstrate consistent improvements over traditional think-answer reasoning, without requiring external tools. Our method improves final task accuracy and overall efficiency by enabling more effective credit assignment during RL. Specifically, our approach achieves a 12.5% improvement in Pass@1 accuracy, while reducing overall reasoning length by 37% and TTFT by over 80% on average. Furthermore, our method, trained solely on question answering and logical reasoning datasets, exhibits strong generalization to complex reasoning datasets such as MATH, GPQA, and MMLU. Additionally, we conduct in-depth analysis to reveal several valuable insights into conditional reward modeling.

1 Introduction

Reasoning large language models (LLMs) (Jaech et al., 2024; Guo et al., 2025) have demonstrated advanced capabilities in complex multi-hop tasks through long chain-of-thought (CoT) (Wei et al., 2022). However, the standard “*think-answer*” paradigm, where models must complete the full reasoning trace before generating answers, introduces

two critical limitations. *First*, it significantly increases time-to-first-token (TTFT), taking seconds or minutes for answer generation. This breaks the interaction flow in real-time AI applications such as conversational assistants, resulting in poor user experience. *Second*, by delaying answer generation until the reasoning concludes, models may follow incorrect intermediate steps, propagate errors, and lead to inaccurate final answers and reasoning inefficiencies such as overthinking (Chen et al., 2024; Sui et al., 2025).

Humans naturally provide incremental feedback during conversations, signaling understanding even as they formulate complete responses. Decomposing a complex problem into smaller steps is also the de-facto approach for many reasoning tasks in LLMs (Wei et al., 2022; Khot et al., 2022; Zhou et al., 2022; Besta et al., 2023). However, current reasoning LLMs treat thinking and answering as strictly sequential processes – answers are available only after reasoning concludes.

Current reinforcement learning (RL) paradigms often treat intermediate reasoning traces as byproducts or unstructured chatter (Kumar et al., 2024; Hou et al., 2025; Guo et al., 2025). However, we argue that for multi-hop reasoning tasks, structured intermediate answers are valuable on several fronts. *First*, unstructured reasoning streams often contain exploratory and potentially contradictory thoughts, and users rarely have the bandwidth to examine such lengthy reasoning traces (Treude and Kula, 2025). Yet these traces may already include partial conclusions that can be useful; clearly presenting such conclusions early can enhance the interaction experience (Liu et al., 2025). *Second*, most production reasoning LLM APIs do not stream reasoning traces (Comanici et al., 2025; OpenAI, 2025). Making the problem-solving process visible in real time provides transparency and helps users verify the model’s final output. *Third*, these partial conclusions can also be utilized as dense supervi-

Question: Who was the director of the film that won the Academy Award for Best Picture six years after the Berlin Wall fell?

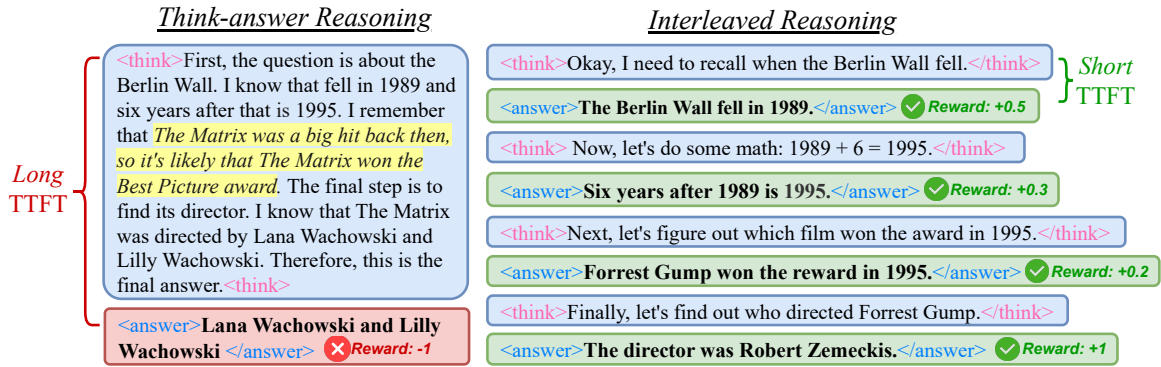


Figure 1: Standard think-answer reasoning (left) completes the full chain-of-thought before generating an answer, resulting in high TTFT and making credit assignment difficult during training when intermediate steps contain errors (highlighted in yellow). Interleaved reasoning (right) alternates between thinking and answering, enabling structured, easy-to-verify reward signals for better credit assignment and significantly reducing TTFT.

sion signals to further improve model’s reasoning during training (Lightman et al., 2023; Cui et al., 2025). Ideally, models should iteratively switch between “think” and “answer” modes based on their understanding of the problem and its complexity. However, effectively applying RL to induce such behavior remains challenging. It is unclear whether models can learn and generalize across various complex tasks. Moreover, effectively leveraging simple, rule-based rewards to detect sufficient intermediate signals during training is largely under-explored.

To address these challenges, we introduce *interleaved reasoning*, a RL training method that enables LLMs to interleave thinking and answering. As shown in Figure 1, an interleaved reasoning model generates concrete and informative intermediate answers during reasoning, while providing reward signals for training. We conduct comprehensive experiments on three RL algorithms (PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), and REINFORCE++ (Hu, 2025)) and five diverse datasets (K&K (Xie et al., 2024), Musique (Trivedi et al., 2022), MATH (Hendrycks et al., 2021), GPQA (Rein et al., 2023), MMLU (Hendrycks et al., 2020)) and found that LLMs are inherently capable of answering questions in an interleaved manner. We introduce a simple yet effective conditional reward scheme that enables models to achieve an average 12.5% relative improvement in Pass@1 accuracy, shortening reasoning length by 37%, and reducing TTFT by over 80%. We conduct comprehensive evaluations to show that the model produces meaningful intermediate steps

and generalizes to tasks without intermediate supervision. Furthermore, our analysis offers several key insights into conditional reward modeling.

2 Related Work

Reward Modeling for LLM Reasoning. In Reinforcement Learning (RL), reward modeling is an effective mechanism for steering models toward desired behaviors (Silver et al., 2021). These rewards typically fall into two categories: Outcome Reward Models (ORMs), which evaluate the final result, and Process Reward Models (PRMs), which provide dense feedback on intermediate steps (Lightman et al., 2024; Uesato et al., 2022; Wang et al., 2024). While PRMs offer granular guidance, they often require labor-intensive human annotations (Lightman et al., 2024; Uesato et al., 2022), which can introduce reward-hacking risks (Rafailov et al., 2024) and increase complexity (Guo et al., 2025; Wang et al., 2024). Conversely, recent breakthroughs like DeepSeek-R1 (Guo et al., 2025) demonstrate that simple, rule-based ORM can significantly enhance performance on complex tasks. In this work, we leverage the concept of PRM, but instead of relying on a separate learned model, we use a simple rule-based reward to capture intermediate signals. Unlike PRMs that generate feedback at each step during rollout, our method operates more like an ORM while granting partial credit to the intermediate answers. Discussions on the distinction between PRM and our method can be found in Appendix A.

LLM Reasoning and Efficiency. Recent developments in RL enable models like OpenAI

o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) to generate long CoT to improve their reasoning abilities. This paradigm towards longer reasoning results in inefficiencies and significantly increased latency and time-to-first-token (TTFT). Recent studies address this issue by proposing more concise reasoning through techniques such as inference-time adjustments (Xu et al., 2025b,a; Kimi, 2025), length control RL (Aggarwal and Welleck, 2025; Fatemi et al., 2025; Yuan et al., 2025), or additional finetuning (Luo et al., 2025). Interleaving reasoning is also a newly emerging research area. Recent work mainly focuses on leveraging *external* tools such as search engines (Jin et al., 2025; Song et al., 2025; Li et al., 2025b) to enhance LLM reasoning. In contrast, we focus on the model’s ability to generate concrete intermediate answers, which can later be used as additional reward signals for training.

3 Methodology

3.1 Multi-hop Question Answering

We conceptualize answering a multi-hop question as a sequence of resolved intermediate sub-problems. A sub-answer is a user-facing partial conclusion that the model confidently derives at a given stage. The model *should* output a sub-answer when it determines that a self-contained part of the problem has been solved. For example, a sub-answer might resolve the first sub-problem and guide the next - such as an intermediate calculation in a multi-step math problem.

Thinking vs. Answering. From a philosophical perspective, thinking constitutes an integral component of answering. However, from a user experience standpoint, a model’s answer effectively *begins* when the first valid answer token is generated. Based on their utility to the user, we define *thinking* as a private internal reasoning process that is not accessible or useful to the user. In contrast, *answering* is the generation of public, finalized conclusions that constitute a meaningful response to the user’s question.

Formally, given user input x requiring N reasoning steps, the policy model π_θ produces a sequence y that alternates between thinking and answering segments. Let $k \in \{1, \dots, N\}$ index the steps. We denote the thinking segment by $y_{\text{think}}^{(k)}$ and the corresponding answer segment by $y_{\text{answer}}^{(k)}$. The in-

You are a helpful assistant. You reason through problems step by step before providing an answer. You conduct your reasoning within `<think></think>` and share partial answers within `<answer></answer>` as soon as you become confident about the intermediate results. You continue this pattern of `<think></think> <answer></answer> <think></think> <answer></answer>` until you reach the final answer. User: **prompt**. Assistant:

Table 1: Template for interleaving thinking and answering. **prompt** will be replaced with the specific reasoning question during training.

terleaved generation thus is:

$$y = y_{\text{think}}^{(1)} \circ y_{\text{answer}}^{(1)} \circ y_{\text{think}}^{(2)} \circ y_{\text{answer}}^{(2)} \circ \dots \circ y_{\text{answer}}^{(N)} \quad (1)$$

where \circ denotes concatenation. The final answer to the original question is $y_{\text{answer}}^{(N)}$, whereas the preceding answer segments $\{y_{\text{answer}}^{(k)}\}_{k=1}^{N-1}$ are intermediate answers. The thinking segments $y_{\text{think}}^{(k)}$ guide the reasoning process but are not part of the user-visible answer for the TTFT calculation until the subsequent answer segment $y_{\text{answer}}^{(k)}$ is produced.

3.2 RL for Interleaved Reasoning

We formulate the task of learning interleaved reasoning as a RL problem. During RL, the policy model π_θ generates sequences that maximize an expected reward. The objective function is:

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta} [r(x, y)] - \beta D_{\text{KL}}[\pi_\theta || \pi_{\text{ref}}] \quad (2)$$

where \mathcal{D} is the training dataset, $\pi_{\text{ref}}(y | x)$ is the reference policy model, β is the KL divergence coefficient, and $r(x, y)$ is the reward function. Detailed hyperparameter choices are discussed in Appendix B. We discuss the policy optimization in §4.1 and compare the performance of different RL algorithms in §5.2. After training, the model should have learned how to dynamically switch between them based on the given task at each step.

