Interleaved Reasoning for Large Language Models via Reinforcement Learning

Roy Xie[†‡] **David Qiu**[†] **Deepak Gopinath**[†] **Dong Lin**[†] **Yanchao Sun**[†]

Chong Wang[†] **Saloni Potdar**[†] **Bhuwan Dhingra**[†‡]

[†]Apple [‡]Duke University

Abstract

Long chain-of-thought (CoT) significantly enhances large language models' (LLM) reasoning capabilities. However, the extensive reasoning traces lead to inefficiencies and an increased time-to-first-token (TTFT). We propose a novel training paradigm that uses 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 rule-based reward to incentivize correct intermediate steps, which guides the policy model toward correct reasoning paths by leveraging intermediate signals generated during interleaved reasoning. Extensive experiments conducted across five diverse datasets and three RL algorithms (PPO, GRPO, and REINFORCE++) demonstrate consistent improvements over traditional think-answer reasoning, without requiring external tools. Specifically, our approach reduces TTFT by over 80% on average and improves up to 19.3% in Pass@1 accuracy. Furthermore, our method, trained solely on question answering and logical reasoning datasets, exhibits strong generalization ability 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.

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.

1 Introduction

Reasoning large language models (LLMs) [18, 13] have demonstrated advanced capabilities in complex multi-hop tasks through long chain-of-thoughts (CoT) [50]. 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 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 [6, 42] and underthinking [49].

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 [50, 21, 56, 2]. However, current reasoning LLMs treat thinking and answering as strictly sequential processes – answers are available only after reasoning concludes.

Currently, reinforcement Learning (RL) [20] is the dominant approach to convert a base LLM into a reasoning LLM [22, 16, 13, 52]. Typically, the model is rewarded based on the correctness of the final answer and adherence to the reasoning format. The intermediate reasoning traces are often treated as a byproduct or unstructured chatter. In this work, we argue that such training paradigm is worth revisiting, especially for multi-hop reasoning tasks. First, users rarely have the time or cognitive bandwidth to thoroughly examine lengthy and often uninformative reasoning traces [44]. Yet, reasoning traces may include partial conclusions that are already beneficial to users; clearly presenting these conclusions early can enhance interaction [30]. Second, in the cases where the reasoning trace is not fully visible to the user, these partial conclusions can assist users in verifying or validating the model's final output. Third, these partial conclusions could also be utilized as dense supervision signals to further improve model's reasoning during training [27, 9]. 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. First, it is unclear whether models can learn and generalize interleaved behaviors across various complex tasks. Second, 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 novel RL training paradigm that enables LLMs to interleave thinking and answering, without leveraging *any* external tools. As shown in Figure 1, interleaved reasoning model generates informative intermediate answers during reasoning, giving timely feedback to the user (reducing TTFT) while providing verifiable reward signal to guide its own subsequent steps toward a correct final answer. We conduct comprehensive experiments on three popular RL algorithms (PPO [37], GRPO [38], and REINFORCE++ [17]), and found that LLMs are inherently capable of answering questions in an interleaved manner, but it is non-trivial to train them to systematically generate useful intermediate answers across diverse tasks. We apply a simple yet effective rule-based reward to encourage models to generate informative intermediate answers. We found that training only on question answering and logical reasoning datasets, models are able to generalize and conduct interleaved reasoning to unseen tasks such as MATH [15], GPQA [35], and MMLU [14]. We summarize our key contributions as follows:

- We propose a novel RL training paradigm that trains LLMs to alternate thinking and answering, inherently reducing Time-to-First-Token (TTFT) by over 80% on average.
- We introduce a rule-based reward that provides consistent, dense feedback for intermediate steps during training, guiding the model to stay on the correct thinking path and significantly improving its reasoning capability, resulting in averagely up to a 19.3% Pass@1 improvement over traditional think-answer reasoning.
- Our conditional reward strategy on intermediate reward allow us to train on datasets with intermediate answers and generalize strongly to unseen reasoning tasks. Comprehensive analysis reveals valuable and practical insights into reward modeling, stable RL training, and model reasoning dynamics.