Interleaved Instruction. To guide the model in learning the interleaved behavior, we use a specific instruction template during training and inference. Following DeepSeek-R1 (Guo et al., 2025), we use two special tags (`<think></think>` and `<answer></answer>`) to instruct the model to perform reasoning and provide answers within each tag, respectively (Table 1). We use DeepSeek-R1’s template for think-answer reasoning (Appendix B.3).

3.2.1 Reward Design

To effectively train the model to reason in the interleaved format, we utilize three rewards: the **format**

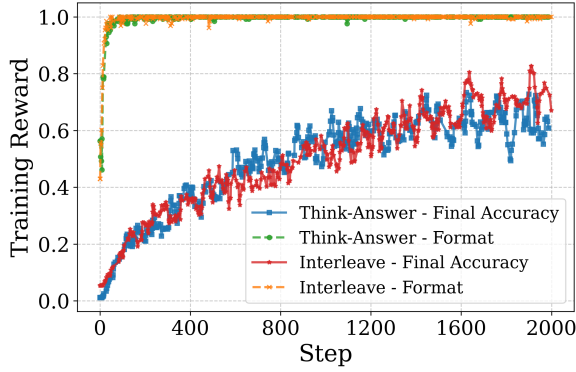


Figure 2: The format reward plateaus rapidly during training, significantly faster than the accuracy reward, suggesting that LLMs inherently possess the ability to reason in structured, interleaved patterns.

reward assesses whether the interleaved format is correctly followed and properly completed; the **final accuracy** reward evaluates the correctness of the final answer; and the **conditional intermediate accuracy** reward (or intermediate reward) provides additional rewards for correct intermediate answers, applied conditionally based on training progress. Following previous work (Jin et al., 2025), our reward design avoids complex neural reward models, instead focusing on simple rule-based rewards that provide clear and consistent feedback without requiring separate reward model training. We discuss the conditions to apply the intermediate reward in §3.2.2. More details about the rewards can be found in Appendix C.1.

Language Models Are Quick Format Learners.

Our initial experiments revealed that models inherently possess the ability to interleave thinking and answering. Base models (without RL training) can generate intermediate answers by directly applying the interleaved template, with some reduced accuracy. Additionally, models rapidly learn the structural format. As illustrated in Figure 2, the format reward for both reasoning methods quickly plateaus, whereas the accuracy reward continues to improve. We also observe that both reasoning methods achieve similar final accuracy reward during training. The finding suggests the main challenge is not stylistic adherence but rather enhancing the quality of their thought processes for different reasoning tasks. This motivates our focus on the reasoning itself: not for its structure per se, but for its potential to improve the model’s reasoning by leveraging its explicit intermediate outputs as learning signals.

3.2.2 Conditional Rewards

Our initial experiments show that directly applying intermediate reward during training often leads to suboptimal results, as the model may prioritize local correctness at the expense of final solution correctness (§5.4). To effectively leverage the benefit of intermediate answers beyond shorter TTFT, we design a conditional reward strategy that incentivizes the model to generate correct intermediate answers early, in order to guide the reasoning toward the correct final answer. We apply intermediate rewards only when the model demonstrates foundational competence and shows meaningful learning progress during training.

Specifically, the rewards are applied when three conditions are met: (1) the final answer is correct, (2) the output format is valid, and (3) the model shows improvement in the current training batch compared to previous one. The core idea is to ensure that the model first masters the primary objective before optimizing for the sub-tasks of generating correct intermediate steps. Formally, the conditional intermediate reward is defined as:

$$r_{\text{intermediate}}(x, y) = \mathbb{1}(\mathcal{G}) \cdot \sum_{k=1}^{N-1} f(y_{\text{ans}}^{(k)}), \quad (3)$$

where the gating condition \mathcal{G} is defined as:

$$\mathcal{G} = \mathcal{F}(y) \wedge \mathcal{C}(y_{\text{ans}}^{(N)}) \wedge (\text{Acc}_B > \text{Acc}_{B-1}), \quad (4)$$

Here, Acc_B and Acc_{B-1} denote the accuracies of the current and previous training batches, and \mathcal{F} and \mathcal{C} denote format and answer correctness, respectively. The batch accuracy criterion serves as a curriculum indicator, gradually introducing intermediate rewards as training stabilizes (Yuan et al., 2025). The overall reward function is:

$$r(x, y) = r_{\text{format}}(y) + r_{\text{final}}(x, y) + r_{\text{intermediate}}(x, y), \quad (5)$$

where $r_{\text{intermediate}}(x, y)$ is invoked only if all the aforementioned conditions are met. The full reward definitions can be found in Appendix C.

Intermediate Reward. We explore three different approaches to calculate intermediate reward given the conditions described above: **All-or-None**, which requires all intermediate steps to be correct in sequence; **Partial Credit**, which gives partial credit for individual correct intermediate steps; and **Time-Discounted**, which assigns higher rewards to earlier correct intermediate steps while assigning extra rewards to all correct intermediate steps.

Intermediate ground truth is used for the rewards calculation. We compare three reward calculation approaches in §5.4. We also discuss in detail in Appendix D for tasks *without intermediate ground truth* and show that intermediate ground truth are not required for our method. Additionally, despite training only on tasks that intermediate ground truths are available, our method generalizes to other unseen tasks without using any intermediate rewards (§4.2).

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on both in- and out-of-domain datasets. For in-domain datasets, we use **Knights and Knaves (K&K)** (Xie et al., 2024) and **Musique** (Trivedi et al., 2022) for both training and evaluation. K&K is a logical reasoning dataset that requires multi-step reasoning to identify the correct character between knights and knaves. Musique is a multi-hop question answering dataset that requires retrieving and combining information from multiple sources. Both datasets naturally contain subproblems and their ground truth. For out-of-domain datasets that do not have intermediate ground truth, we evaluate on **GPQA** (Rein et al., 2023), **MMLU** (Hendrycks et al., 2020), and **MATH** (Hendrycks et al., 2021). More details about the datasets and the selections can be found in Appendix E.

Models and Baselines. We conduct experiments using Qwen2.5 instruct models with 1.5B and 7B parameters. We evaluate various baselines: **Direct Inference**, where the model generates answers without explicit reasoning steps; **Chain-of-Thought (CoT)** (Wei et al., 2022), where the model performs all reasoning before generating the final answer; **SFT** (Chung et al., 2024), where the model is trained with supervised fine-tuning; **Think-answer**, where we train the same model with the standard think-answer RL methods proposed in Guo et al. (2025). We compare the baselines with two interleaved reasoning approaches: **Interleave**, our base approach without intermediate rewards; and **Interleave + IR**, our main approach with conditional *intermediate rewards* (IR) using time-discounted approach, as described in §3.2.2. For fair evaluation, we use the same setup (e.g., datasets, RL algorithms, etc.) for all RL training.

Evaluation Metrics. We use two key metrics: **Pass@1 accuracy** (How many problems are solved correctly) and **time-to-first-token (TTFT)** (How quickly the model provides answers to users). Following previous work (Meng et al., 2024; Jin et al., 2025), we use Exact Match to calculate the percentage of correct final answers against the ground truth for pass@1 score. For each test instance, we compare the model’s final answer against the ground truth answer after normalization. In conventional settings, TTFT is typically measured in absolute time units (e.g., milliseconds). However, to apply it across different reasoning approaches, we define TTFT as the relative position of the first answer token in the complete response. More details on the evaluation metrics can be found in Appendix F.

Policy Optimization. We experiment with three policy optimization approaches: the traditional Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its two variants, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and REINFORCE++ (Hu, 2025). PPO is typically more stable but requires additional warm-up due to a critic network, whereas GRPO and REINFORCE++ are more sample-efficient but sensitive to hyperparameter choices. We use PPO as our primary training algorithm. To ensure a fair comparison, we train models for up to 2,000 steps and report the best checkpoint for both think-answer and interleaved training. Time-Discounted Intermediate rewards are used for the main experiments.

4.2 Main Results.

Table 2 demonstrates both efficiency and accuracy benefits of interleaved reasoning. The base interleaved approach (Interleave), without using intermediate rewards, maintains comparable Pass@1 accuracy to think-answer reasoning while reducing TTFT by more than 80% on average. When intermediate rewards are applied (Interleave + IR), we observed a 12.5% relative improvement in average Pass@1 accuracy. Moreover, training on *only* the datasets with intermediate ground truth, our method exhibits strong generalization across diverse tasks (GPQA, MMLU, and MATH), maintaining superior accuracy and reduced TTFT without *any* training data from these datasets.

Our method also reduces overall response length by up to 37% compared to think-answer reasoning (§5.3). Furthermore, we conduct both qualitative and quantitative evaluation in Appendix G to show

Methods	K&K [‡]		Musique [‡]		GPQA [†]		MMLU [†]		MATH [†]		Avg.	
	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]
<i>Qwen2.5-1.5B-Instruct</i>												
Direct Inference	0.060	0.000	0.115	0.000	0.051	0.000	0.081	0.000	0.278	0.000	0.117	0.000
CoT	0.097	0.000	0.195	0.000	0.066	0.000	0.167	0.000	0.308	0.000	0.167	0.000
SFT	0.223	0.000	0.290	0.000	0.046	0.000	0.112	0.000	0.263	0.000	0.187	0.000
Think-answer	0.342	0.819	0.675	0.763	0.328	0.929	0.434	0.913	0.323	0.952	0.420	0.875
Interleave	0.357	0.118	0.700	0.210	0.308	0.181	0.429	0.189	0.288	0.163	0.416	0.172
Interleave + IR	0.533	0.132	0.710	0.155	0.489	0.192	0.460	0.211	0.313	0.157	0.501	0.169
<i>Qwen2.5-7B-Instruct</i>												
Direct Inference	0.150	0.000	0.295	0.000	0.157	0.000	0.444	0.000	0.475	0.000	0.304	0.000
CoT	0.230	0.000	0.295	0.000	0.192	0.000	0.495	0.000	0.561	0.000	0.355	0.000
SFT	0.343	0.000	0.425	0.000	0.147	0.000	0.465	0.000	0.460	0.000	0.368	0.000
Think-answer	0.843	0.882	0.705	0.917	0.495	0.923	0.758	0.919	0.712	0.876	0.703	0.903
Interleave	0.803	0.133	0.735	0.155	0.505	0.182	0.769	0.199	0.707	0.173	0.704	0.168
Interleave + IR	0.877	0.129	0.750	0.167	0.551	0.166	0.803	0.178	0.732	0.167	0.743	0.161

Table 2: Comparison between our proposed methods and baselines. [‡] and [†] represents in-domain and out-of-domain datasets, respectively. Higher Pass@1 (↑) is better, while lower TTFT (↓) is better. The best performance is in bold. For the non-reasoning baselines (Direct Inference, CoT, SFT) TTFT is naturally 0.

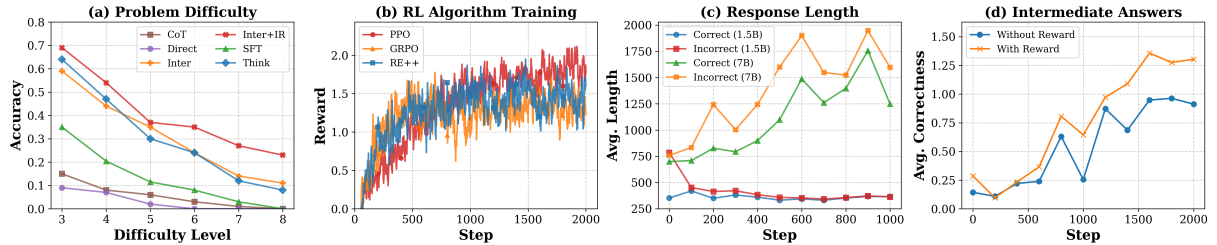


Figure 3: Analysis of interleaved reasoning: (a) Pass@1 accuracy vs. problem difficulty: performance gap widens as K&K problem difficulty increases; (b) Training dynamics: convergence patterns across different RL algorithms; (c) Response length vs. correctness: correct answers are typically shorter across all methods; (d) Intermediate reward impact: percentage of correct intermediate answers increases with conditional intermediate rewards.

that the intermediate answers are meaningful. We also present detailed case studies in Appendix I to demonstrate how models learn to generate intermediate conclusions. These findings combined indicate the effectiveness of interleaved reasoning in enhancing both model accuracy and efficiency.

5 Analysis

5.1 Scaling to Harder Problems.

The K&K dataset contains multiple levels of problem difficulty, with the difficulty increasing as more characters are involved. We train the model with datasets involving three, four, and five characters and evaluate on the full range of difficulties (three through eight; see Appendix E.1 for dataset details). Figure 3(a) shows that the gap between our method and the think-answer baseline widens as the difficulty increases. During logical deduction, the model builds each deduction step upon the previous one; encouraging the model to articulate and produce correct intermediate steps keeps the deductive

chain intact and makes a correct final conclusion more likely. This trend indicates that interleaved reasoning not only offers TTFT speedups but also improves reasoning for harder multi-hop problems.

5.2 RL Algorithms Comparison.

Table 3 shows the performance differences among the three RL algorithms. PPO achieves higher Pass@1 scores for most tasks, though it generally requires more training steps to converge compared to the other two, as shown in Figure 3(b). Conversely, GRPO and REINFORCE++ demonstrate better sample efficiency, reaching competitive performance more rapidly, but they are less stable during training, which aligns with the observation from previous work (Jin et al., 2025). Overall, PPO emerges as the more stable choice for interleaved reasoning, especially when computational resources permit longer training durations, whereas GRPO and REINFORCE++ provide viable alternatives. Note that across all algorithms,

Methods	K&K [‡]		Musique [‡]		GPQA [†]		MMLU [†]		MATH [†]		Avg.	
	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]
<i>GRPO</i>												
Think-answer	0.387	0.878	0.690	0.755	0.333	0.805	0.419	0.795	0.374	0.897	0.441	0.826
Interleave	0.383	0.221	0.650	0.205	0.409	0.151	0.424	0.123	0.313	0.244	0.436	0.189
Interleave + IR	0.473	0.164	0.690	0.132	0.465	0.133	0.455	0.230	0.323	0.198	0.481	0.171
<i>REINFORCE++</i>												
Think-answer	0.347	0.859	0.655	0.794	0.389	0.868	0.424	0.912	0.278	0.751	0.419	0.837
Interleave	0.437	0.202	0.645	0.234	0.270	0.113	0.434	0.163	0.354	0.104	0.428	0.163
Interleave + IR	0.493	0.148	0.720	0.186	0.439	0.123	0.429	0.146	0.348	0.204	0.486	0.161
<i>PPO</i>												
Think-answer	0.342	0.819	0.675	0.763	0.328	0.929	0.434	0.913	0.323	0.952	0.420	0.875
Interleave	0.357	0.118	0.700	0.210	0.308	0.181	0.429	0.189	0.288	0.163	0.416	0.172
Interleave + IR	0.533	0.132	0.710	0.155	0.489	0.192	0.460	0.211	0.313	0.157	0.501	0.169

Table 3: Comparison between different RL algorithms. PPO yields the best average Pass@1 as training steps increase and is more stable during training. GRPO and REINFORCE++ are sampling efficient yet less stable.

Method	Use IR	K&K [‡]		GPQA [†]		MMLU [†]		MATH [†]		Avg.	
		Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]
Delayed intermediate	No	0.287	0.762	0.273	0.805	0.409	0.835	0.298	0.821	0.317	0.806
	Yes	0.323	0.789	0.298	0.812	0.419	0.833	0.283	0.810	0.331	0.811
Interleave	No	0.357	0.118	0.308	0.181	0.429	0.189	0.288	0.163	0.346	0.163
	Yes	0.533	0.132	0.489	0.192	0.460	0.211	0.313	0.157	0.449	0.173

Table 4: Comparison between interleaved reasoning (providing intermediate answers incrementally) versus the delayed version (providing intermediate conclusions only *after* the full reasoning trace, similar to “think-answer”). Interleaved reasoning significantly outperforms the delayed version, which suggests that timely, incremental feedback is crucial.

our method (Interleave + IR) consistently outperforms the think-answer baseline, providing further evidence of its effectiveness.

5.3 Reasoning Length Analysis.

Figure 3(c) shows the response length of interleaved reasoning during training. We observe that 7B and 1.5B models differ in how their response length changes. While both models achieve better performance, the response length of the 7B model grows, whereas that of the 1.5B model becomes shorter. This indicates that response length is not a reliable indicator of performance, aligning with recent findings (Wang et al., 2025; Xie et al., 2025). Table 5 shows the average number of tokens produced by each method between correct and incorrect responses. Our approach consistently produces shorter (up to 37%) responses compared to think-answer reasoning. We also observe that incorrect responses in the think-answer reasoning are notably longer (exceeding 2 \times) than correct ones. Conversely, our interleaved approach yields consistent shorter lengths by enforcing concrete intermediate conclusions that curtail unproductive exploration. This suggests that interleaved reasoning improves

Method	Correct	Incorrect	Overall
Think-Answer	198.1	445.8	401.0
Interleave	207.6	259.9	252.8
Interleave+IR	308.3	380.3	368.7

Table 5: Average number of tokens per response. Our method achieved shorter reasoning responses.

efficiency by fostering early accuracy and minimizing unproductive explorations such as backtracking (see Appendix H for further discussion).

Impact of Intermediate Answers. We investigate how intermediate answers influence model performance and training dynamics. First, as shown in Figure 3(d), applying intermediate rewards during training leads to a clear increase in the number of correct intermediate answers. This indicates that the reward signal effectively encourages the model to produce more accurate sub-answers, which helps steer the model along more reliable reasoning paths. Second, the timing of intermediate answers is critical. Table 4 compares our standard interleave methods with a delayed intermediate variant where intermediate answers are generated only *after* the full reasoning trace and *before* the final answer, both

Methods	K&K [‡]		Musique [‡]		GPQA [†]		MMLU [†]		MATH [†]		Avg.	
	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]	Pass@1 [↑]	TTFT [↓]
No IR	0.357	0.118	0.700	0.210	0.308	0.181	0.429	0.189	0.288	0.163	0.416	0.172
Direct IR	0.313	0.109	0.640	0.194	0.303	0.166	0.409	0.177	0.293	0.150	0.392	0.159
Cond. IR (Partial)	0.498	0.168	0.690	0.190	0.465	0.171	0.439	0.170	0.298	0.161	0.478	0.172
Cond. IR (All)	0.513	0.102	0.695	0.185	0.475	0.162	0.455	0.208	0.308	0.152	0.489	0.162
Cond. IR (Time)	0.533	0.132	0.710	0.155	0.489	0.192	0.460	0.211	0.313	0.157	0.501	0.169

Table 6: Directly applying intermediate reward yields suboptimal performance. Time-discounted conditional intermediate rewards improve interleaved reasoning by incentivizing early correct steps, outperforming direct and other conditional reward methods.

with and without Intermediate Rewards (IR). The delayed intermediate variant shows that generating intermediate answers *early* – not merely having them – drives both lower TTFT and higher Pass@1. Furthermore, the benefits of IR are diminished in the delayed intermediate setting, which suggests that timely, incremental feedback throughout the reasoning process is key to the effectiveness of interleaved reasoning.

5.4 Reward Strategies Comparison.

We investigate the effectiveness of different intermediate reward strategies and present the results in Table 6. Results show that directly applying intermediate rewards (Direct IR) yields lower accuracy compared to not applying intermediate reward at all (No IR). This is likely due to challenges in credit assignment inherent to reinforcement learning, where ambiguous reward signals complicate the attribution of specific actions (Leike et al., 2018). Conditional reward strategies (§3.2.2) significantly mitigate this issue by introducing intermediate rewards only when training is stable. The All-or-None (All) method slightly outperforms Partial Credit (Partial), suggesting that enforcing strict correctness criteria across intermediate steps better supports coherent reasoning paths than rewarding individual correct steps independently. The Time-Discounted (Time) method achieves the best performance. This result indicates that providing higher incentives for early correct reasoning steps effectively guides the model toward accurate reasoning paths.

5.5 Intermediate Reward Distribution.

We visualize how frequently intermediate rewards are applied during training in Figure 4. We observe that intermediate rewards are primarily given in the early stages of training. As training progresses and the batch accuracy threshold rises, the application rate of intermediate rewards decreases. This implies that only a modest amount of intermediate re-

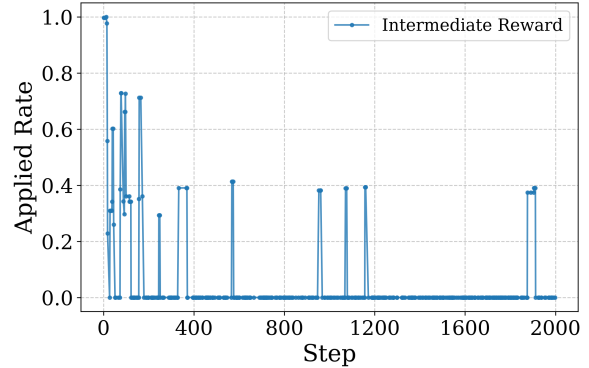


Figure 4: Visualization of intermediate reward application rate during training. The rate decreases as training progresses due to increasing batch accuracy thresholds.

ward is needed to effectively incentivize the model to produce better intermediate steps and ultimately improve final accuracy, especially in the early training stage. The conditional reward strategy thus works as intended: a frequent, always-on intermediate reward is not necessary – a targeted, conditional approach is sufficient to guide the model.