2 Related Work

Reinforcement Learning for LLM Reasoning. In the context of LLMs, reinforcement learning [20] is widely used for human preferences alignment [7, 33, 23]. Recently, RL's usage has gradually shifted towards enhancing LLM's reasoning capabilities. Reward modeling is a strong means of guiding a model to learn new skills during RL [40]. There are primarily two type of rewards used during RL: Outcome Reward Model (ORM) and the Process Reward Model (PRM). DeepSeek R1 [13] demonstrates that simple rule-based ORM can significantly improve performance on challenging reasoning tasks. PRM are often used to provide denser feedback on intermediate steps [28, 46, 48]. However, they face significant practical challenges - they often require human annotation for generated output [28, 46], which inevitably introduces risks of reward hacking [34], requiring training a separate reward model [48] and adding complexity to the training pipeline [13]. In this work, we leverage the concept of PRM, but instead of relying on a separate learned model, we only 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 Section 5. We leverage a conditional reward scheme similar to Yuan et al. [55]. However, instead of focusing on reducing response length, our work focuses on improving the quality of intermediate reasoning.

LLM Reasoning and Efficiency. Research on enhancing LLMs' reasoning capabilities has followed several key directions. Early approaches focused on improving base or instruction-tuned LLMs through techniques like chain-of-thought prompting [50], self-consistency [29], and few-shot learning [3], while others explored structured reasoning through graph-based methods [2]. Another line of work leverages external tools and APIs [25, 11, 5] to augment model capabilities. Recent development in RL enable models like OpenAI-o1 [18] and DeepSeek-R1 [13] to generate long CoT to improve their reasoning ability. This shift towards longer reasoning also results in inefficiency 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 [54, 53, 43], length control RL [1, 10, 55], or additional finetuning [31]. Interleaving between reasoning with action using RL is also a newly emerged research area. Concurrent work mainly focuses on leveraging *external* tools such as search engine [19, 4, 41, 26] during the reasoning process. In contrast, we focus on model's *internal* ability of generating verifiable intermediate answers, which can be later used as additional reward signal for training.

3 Training LLMs for Interleaved Reasoning

In this section, we present our approach for training LLMs to interleave thinking and answering. We first formalize the interleaving process and then describe our reinforcement learning formulation.

3.1 Multi-hop Problem Decomposition

We conceptualize the process of answering a multi-hop question as a sequence of resolved intermediate steps. A "sub-answer" is a distinct, user-facing piece of information or partial conclusion that the model confidently derives at a given reasoning stage. The model should output a sub-answer when it identifies that a self-contained part of the problem has been solved or a meaningful milestone in reasoning has been reached. For example, in a multi-hop question, a sub-answer might resolve the first hop and guide the next. In a mathematical problem, it could be an intermediate calculation. The key is that each sub-answer is presented as a public and conclusive statement for that stage of the reasoning, allowing the overall response to be built incrementally.

3.2 Thinking vs. Answering

The distinction between thinking and answering requires careful consideration. From a philosophical perspective, thinking constitutes an integral component of answer formulation. 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. These conclusions may

represent partial solutions to the overall problem, but they are presented as complete intermediate steps that advance the user's understanding or problem-solving process.

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 $k \in \{1, \ldots, N\}$ index the steps. We denote the thinking segment by $y^{(k)}_{\text{think}}$ and the corresponding answer segment by $y^{(k)}_{\text{answer}}$. The interleaved generation thus is

$$y \;=\; y^{(1)}_{\text{think}} \circ y^{(1)}_{\text{answer}} \circ y^{(2)}_{\text{think}} \circ y^{(2)}_{\text{answer}} \circ \cdots \circ y^{(N)}_{\text{answer}}, \tag{1}$$

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

3.3 Interleaved Reasoning Template

To guide the model in adopting the interleaved reasoning process, we use a specific instruction template during training and inference. The template uses only two special tags: `<think></think>` and `<answer></answer>` to explicitly ask the model to perform reasoning and provide answers within each tag, respectively. We use the original template proposed in Guo et al. [13] for think-answer reasoning (Appendix A). The complete interleaved template is shown in Table 1.