6 Conclusion

In this work, we present interleaved reasoning, a RL training method that enables LLMs to alternate between reasoning and generating structured intermediate answers. Our experiments across five datasets and three RL algorithms show over 80% reduction in TTFT and a 12.5% average increase in Pass@1 accuracy. We propose a simple reward scheme that incentivizes correct intermediate steps and further enhances reasoning ability, enabling the model to generalize well to harder and unseen tasks while reducing reasoning length by up to 37%. Our comprehensive analysis provides several insights into conditional reward modeling and LLM reasoning dynamics. Interleaved reasoning offers a promising path toward more accurate, efficient, and interactive LLMs.

7 Limitations

Our experiments focus on the Qwen2.5 series (1.5B and 7B). We selected these models as they represent the most popular open-weights models for RL training, and we observed consistent robustness across three different RL algorithms (PPO, GRPO, REINFORCE++). Additionally, our current training framework uses datasets with intermediate ground truth as a proof of concept. While we demonstrate that the model generalizes effectively to tasks without intermediate steps, we plan to extend this paradigm to scenarios where intermediate supervision is entirely absent during training for future work.

References

- Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. 2025. The unreasonable effectiveness of entropy minimization in llm reasoning. *arXiv preprint arXiv:2505.15134*.
- Pranjal Aggarwal and Sean Welleck. 2025. **L1: controlling how long A reasoning model thinks with reinforcement learning**. *CoRR*, abs/2503.04697.
- Maciej Besta, Niklas Blach, Vít Kubíček, Michael Gerstenberger, Matthew Gianinazzi, Piotr Nyczyk, and Torsten Hoefler. 2023. **Graph of thoughts: Solving elaborate problems with large language models**. In *AAAI Conference on Artificial Intelligence*.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. **Scaling instruction-finetuned language models**. *J. Mach. Learn. Res.*, 25:70:1–70:53.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

- Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. 2025. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*.
- Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. 2024. **Are we done with mmlu?** *Preprint*, arXiv:2406.04127.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shiron Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. **Measuring massive multitask language understanding**. *ArXiv*, abs/2009.03300.
- 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. *arXiv preprint arXiv:2103.03874*.
- Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*.
- Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.
- Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.
- Tushar Khot, H. Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. **Decomposed prompting: A modular approach for solving complex tasks**. *ArXiv*, abs/2210.02406.
- Kimi. 2025. **Kimi k1.5: Scaling reinforcement learning with llms**. *CoRR*, abs/2501.12599.

667	Aviral Kumar, Vincent Zhuang, Rishabh Agarwal,	model overoptimization in direct alignment algo-	720
668	Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli,	rithms . In <i>Advances in Neural Information Process-</i>	721
669	Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and	<i>ing Systems 38: Annual Conference on Neural In-</i>	722
670	1 others. 2024. Training language models to self-	<i>formation Processing Systems 2024, NeurIPS 2024,</i>	723
671	correct via reinforcement learning. <i>arXiv preprint</i>	<i>Vancouver, BC, Canada, December 10 - 15, 2024.</i>	724
672	<i>arXiv:2409.12917</i> .		
673	Jan Leike, David Krueger, Tom Everitt, Miljan Martic,	David Rein, Betty Li Hou, Asa Cooper Stickland,	725
674	Vishal Maini, and Shane Legg. 2018. Scalable agent	Jackson Petty, Richard Yuanzhe Pang, Julien Di-	726
675	alignment via reward modeling: a research direction.	rani, Julian Michael, and Samuel R. Bowman. 2023.	727
676	<i>arXiv preprint arXiv:1811.07871</i> .	Gpqa: A graduate-level google-proof q&a bench-	728
677	Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey	mark . <i>ArXiv</i> , abs/2311.12022.	729
678	Kuznetsov, and Ivan Oseledets. 2025a. Confidence	John Schulman, Filip Wolski, Prafulla Dhariwal,	730
679	is all you need: Few-shot rl fine-tuning of language	Alec Radford, and Oleg Klimov. 2017. Proxi-	731
680	models. <i>arXiv preprint arXiv:2506.06395</i> .	mal policy optimization algorithms. <i>arXiv preprint</i>	732
681	Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yu-	<i>arXiv:1707.06347</i> .	733
682	tao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	734
683	Dou. 2025b. Webthinker: Empowering large reason-	Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan	735
684	ing models with deep research capability. <i>arXiv</i>	Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-	736
685	<i>preprint arXiv:2504.21776</i> .	math : Pushing the limits of mathematical reason-	737
686	Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Ji-	ing in open language models. <i>arXiv preprint</i>	738
687	axin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu,	<i>arXiv:2402.03300</i> .	739
688	Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 oth-	Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin	740
689	ers. 2025c. From system 1 to system 2: A survey	Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin	741
690	of reasoning large language models. <i>arXiv preprint</i>	Lin, and Chuan Wu. 2024. Hybridflow: A flexible	742
691	<i>arXiv:2502.17419</i> .	and efficient rlhf framework. <i>arXiv preprint arXiv:</i>	743
692	Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri-	<i>2409.19256</i> .	744
693	son Edwards, Bowen Baker, Teddy Lee, Jan Leike,	David Silver, Satinder Singh, Doina Precup, and	745
694	John Schulman, Ilya Sutskever, and Karl Cobbe.	R. Sutton. 2021. Reward is enough . <i>Artif. Intell.</i> ,	746
695	2023. Let's verify step by step. In <i>The Twelfth Inter-</i>	299:103535.	747
696	<i>national Conference on Learning Representations</i> .	Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen,	748
697	Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri-	Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-	749
698	son Edwards, Bowen Baker, Teddy Lee, Jan Leike,	Rong Wen. 2025. R1-searcher: Incentivizing the	750
699	John Schulman, Ilya Sutskever, and Karl Cobbe.	search capability in llms via reinforcement learning.	751
700	2024. Let's verify step by step . In <i>The Twelfth In-</i>	<i>arXiv preprint arXiv:2503.05592</i> .	752
701	<i>ternational Conference on Learning Representations,</i>	Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu	753
702	<i>ICLR 2024, Vienna, Austria, May 7-11, 2024</i> . Open-	Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, An-	754
703	Review.net.	drew Wen, Shaochen Zhong, Hanjie Chen, and 1	755
704	Xingyu Bruce Liu, Haijun Xia, and Xiang 'Anthony'	others. 2025. Stop overthinking: A survey on ef-	756
705	Chen. 2025. Interacting with thoughtful ai . <i>ArXiv</i> ,	cient reasoning for large language models. <i>arXiv</i>	757
706	abs/2502.18676.	<i>preprint arXiv:2503.16419</i> .	758
707	Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shi-	Christoph Treude and Raula Gaikovina Kula. 2025.	759
708	wei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao,	Interacting with ai reasoning models: Harnessing	760
709	and Dacheng Tao. 2025. O1-pruner: Length-	"thoughts" for ai-driven software engineering . <i>ArXiv</i> ,	761
710	harmonizing fine-tuning for o1-like reasoning prun-	abs/2503.00483.	762
711	ing . <i>ArXiv</i> , abs/2501.12570.	Harsh Trivedi, Niranjana Balasubramanian, Tushar Khot,	763
712	Yu Meng, Mengzhou Xia, and Danqi Chen. 2024.	and Ashish Sabharwal. 2022. Musique: Multi-	764
713	Simpo: Simple preference optimization with a	hop questions via single-hop question composition.	765
714	reference-free reward . <i>ArXiv</i> , abs/2405.14734.	<i>Transactions of the Association for Computational</i>	766
715	OpenAI. 2025. Introducing openai o3 and o4-mini .	<i>Linguistics</i> , 10:539–554.	767
716	Technical report, OpenAI.	Jonathan Uesato, Nate Kushman, Ramana Kumar,	768
717	Rafael Rafailov, Yaswanth Chittooru, Ryan Park, Harshit	H. Francis Song, Noah Y. Siegel, Lisa Wang, An-	769
718	Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn,	tonia Creswell, Geoffrey Irving, and Irina Higgins.	770
719	and Scott Niekum. 2024. Scaling laws for reward	2022. Solving math word problems with process- and	771
		outcome-based feedback . <i>CoRR</i> , abs/2211.14275.	772

773 Junlin Wang, Shang Zhu, Jon Saad-Falcon, Ben Athi-
774 waratkun, Qingyang Wu, Jue Wang, Shuaiwen Leon
775 Song, Ce Zhang, Bhuwan Dhingra, and James Zou.
776 2025. Think deep, think fast: Investigating effi-
777 ciency of verifier-free inference-time-scaling meth-
778 ods. *arXiv preprint arXiv:2504.14047*.

779 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai
780 Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui.
781 2024. [Math-shepherd: Verify and reinforce llms step-](#)
782 [by-step without human annotations](#). In *Proceedings*
783 *of the 62nd Annual Meeting of the Association for*
784 *Computational Linguistics (Volume 1: Long Papers),*
785 *ACL 2024, Bangkok, Thailand, August 11-16, 2024,*
786 *pages 9426–9439*. Association for Computational
787 Linguistics.

788 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten
789 Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le,
790 and Denny Zhou. 2022. Chain-of-thought prompt-
791 ing elicits reasoning in large language models. In
792 *Advances in Neural Information Processing Systems,*
793 *volume 35,* pages 24824–24837. Curran Associates,
794 Inc.

795 Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu,
796 Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih
797 Ghazi, and Ravi Kumar. 2024. [On memoriza-](#)
798 [tion of large language models in logical reasoning](#).
799 <https://arxiv.org/abs/2410.23123>.

800 Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo,
801 Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhi-
802 rong Wu, and Chong Luo. 2025. Logic-rl: Un-
803 leashing llm reasoning with rule-based reinforcement
804 learning. *arXiv preprint arXiv:2502.14768*.

805 Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng
806 He. 2025a. [Chain of draft: Thinking faster by writing](#)
807 [less](#). *CoRR*, abs/2502.18600.

808 Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Jun-
809 nan Li, and Caiming Xiong. 2025b. Scalable chain
810 of thoughts via elastic reasoning. *arXiv preprint*
811 *arXiv:2505.05315*.

812 Danlong Yuan, Tian Xie, Shaohan Huang, Zhuocheng
813 Gong, Huishuai Zhang, Chong Luo, Furu Wei, and
814 Dongyan Zhao. 2025. [Efficient rl training for reason-](#)
815 [ing models via length-aware optimization](#). *Preprint,*
816 *arXiv:2505.12284*.

817 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen
818 Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jin-
819 gren Zhou, and Junyang Lin. 2025. The lessons of
820 developing process reward models in mathematical
821 reasoning. *arXiv preprint arXiv:2501.07301*.

822 Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei,
823 Nathan Scales, Xuezhi Wang, D. Schuurmans,
824 O. Bousquet, Quoc Le, and Ed H. Chi. 2022. [Least-](#)
825 [to-most prompting enables complex reasoning in](#)
826 [large language models](#). *ArXiv*, abs/2205.10625.

Appendix Contents

827		
828	A	Comparison with Process Reward Models 13
829	B	Additional Training Details..... 13
830	B.1	Training Setup..... 13
831	B.2	Training Stability..... 13
832	B.3	Think-answer Template 14
833	C	Additional Reward Details 14
834	C.1	Reward Definition 14
835	C.2	Conditional Intermediate Reward 14
836	D	Beyond Intermediate Ground Truth 15
837	E	Dataset Details..... 17
838	E.1	In-Domain..... 17
839	E.2	Out-of-Domain 17
840	F	Evaluation Metrics..... 17
841	F.1	Main Metrics..... 17
842	F.2	Substring Exact Match (SubEM) and Reward Hacking 18
843	G	Analysis of Intermediate Answers 18
844	G.1	Qualitative Analysis on Interleaved Reasoning Trace..... 18
845	G.2	Information Sufficiency of Intermediate Answers..... 19
846	H	Backtracking in Interleaved Reasoning 20
847	I	Case Studies of Interleaved Reasoning..... 20
848	J	Use of Large Language Models 26

A Comparison with Process Reward Models

Our reward approach differs from Process Reward Models (PRMs) in several key aspects. While PRMs typically provide token-level feedback during generation, our method evaluates the entire trajectory after completion and assigns rewards based on identifiable intermediate answers. This design choice helps avoid common PRM challenges such as reward hacking and complex training pipelines while still providing meaningful feedback on intermediate reasoning steps. Our results suggest that a simple rule-based reward can achieve similar benefits to more complex PRM implementations, in terms of guiding the model towards correct solutions.

B Additional Training Details

B.1 Training Setup.

All experiments were conducted using VERL (Sheng et al., 2024), an efficient reinforcement learning framework for language models. We performed all experiments on 8 NVIDIA H100 GPUs with 80GB memory. We also used a consistent set of hyperparameters to ensure fair comparison between methods. We evaluate and save every 100 steps during training, and continue training from the last saved checkpoint if the training is interrupted (e.g., OOM). The core parameters are listed in Table 7.

Parameter	Value
Actor learning rate	1×10^{-6}
Critic learning rate	1×10^{-6}
Train batch size	16
Validation batch size	2048
PPO mini batch size	32
PPO micro batch size	16
Critic micro batch size	8
KL coefficient	0.001
KL loss type	low variance KL
Max prompt length	3096 tokens
Max response length	2548 tokens
Sampling temperature	0.8
Number of samples per prompt	8
Stable training threshold (ϵ)	0.05
Critic warmup steps	0
Evaluation frequency	200 steps
Tensor model parallel size	2

Table 7: Core training hyperparameters.

B.2 Training Stability.

To validate the stability of our reward logic and training approach, we conduct experiments using three different random seeds (0, 1, and 42) on Qwen2.5-1.5B and report the results across all datasets. Appendix B.2 presents the stability analysis results. The table shows Pass@1 accuracy and TTFT metrics for each seed across all five datasets, along with the variance (Δ) calculated as the difference between maximum and minimum values across seeds. The low variance across seeds (Δ row) confirms the stability of our training approach.

Seed	K&K		Musique		GPQA		MMLU		MATH		Average	
	Pass@1	TTFT	Pass@1	TTFT	Pass@1	TTFT	Pass@1	TTFT	Pass@1	TTFT	Pass@1	TTFT
0	0.537	0.155	0.715	0.175	0.499	0.162	0.440	0.221	0.333	0.177	0.505	0.178
1	0.510	0.140	0.675	0.148	0.478	0.185	0.445	0.203	0.283	0.149	0.478	0.165
42	0.533	0.132	0.710	0.155	0.489	0.192	0.460	0.211	0.313	0.157	0.501	0.169
Δ	0.027	0.023	0.040	0.027	0.021	0.030	0.020	0.018	0.050	0.028	0.027	0.013

Table 8: Training stability analysis across three random seeds (0, 1, 42) for the interleaved reasoning method with intermediate rewards. Δ represents the variance (max - min) across seeds, demonstrating consistent performance.

B.3 Think-answer Template

We use the original DeepSeek-R1 training template for think-answer baseline, as shown in Table 9.

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within `<think>` `</think>` and `<answer>` `</answer>` tags, respectively, i.e., `<think>` reasoning process here `</think>` `<answer>` answer here `</answer>`. User: `prompt`. Assistant:

Table 9: Think-answer template. `prompt` will be replaced with the specific reasoning question during training.

C Additional Reward Details

C.1 Reward Definition

Given the generated sequence y and the ground truth answer $g = \{g_1, g_2, \dots, g_N\}$, which contains all intermediate and the final answer, we perform the reward calculation based on three main components:

1. **Format Reward:** This basic component evaluates the structural aspects of the generated response. It checks whether the model properly alternates between thinking and answering phases using the designated tags (`<think>``</think>` and `<answer>``</answer>`). The reward is calculated as:

$$r_{\text{format}}(y) = \lambda_f \cdot \begin{cases} 1.0 & \text{if format is correct} \\ -1.0 & \text{if format is incorrect} \end{cases} \quad (6)$$

where “correct” format means all tags are properly opened and closed, with proper alternation between thinking and answering. This reward is applied to both think-answer and interleaved reasoning.

2. **Final Accuracy Reward:** This component evaluates whether the final answer provided by the model matches the ground truth. We apply this reward *only when the format is correct* and use exact match for evaluation:

$$r_{\text{final}}(x, y) = \lambda_a \cdot \begin{cases} 2.0 & \text{if } y_{\text{answer}}^{(N)} = g_N \\ -1.5 & \text{if } y_{\text{answer}}^{(N)} \neq g_N \\ -2.0 & \text{if answer is not parseable} \end{cases} \quad (7)$$

where g_N is the final ground truth answer. For structured outputs (like numerical answers or multi-choice questions), we normalize both the model’s answer and ground truth and use exact match for evaluation. This reward is applied to both think-answer and interleaved reasoning.

3. **Intermediate Accuracy Reward:** This component provides rewards for correct intermediate answers, calculated using one of the three strategies discussed in Section 3.2.2. The intermediate reward is applied conditionally, as detailed in Algorithm 1, and is only used for interleaved reasoning.

C.2 Conditional Intermediate Reward

We provide detailed descriptions on three intermediate reward strategies in this section. The base intermediate reward value R_{base} is set to be 0.5 in this work.

1. **All-or-None:** This strategy requires all intermediate answers to be correct in sequence to receive any reward. The reward calculation is:

$$r_{\text{intermediate}}^{\text{all-or-none}}(x, y) = \begin{cases} R_{\text{base}} & \text{if } \mathcal{C}(y_{\text{answer}}^{(k)}), \forall k \in [1, N - 1], \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

This strategy is the most demanding but ensures the model maintains a consistent reasoning path throughout.

2. **Partial Credit:** This strategy rewards each correct intermediate answer independently, providing partial credit regardless of other steps:

$$r_{\text{intermediate}}^{\text{partial}}(x, y) = \frac{R_{\text{base}}}{N-1} \sum_{k=1}^{N-1} \mathcal{C}(y_{\text{answer}}^{(k)}) \quad (9)$$

This approach is more forgiving, allowing the model to recover from early mistakes while still incentivizing correct intermediate steps.

3. **Time-Discounted:** This strategy awards the full base reward R_{base} when every intermediate answer is correct. If any intermediate answers are missing or wrong, the reward is shared among the correct ones with higher weight on earlier correct answers. Formally,

$$r_{\text{intermediate}}^{\text{time-disc}}(x, y) = \begin{cases} R_{\text{base}}, & \text{if } S_{\text{correct}} = g, \\ R_{\text{base}} \frac{1}{N-1} \sum_{g_j \in S_{\text{correct}}} \frac{1}{k_j}, & \text{otherwise,} \end{cases} \quad (10)$$

where $S_{\text{correct}} \subseteq g$ is the set of ground-truth intermediate answers that the model outputs at least once, k_j is the index of the first step in which the model’s answer matches g_j , and $N-1$ is the total number of ground-truth intermediate answers. The harmonic weight $1/k_j$ gives greater credit to earlier correct answers while still granting some credit to later ones. Note that the time-discounted partial reward calculation will *not* be used if all intermediate answers are correct. Therefore the model receives a larger reward when all intermediate answers are correct, and the reward quickly drops even if one intermediate answer is incorrect. This design choice was intentionally made to strongly incentivize the model to generate all correct intermediate steps, rather than being satisfied with partial correctness.

Our analysis reveals that early correct token generation actually enhances reasoning (§4.2). Our reward strategy analysis (Table 6) shows that the time-aware reward is more effective than the time-agnostic reward, suggesting that generating correct intermediate answers early helps the model reason more efficiently. Consequently, *behaviors such as backtracking or rethinking are less frequently observed*, as the reward explicitly encourages the model to generate early intermediate answers. While our analysis demonstrates that early correct token generation is beneficial, investigating the depth and breadth of interleaved reasoning represents an interesting direction for future research.

D Beyond Intermediate Ground Truth

Intermediate ground truths help during training, but they are not a strict requirement for deploying interleaved reasoning. *Firstly*, strong generalization mitigates training requirements. Although our training uses datasets with intermediate ground truths (K&K and Musique), the resulting models generalize to tasks that do not provide any intermediate annotations at evaluation time (Table 2). Trained only on K&K and Musique, our models achieve superior performance on MATH, GPQA, and MMLU – none of which include intermediate ground truths for evaluation. This reduces the practical limitation of needing intermediate annotations across domains.

Secondly, alternative supervision methods are readily available. The number of concurrent works using the model’s internal confidence as reward signals (Li et al., 2025a; Agarwal et al., 2025) or process rewards (Khalifa et al., 2025; Zhang et al., 2025) provides viable approaches to apply our method to datasets lacking explicit intermediate annotations. While we do not explore these combinations in this work, existing techniques for generating intermediate supervision can be integrated with our method to expand its applicability, representing an exciting future direction.

Lastly, we conduct additional ablations on GSM8K with Qwen2.5-1.5B model (Table 10), without intermediate ground truths. Even *without* intermediate rewards, interleaving preserves accuracy while substantially reducing TTFT. The results indicate that the interleaving structure itself brings substantial responsiveness gains by-default with comparable accuracy to standard think-answer training, even when intermediate rewards are absent.

Algorithm 1 Intermediate Reward Calculation

```
1: Input: Generated sequence  $y$ , ground truth intermediate answers  $g = \{g_1, g_2, \dots, g_N\}$ , current training batch  $B$ , reward strategy  $S$ 
2: Parameters: Base reward value  $R_{\text{base}}$ , stable training threshold  $\epsilon$ 
3: Output: Intermediate reward value
4: Parse  $y$  to extract all intermediate answers  $y_{\text{answer}} = \{y_{\text{answer}}^{(1)}, \dots, y_{\text{answer}}^{(N)}\}$ , where  $y_{\text{answer}}^{(N)}$  is the final answer
5:  $\text{is\_final\_correct} \leftarrow \mathcal{C}(y_{\text{answer}}^{(N)})$ 
6:  $\text{is\_format\_valid} \leftarrow \mathcal{F}(y)$ 
7:  $\text{is\_progressing} \leftarrow (\text{Acc}(B) > \text{Acc}(B - 1) - \epsilon)$ 
8: if  $\text{is\_final\_correct}$  AND  $\text{is\_format\_valid}$  AND  $\text{is\_progressing}$  then
9:    $\text{reward\_sum} \leftarrow 0$ 
10:  if  $S = \text{"All-or-None"}$  then
11:     $\text{all\_correct} \leftarrow \text{TRUE}$ 
12:    for  $k = 1$  to  $N - 1$  do
13:      if NOT  $\mathcal{C}(y_{\text{answer}}^{(k)})$  then
14:         $\text{all\_correct} \leftarrow \text{FALSE}$ 
15:        break
16:      end if
17:    end for
18:    if  $\text{all\_correct}$  then
19:       $\text{reward\_sum} \leftarrow R_{\text{base}}$ 
20:    end if
21:  else if  $S = \text{"Partial Credit"}$  then
22:    for  $k = 1$  to  $N - 1$  do
23:      if  $\mathcal{C}(y_{\text{answer}}^{(k)})$  then
24:         $\text{reward\_sum} \leftarrow \text{reward\_sum} + R_{\text{base}}/N$ 
25:      end if
26:    end for
27:  else if  $S = \text{"Time-Discounted"}$  then
28:     $\text{correct\_step} \leftarrow \{\}$  {Track all correct steps}
29:    for  $k = 1$  to  $N - 1$  do
30:      for each required answer  $g_j$  in  $g$  do
31:        if  $g_j$  not in  $\text{correct\_step}$  AND  $\mathcal{C}(y_{\text{answer}}^{(k)})$  then
32:           $\text{correct\_step}[g_j] \leftarrow k$ 
33:        end if
34:      end for
35:    end for
36:    if  $|\text{correct\_step}| = N - 1$  then
37:       $\text{reward\_sum} \leftarrow R_{\text{base}}$ 
38:    else
39:       $\text{sum\_weights} \leftarrow \sum_{\text{step} \in \text{correct\_step}} 1/\text{step}$ 
40:       $\text{reward\_sum} \leftarrow (\text{sum\_weights}/N - 1) \cdot R_{\text{base}}$ 
41:    end if
42:  end if
43:  return  $\text{reward\_sum}$ 
44: else
45:  return 0
46: end if
```

Algo.	Method	Acc.	TTFT
PPO	Normal	79.3	88.8
	Interleaved	79.7	<u>26.2</u>
GRPO	Normal	78.4	88.6
	Interleaved	78.6	<u>26.2</u>
RF++	Normal	78.1	88.5
	Interleaved	78.1	<u>25.9</u>

Table 10: Our approach provides efficiency benefits even for datasets without intermediate ground truths, with comparable accuracy gains to the think-answer structure.

E Dataset Details 946

E.1 In-Domain 947

Knights and Knaves (K&K). K&K is a logical reasoning dataset that requires multi-step reasoning to identify the correct characters (Xie et al., 2024). The dataset contains problems with varying difficulty levels based on the number of characters involved. In our experiments, we use problems with 3, 4, and 5 characters for both training and evaluation. Each difficulty level consists of 900 training examples and 100 test examples. To evaluate generalization across difficulty levels, we also test our models on problems with 6, 7, and 8 characters, which were not seen during training (Figure 3(a)). Our results indicate that interleaved reasoning is particularly effective for more challenging problems. 948
949
950
951
952
953
954

Musique. Musique is a multi-hop question answering dataset that requires retrieving and combining information from multiple sources (Trivedi et al., 2022). Problems in Musique are categorized by the number of reasoning hops needed (i.e., 2-hop, 3-hop, 4-hop). For our experiments, we use 3-hop and 4-hop questions, with 900 training examples and 100 test examples for each hop category. For efficient training and inference, we select only up to 1,000 tokens in total for the context, which includes all the supporting documents and a portion of distraction documents. Both K&K and Musique naturally contain intermediate reasoning steps and ground truth, making them ideal for training and evaluating interleaved reasoning approaches. 955
956
957
958
959
960
961
962

E.2 Out-of-Domain 963

GPQA. We use the GPQA-diamond version (Rein et al., 2023), which consists of 198 data points. GPQA is crafted by domain experts in biology, physics, and chemistry, designed to assess LLMs’ advanced reasoning and knowledge. 964
965
966

MMLU. We use MMLU-redux-2.0 (Gema et al., 2024), a cleaned and reannotated version of MMLU (Hendrycks et al., 2020). To match with GPQA, we select a subset of 198 data points from domains requiring formal reasoning: college computer science, college mathematics, abstract algebra, formal logic, college physics, and machine learning. 967
968
969
970

MATH. We also use 198 data points from the level 5 subset of MATH (Hendrycks et al., 2021), which are the most challenging problems within the dataset. These problems require complex mathematical reasoning and often involve multiple steps of computation and logical deduction. 971
972
973

F Evaluation Metrics 974

F.1 Main Metrics 975

Pass@1 Accuracy. Pass@1 accuracy measures the proportion of problems that the model solves correctly on its first attempt. We follow the evaluation methodology established in prior work (Wei et al., 2022; Guo et al., 2025; Jin et al., 2025), using Exact Match (EM) to determine correctness. For each test instance, we compare the model’s final answer against the ground truth answer after normalizing both 976
977
978
979

(removing punctuation, converting to lowercase, and standardizing numerical formats). A prediction is considered correct only if it exactly matches the normalized ground truth.

Time-to-first-token (TTFT). TTFT measures how quickly a model produces its first useful output to the user. While traditional approaches measure TTFT in absolute time (milliseconds), we normalize TTFT as the ratio of the first answer token’s position to the total response length to ensure fair comparison across different model configurations and reasoning strategies:

$$\text{TTFT} = \frac{\text{Position of first answer token}}{\text{Total response length}} \quad (11)$$

This normalized metric ranges from 0 to 1, where lower values indicate faster initial responses. This metric is particularly important for interactive applications where immediate response could vastly improve user experience.

F.2 Substring Exact Match (SubEM) and Reward Hacking

We initially experimented with SubEM as an additional evaluation metric for intermediate answers. SubEM is more lenient than EM – it measures whether the ground truth answer appears as a *substring* in the model’s response. We found that models trained with SubEM quickly learned to generate *excessively long* intermediate answers containing numerous potential responses, significantly increasing the probability of including the correct answer somewhere in the text. For example, instead of generating a concise intermediate step "The value is 42," models would produce verbose outputs like "Let me consider different possibilities: the value is 41, the value is 42, the value is 43 ..." This gaming behavior provided no pedagogical value and undermined the training.

This observation aligns with prior findings in reinforcement learning, where models exploit evaluation metrics in unintended ways (Xie et al., 2025), which is known as reward hacking. Therefore, we use EM as our main evaluation metric.

G Analysis of Intermediate Answers

In this section, we present both qualitative and quantitative analysis to show the meaningfulness of interleaved reasoning structure.

G.1 Qualitative Analysis on Interleaved Reasoning Trace

We compared two versions of the interleaved method (with and without intermediate rewards) against the standard think-answer method using an LLM-based judge (gpt-4o-mini-2024-07-18). For each problem that are solved correctly by all three methods (126 problems in total, 38 in-domain, 88 out-of-domain), we presented the problem statement and the model responses to the LLM evaluator, asking it to rate each answer on three criteria: (1) clarity and usefulness of intermediate steps, (2) timeliness and informativeness of feedback, and (3) overall user experience. The LLM was instructed to mimic a human evaluator and assign scores for each criterion and to select a winner between the two methods for each example. The evaluation prompt is shown below.

Dataset	Think-Ans vs. Interleave		Think-Ans vs. Inter+IR	
	Think-Ans Win (%)	Inter Win (%)	Think-Ans Win (%)	Interleave+IR Win (%)
In-domain	36.8	63.2	44.7	55.3
Out-of-domain	70.1	29.9	52.3	47.7
<i>Average</i>	53.4	46.7	47.8	52.2

Table 11: Our method achieves a comparable win rate with think-answer reasoning, suggesting the intermediate answers are indeed meaningful.

We calculate the win rates for each method, as shown in Table 11. Win rate is calculated as the percentage of pairwise wins (excluding ties). The results show that the base interleaved method (without

intermediate rewards) had a lower win rate compared to think-answer, indicating that not all intermediate answers were useful by default. However, when intermediate rewards were used to encourage the model to produce more meaningful intermediate answers, the interleaved method outperformed think-answer. The following prompt was used to instruct the LLM judge for qualitative evaluation:

1016
1017
1018
1019

Evaluation Prompt

You are an expert evaluator of large language model reasoning. You are given a multi-hop problem and two model-generated answers. The first answer uses interleaved reasoning: it alternates between thinking and answering, providing intermediate answers as soon as they are derived. The second answer uses the traditional think-answer reasoning: it completes all reasoning before providing the final answer. For each answer, your task is to rate it on a scale from 1 (very poor) to 10 (excellent) for each of the following criteria:

- Clarity and usefulness of intermediate reasoning steps
- Timeliness and informativeness of feedback (does the response help the user understand the reasoning?)
- Overall user experience

Instructions:

- Assign a score (1-10) for each criterion for both answers.
- After scoring, briefly explain your reasoning for the scores.
- Respond in JSON as:

```
{  
  "interleave": {  
    "clarity_usefulness": <int>,  
    "timeliness_informativeness": <int>,  
    "overall_experience": <int>  
  },  
  "think_answer": {  
    "clarity_usefulness": <int>,  
    "timeliness_informativeness": <int>,  
    "overall_experience": <int>  
  },  
  "explanation": "<your reasoning for these scores>"  
}
```

Problem:

<problem_text>

Interleaved Reasoning Answer:

<interleave_response>

Think-Answer Reasoning Answer:

<think_answer_response>

1020

G.2 Information Sufficiency of Intermediate Answers

1021

To quantify the utility of intermediate answers, we measure whether the intermediate answers alone (excluding <think> traces) contain sufficient information to predict the final answer. We prompt an external LLM (gpt-4o-mini-2024-07-18) to derive the final answer based on two types of condensed

1022

1023

1024

1025 contexts. The first context, *Interleaved Answers*, consists of the sequence of sub-answers extracted from
 1026 the <answer> tags. The second context, *CoT Snippets*, comprises N sentences randomly sampled from
 1027 the standard think-answer reasoning, where N equals the number of intermediate answers generated by
 1028 our method for that problem. The evaluation prompt is shown below. We present the results in Table 12.
 1029 We report the percentage of times the external LLM could correctly answer the final answer given the
 1030 condensed context. The results show that Interleaved Answers outperform the CoT snippet by 20.2%
 1031 on average. This gap confirms that our method’s intermediate answers are not merely arbitrary text but
 1032 meaningful checkpoints that effectively capture the state of the solution.

Dataset	CoT Snippets	Interleaved Answers	Δ
In-domain	76.3	94.7	+18.4
Out-of-domain	71.2	93.2	+22.0
<i>Average</i>	73.8	94.0	+20.2

Table 12: Performance of an external LLM (GPT-4o) in predicting the final answer given only condensed contexts.

Information Sufficiency Evaluation Prompt

I will provide you with a Problem and a series of Reasoning Checkpoints derived from a model’s solution attempt. The full reasoning trace has been hidden; only the partial context remains. Your task is to read the checkpoints and infer the Final Answer to the problem based on the information provided in these checkpoints.

Problem: <problem_text>

Reasoning Checkpoints: <extracted_content>

Final Answer:

H Backtracking in Interleaved Reasoning

1033 The interleaved reasoning structure means that intermediate answers are committal, which restricts
 1034 backtracking seen in regular CoT reasoning. However, while backtracking might potentially improve
 1035 reasoning, it could also lead to overthinking and inefficiency (Li et al., 2025c). Our results suggest
 1036 that interleaved reasoning encourages more efficient reasoning by incentivizing models to reach correct
 1037 intermediate conclusions early (§4.2), reducing the need for backtracking. Our results also show that
 1038 the structural benefits of interleaved reasoning outweigh potential committal disadvantages, achieving
 1039 higher accuracy, lower TTFT (Table 2), and shorter reasoning traces (Table 5). We leave the exploration
 1040 of backtracking in interleaved reasoning for future work.
 1041
 1042

I Case Studies of Interleaved Reasoning

1043 We present case studies to gain a deeper understanding of interleaved reasoning. We focus on Qwen2.5-
 1044 1.5B-Instruct that was trained using PPO and the Time-Discounted reward strategy. We present the
 1045 examples from both in-domain (K&K, Musique) and out-of-domain (MATH, MMLU, GPQA) datasets.
 1046 The model successfully generalizes to unseen, complex problems, demonstrating its ability to break down
 1047 problems into logical steps, provide intermediate conclusions, and arrive at final answers. **Green** for
 1048 correct answers and **red** for incorrect answers.
 1049

MATH

The model successfully decomposes the problem into smaller steps and correctly computes the final answer.

Question:

Expand the product $(x - 2)^2(x + 2)^2$. What is the product of the nonzero coefficients of the

resulting expression, including the constant term?

Ground Truth: -128

Interleaved Reasoning:

<think>First, let's expand the product step by step. We start with the expression $(x - 2)^2(x + 2)^2$. We can simplify this by recognizing that $(x - 2)(x + 2) = x^2 - 4$. So, the expression becomes $(x^2 - 4)^2$.</think>

<answer> $(x^2 - 4)^2$ </answer>

<think>Next, we need to expand $(x^2 - 4)^2$. This is a square of a binomial, which can be expanded using the formula $(a - b)^2 = a^2 - 2ab + b^2$.</think>

<answer> $(x^2 - 4)^2 = x^4 - 8x^2 + 16$ </answer>

<think>The expanded form of the expression is $x^4 - 8x^2 + 16$. The nonzero coefficients are 1, -8, and 16. We need to find the product of these coefficients.</think>

<answer>The product of the coefficients 1, -8, and 16 is $1 \times -8 \times 16 = -128$.</answer>

<think>The final answer is:</think>

<answer>-128</answer>

1051

MATH

A failed case: the model follows the interleaved structure but makes a calculation error in step 4, resulting in an incorrect final answer. However, with interleaved reasoning, the incorrect intermediate answer can potentially be spotted early by users.

Question:

Let $f(x)$ be the polynomial

$$f(x) = 3x^4 + 5x^2 - 9x - 2.$$

If $g(x)$ is equal to the polynomial $f(x - 1)$, what is the sum of the coefficients of g ?

Ground Truth: -2

Interleaved Reasoning:

<think> First, we need to find the polynomial $g(x)$ by substituting $x - 1$ into $f(x)$. </think>

<answer> $f(x) = 3x^4 + 5x^2 - 9x - 2$ </answer>

<think> Second, we substitute $x - 1$ into $f(x)$ to get $g(x)$. </think>

<answer> $g(x) = 3(x - 1)^4 + 5(x - 1)^2 - 9(x - 1) - 2$ </answer>

<think> Third, we expand $g(x)$ to find the polynomial in standard form. </think>

<answer> $g(x) = 3(x^4 - 4x^3 + 6x^2 - 4x + 1) + 5(x^2 - 2x + 1) - 9(x - 1) - 2$ </answer>

<think> Fourth, we simplify $g(x)$ to find the coefficients. </think>

<answer> $g(x) = 3x^4 - 12x^3 + 24x^2 - 24x + 3 + 5x^2 - 10x + 5 - 9x + 9 - 2$ </answer>

<think> Fifth, we combine like terms to get the polynomial in standard form. </think>

<answer> $g(x) = 3x^4 - 12x^3 + 29x^2 - 43x + 15$ </answer>

<think> Sixth, we find the sum of the coefficients of $g(x)$. </think>

<answer> $3 - 12 + 29 - 43 + 15 = -4$ </answer>

<think> Seventh, the final answer is:</think>

<answer> -4 </answer>

1052

Knights and Knaves

The model successfully breaks down a multi-hop logic puzzle, providing correct intermediate conclusions at each step, and arrives at the correct final answer.

Question:

A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 5 inhabitants: Victoria, Mia, Sebastian, Ethan, and Olivia.

"Mia is a knave and Ethan is a knave" - Victoria.

Mia said that If Mia is a knight then Sebastian is a knave.

1053

Sebastian expressed that Victoria is a knave if and only if Ethan is a knave.

Ethan remarked, "Sebastian is a knight or Victoria is a knave".

"Sebastian is a knight and Sebastian is a knave" - Olivia.

So who is a knight and who is a knave?

Ground Truth: (1) Victoria is a knave (2) Mia is a knave (3) Sebastian is a knight (4) Ethan is a knight (5) Olivia is a knave

Interleaved Reasoning:

<think> First, consider Victoria's statement: 'Mia is a knave and Ethan is a knave'. This must be a lie because an implication can only be true if both conditions are false. </think>

<answer> Victoria is a knave </answer>

<think> Second, consider Sebastian's statement: 'Victoria is a knave if and only if Ethan is a knave'. Since Victoria is a knave (from our previous deductions) and Sebastian is a knight (since he always lies), Sebastian must be a knight. </think>

<answer> Sebastian is a knight </answer>

<think> Third, consider Mia's statement: 'If Mia is a knight then Sebastian is a knave'. Since Mia is a knight (since she always lies) and Sebastian is a knight (since he always lies), Mia must be a knave. </think>

<answer> Mia is a knave </answer>

<think> Fourth, consider Ethan's statement: 'Sebastian is a knight or Victoria is a knave'. Since Sebastian is a knight (since he always lies) and Victoria is a knave (since she always lies), Ethan must be a knight. </think>

<answer> Ethan is a knight </answer>

<think> Fifth, consider Olivia's statement: 'Sebastian is a knight and Sebastian is a knave'. This must be a lie because an implication can only be true if both conditions are simultaneously true, which is impossible. </think>

<answer> Olivia is a knave </answer>

<think> So, the final answer is:</think>

<answer>(1) Victoria is a knave (2) Mia is a knave (3) Sebastian is a knight (4) Ethan is a knight (5) Olivia is a knave</answer>

Musique

The model sequentially solves each sub-question, yielding the correct final answer. Occasionally, the model provides intermediate answers that are duplicates of the final answer, which is not ideal.

Question:

Where does Plymouth rank in population among the cities of the country having a gallery displaying the painting named The Woman Taken in the main subject of milkman joke?

Ground Truth: 30th

Interleaved Reasoning:

<think> First, identify the country where Plymouth is located. </think>

<answer> United Kingdom </answer>

<think> Second, identify the painting named The Woman Taken in Adultery. </think>

<answer> The Woman Taken in Adultery </answer>

<think> Third, identify the gallery displaying this painting. </think>

<answer> National Gallery </answer>

<think> Fourth, determine Plymouth's ranking in population among the cities of the United Kingdom. </think>

<answer> 30th </answer>

<think> Fifth, state the final answer. </think>

<answer> 30th </answer>

The model translates logic formulas stepwise, mapping each part to natural language and selecting the correct answer.

Question:

Use the following key to translate the given formula of PL to natural, English sentences.

A: Marina reads a Percy Jackson book.

B: Izzy plays Minecraft.

C: Emily stops working.

D: Russell makes dinner.

E: Ashleigh stops by.

$\sim (A \supset B) \cdot (B \supset \sim E)$

A) It's not the case that Marina's reading a Percy Jackson book entails that Izzy plays Minecraft, but Izzy's playing Minecraft does entail that Ashleigh doesn't stop by.

B) If Marina doesn't read a Percy Jackson book, then Izzy plays Minecraft, which entails that Ashleigh doesn't stop by.

C) Marina's reading a Percy Jackson book does not entail that Izzy plays Minecraft, but Izzy plays Minecraft provided that Ashleigh doesn't stop by.

D) It's not true that Marina reads a Percy Jackson book only when Izzy plays Minecraft, but Izzy plays Minecraft only when Ashleigh stops by.

Ground Truth: A

Interleaved Reasoning:

<think>First, identify the logical structure of the given formula. The formula is a conjunction of two main parts: $\sim (A \supset B)$ and $(B \supset \sim E)$. The symbol \supset represents material implication ("entails", "if...then...", "only when"), \sim represents negation ("not", "it's not the case that"), and \cdot represents conjunction ("and", "but").</think>

<answer>The formula is $\sim (A \supset B) \cdot (B \supset \sim E)$.</answer>

<think>Second, translate the first part of the conjunction: $\sim (A \supset B)$. $A \supset B$ translates to "Marina reads a Percy Jackson book entails Izzy plays Minecraft" or "If Marina reads a Percy Jackson book, then Izzy plays Minecraft." The negation $\sim (A \supset B)$ means "It is not the case that (Marina reads a Percy Jackson book entails Izzy plays Minecraft)."</think>

<answer> $\sim (A \supset B)$ translates to "It's not the case that Marina's reading a Percy Jackson book entails that Izzy plays Minecraft."</answer>

<think>Third, translate the second part of the conjunction: $(B \supset \sim E)$. B translates to "Izzy plays Minecraft." $\sim E$ translates to "Ashleigh doesn't stop by." So, $B \supset \sim E$ translates to "Izzy plays Minecraft entails that Ashleigh doesn't stop by" or "If Izzy plays Minecraft, then Ashleigh doesn't stop by."</think>

<answer> $(B \supset \sim E)$ translates to "Izzy's playing Minecraft does entail that Ashleigh doesn't stop by."</answer>

<think>Fourth, combine the translations of the two parts with a conjunction (e.g., "but" or "and"). The overall formula $\sim (A \supset B) \cdot (B \supset \sim E)$ combines these two statements.</think>

<answer>Combining the parts: "It's not the case that Marina's reading a Percy Jackson book entails that Izzy plays Minecraft, but Izzy's playing Minecraft does entail that Ashleigh doesn't stop by."</answer>

<think>Fifth, compare this translation with the given options to select the correct one. Option A matches our translation. Option B incorrectly translates $\sim (A \supset B)$ as "If Marina doesn't read a Percy Jackson book, then Izzy plays Minecraft". Option C incorrectly translates $(B \supset \sim E)$ as "Izzy plays Minecraft provided that Ashleigh doesn't stop by", which would be $(\sim E \supset B)$. Option D incorrectly translates $(B \supset \sim E)$ as "Izzy plays Minecraft only when Ashleigh stops by", which would be $(B \supset E)$. Therefore, option A is the correct translation.</think>

<answer>The final answer is: A</answer>

MMLU

The model successfully applies interleaved reasoning to derive and combine gradients step by step, leading to the correct answer.

Question:

Suppose we have the following objective function: $\arg \min_w \frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} \gamma \|w\|_2^2$. What is the gradient of $\frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} \lambda \|w\|_2^2$ with respect to w ?

- A) $\nabla_w f(w) = (X^T X + \lambda I)w - X^T y + \lambda w$
- B) $\nabla_w f(w) = X^T Xw - X^T y + \lambda$
- C) $\nabla_w f(w) = X^T Xw - X^T y + \lambda w$
- D) $\nabla_w f(w) = X^T Xw - X^T y + (\lambda + 1)w$

Ground Truth: A

Interleaved Reasoning:

<think>First, let's identify the objective function and the terms involved. The objective function is given by:</think>

<answer>the objective function is $\frac{1}{2} \|Xw - y\|_2^2 + \frac{1}{2} \gamma \|w\|_2^2$ </answer>

<think>Now, let's break down the terms:</think>

<answer>the terms are $\frac{1}{2} \|Xw - y\|_2^2$ and $\frac{1}{2} \gamma \|w\|_2^2$ </answer>

<think>Next, we need to find the gradient of the first term $\frac{1}{2} \|Xw - y\|_2^2$ with respect to w . Recall that the gradient of $\|z\|_2^2$ with respect to z is $2z$. So, the gradient of $\frac{1}{2} \|Xw - y\|_2^2$ is:</think>

<answer>the gradient of $\frac{1}{2} \|Xw - y\|_2^2$ is $X^T(Xw - y)$ </answer>

<think>Now, let's find the gradient of the second term $\frac{1}{2} \gamma \|w\|_2^2$ with respect to w . The gradient of $\|w\|_2^2$ with respect to w is $2w$. So, the gradient of $\frac{1}{2} \gamma \|w\|_2^2$ is:</think>

<answer>the gradient of $\frac{1}{2} \gamma \|w\|_2^2$ is γw </answer>

<think>Now, we can combine these gradients to find the gradient of the entire objective function:</think>

<answer>the gradient of the entire objective function is $X^T(Xw - y) + \gamma w$ </answer>

<think>Now, let's compare this with the given options:</think>

<answer>The correct option is A) $\nabla_w f(w) = (X^T X + \lambda I)w - X^T y + \lambda w$ </answer>

GPQA

The model answers the problem step by step, ruling out distractors and resulting in the correct final answer.

Question:

In the lab, a chemist discovers an unnamed bottle with a white solid inside. He gathers the following spectral information in order to identify the liquid. Make a reasonable structural suggestion for the unidentified drug using the information provided.

Mass: molecular ion peak at $m/z = 156$ (100%)

a peak at $m/z = 158$ (32%)

IR : broad peak from $3500-2700 \text{ cm}^{-1}$, strong sharp peak at 1720 cm^{-1}

^1H NMR: 11.0 ppm (s, 1H), 8.02 ppm (d, 2H), 7.72 (d, 2H)

- A) Phenyl chloroformate
- B) 2-chlorobenzoic acid
- C) 3-Chloro-2-hydroxybenzaldehyde
- D) 4-chlorobenzoic acid

Ground Truth: D

Interleaved Reasoning:

<think>First, analyze the Mass Spectrum data. A molecular ion peak (M^+) at $m/z = 156$ and an $M+2$ peak at $m/z = 158$ with an intensity ratio of approximately 100:32 (or 3:1) is characteristic of a compound containing one chlorine atom ($^{35}\text{Cl}/^{37}\text{Cl}$ isotopic abundance). The molecular weight

of $C_7H_5ClO_2$ is approximately 156.5 g/mol (using ^{35}Cl), which matches the M^+ peak. All given options have this molecular formula.</think>

<answer>The mass spectrum indicates a molecular weight of approximately 156 and the presence of one chlorine atom. All options (Phenyl chloroformate, 2-chlorobenzoic acid, 3-Chloro-2-hydroxybenzaldehyde, 4-chlorobenzoic acid) have the molecular formula $C_7H_5ClO_2$.</answer>

<think>Second, analyze the IR Spectrum. A very broad peak spanning $3500-2700\text{ cm}^{-1}$ is highly characteristic of the O-H stretch of a carboxylic acid, which is broadened due to hydrogen bonding. The strong, sharp peak at 1720 cm^{-1} is characteristic of a C=O (carbonyl) stretch, consistent with a carboxylic acid (typically $1700-1725\text{ cm}^{-1}$). This strongly suggests the presence of a -COOH group.</think>

<answer>The IR spectrum strongly suggests the presence of a carboxylic acid group (-COOH) due to the broad O-H stretch ($3500-2700\text{ cm}^{-1}$) and the C=O stretch (1720 cm^{-1}).</answer>

<think>Third, analyze the 1H NMR Spectrum. The signal at 11.0 ppm (singlet, 1H) is highly deshielded and is characteristic of a carboxylic acid proton (-COOH). The signals at 8.02 ppm (doublet, 2H) and 7.72 ppm (doublet, 2H) are in the aromatic region. A pattern of two doublets, each integrating to 2 protons, is indicative of a para-substituted benzene ring. The two protons of one type are chemically equivalent, and the two protons of the other type are also chemically equivalent, and they are ortho to each other respectively.</think>

<answer>The 1H NMR spectrum shows a carboxylic acid proton at 11.0 ppm and a para-substituted benzene ring pattern (two doublets, each 2H, at 8.02 ppm and 7.72 ppm).</answer>

<think>Fourth, evaluate the options based on the combined spectral data: A) Phenyl chloroformate: Contains an ester-like C=O and an acid chloride C-Cl. IR for C=O in R-O-CO-Cl is typically higher (around $1760-1780\text{ cm}^{-1}$). It lacks the carboxylic acid O-H and the acidic proton at 11.0 ppm. B) 2-chlorobenzoic acid: Is a carboxylic acid, consistent with IR and the 11.0 ppm NMR signal. However, an ortho-substituted benzene ring (1,2-disubstituted) would typically give a more complex pattern for the 4 aromatic protons (e.g., 4 distinct signals or complex multiplets), not two doublets of 2H each. C) 3-Chloro-2-hydroxybenzaldehyde: Contains an aldehyde and a phenolic -OH. An aldehyde C=O IR is plausible ($1720-1740\text{ cm}^{-1}$), but the broad O-H from $3500-2700\text{ cm}^{-1}$ points to a carboxylic acid, not typically a phenol (phenols are usually sharper, $3200-3600\text{ cm}^{-1}$). The aldehyde proton (-CHO) signal is typically around 9-10 ppm. The aromatic region for a 1,2,3-trisubstituted benzene would show 3 distinct aromatic protons. D) 4-chlorobenzoic acid: Is a carboxylic acid. This fits the IR data (broad O-H, C=O at 1720 cm^{-1}) and the NMR signal at 11.0 ppm (-COOH). It is a para-substituted benzene ring (chlorine at position 4, carboxylic acid at position 1). This substitution pattern gives rise to two sets of equivalent aromatic protons (H-2/H-6 and H-3/H-5), which would appear as two doublets, each integrating to 2H. This perfectly matches the NMR signals at 8.02 ppm and 7.72 ppm.</think>

<answer>Comparing the spectral interpretations with the options: A) Phenyl chloroformate is inconsistent with IR (no broad O-H) and NMR (no signal at 11 ppm). B) 2-chlorobenzoic acid is inconsistent with the aromatic NMR pattern (would not be two doublets of 2H each). C) 3-Chloro-2-hydroxybenzaldehyde is inconsistent with IR (carboxylic acid O-H unlikely) and NMR (aldehyde H at 9-10 ppm, different aromatic pattern). D) 4-chlorobenzoic acid is fully consistent with all spectral data: MS indicates $C_7H_5ClO_2$, IR indicates a carboxylic acid, and 1H NMR indicates a carboxylic acid proton and a para-substituted benzene ring.</answer>

<think>The final answer is therefore 4-chlorobenzoic acid.</think>

<answer>D) 4-chlorobenzoic acid</answer>

A failed case: the model occasionally fails to follow the interleaved structure. The model also sometimes generates shorter thinking and longer answers than necessary.

Question:

Reaction of solid A with 8 equivalents of gas B forms bright red product C.

When C reacts with 2 equivalents of gas D, it produces the extremely hazardous product E.

C reacts with water to reform A plus two different acids F and G. F is a strong acid while G is a weak acid.

D reacts with B in a 1:1 ratio to form H, which is used as a solvent. What is the molecular symmetry group of E?

- A) C_{2v}
- B) D_{4h}
- C) $D_{\infty h}$
- D) C_2

Ground Truth: A

Interleaved Reasoning:

<think>First, let's identify the products and their properties:</think>

<answer>The products are C, E, F, and G. C is a bright red product, E is an extremely hazardous product, F is a strong acid, and G is a weak acid. D reacts with B in a 1:1 ratio to form H, which is used as a solvent.</answer>

<think>Next, let's analyze the reaction of C with water to reform A plus two different acids F and G:</think>

<answer>The reaction of C with water to reform A plus two different acids F and G is a typical acid-base reaction. Since C is a bright red product, it is likely a metal oxide or a metal hydroxide. When it reacts with water, it will form a metal hydroxide and a metal ion, which will then react with the acid to form the two different acids F and G.</answer>

<think>Now, let's consider the reaction of E with D to produce H:</think>

<answer>The reaction of E with D to produce H is a typical acid-base reaction. Since E is an extremely hazardous product, it is likely a strong acid. When it reacts with D, it will form H, which is used as a solvent.</answer>

<think>Finally, let's determine the molecular symmetry group of E:</think>

<answer>The molecular symmetry group of E is $D_{\infty h}$. This is because E is an extremely hazardous product, which suggests that it has a high degree of symmetry. The $D_{\infty h}$ group has 16 symmetry operations, including 8 mirror planes and 8 axes of inversion. This group is also known as the point group D_{4h} , but the presence of an infinite number of axes of inversion makes $D_{\infty h}$ a more appropriate choice.</answer>

<answer>C) $D_{\infty h}$ </answer>

J Use of Large Language Models

We used LLMs solely for proofreading and polishing text. All ideas, results, and arguments are original.