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.

3.4 Reinforcement Learning for Interleaved Reasoning

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

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

where $\mathcal{D}$ is the training dataset, $\pi_{\text{ref}}(y \mid x)$ 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 Section 4 and compare the performance of different RL algorithms in Section 5. After training, the model should have learned how to dynamically switch between them based on the given task at each step.

3.4.1 Rule-based Rewards

To effectively train the model to reason within the interleaved format, we utilize three rule-based rewards: the **format 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 [13, 19], our reward design avoids complex neural reward models, instead focusing on simple rule-based reward that provide clear and consistent feedback without requiring separate reward model training. We discuss the methods to apply the intermediate reward in Section 3.4.3. More details about the rewards can be found in Appendix C.

3.4.2 Models Are Quick Format Learner

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.

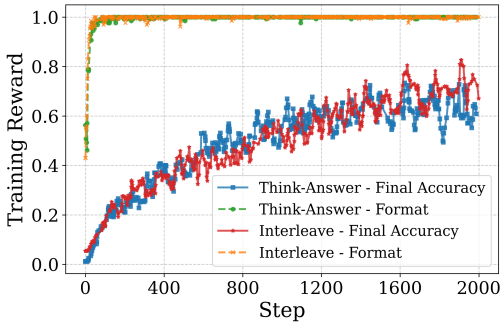

Figure 2: The format reward rapidly reaches a plateau during training, significantly faster than the accuracy reward, suggesting that LLMs naturally adopt structural patterns.

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.

3.4.3 Conditional Rewards

Our finding shows 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 (Section 5). 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 a conditional reward scheme where intermediate rewards are only invoked 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:

$$r_{\text{intermediate}}(x, y) \;=\; \mathbb{1}(C) \cdot \sum_{k=1}^{N-1} \text{Correct}\big(y_{\text{answer}}^{(k)}\big), \tag{3}$$

$$\textit{where } C = \text{FormatCheck}(y) \;\wedge\; \text{Correct}\big(y_{\text{answer}}^{(N)}\big) \;\wedge\; \big(\text{Acc}(B) > \text{Acc}(B-1) - \epsilon\big), \tag{4}$$

where $\text{Acc}(B)$ denotes the accuracy for the current training batch B, $\mathbb{1}(\cdot)$ is the indicator function, $\text{Correct}(y_{\text{answer}}^{(k)})$ evaluates the answer correctness at step k, and ϵ is the threshold for training stability. The batch accuracy criterion serves as a curriculum indicator, gradually introducing intermediate rewards as training progresses. Therefore, the overall reward function is:

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

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. We discuss different approaches to calculating the intermediate reward value in Section 3.4.4.

3.4.4 Intermediate Reward Calculation.

We explore different approaches to calculate intermediate reward under the conditional nature. While all approaches use the conditional scheme described above, they differ in how they calculate the actual reward value. We explore three approaches: **(1) All-or-None**, which requires all intermediate steps to be correct in sequence; **(2) Partial Credit**, which gives partial credit for individual correct intermediate steps; and **(3) Time-Discounted**, which assigns higher rewards to earlier correct intermediate steps while assigning extra rewards to the all correct intermediate steps. Note that the intermediate rewards calculation requires the intermediate ground truth answers. However, despite training *only* on datasets with intermediate ground truths, we are able to generalize to other unseen datasets (Section 4). We compare these approaches in Section 5, provide additional details in Appendix C.2, and present the complete algorithm in Algorithm 1.

4 Main Experiments

Datasets. We evaluate our method on both in-domain and out-of-domain datasets. For in-domain datasets, we use **Knights and Knaves (K&K)** [51] and **Musique** [45] for both training and evaluation. K&K is a logical reasoning dataset that requires multi-step reasoning to identify the correct characters. It consists of multiple problem difficulty levels depending on the number of characters involved. 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. We leave the exploration of dataset without intermediate ground truth for future work. For out-of-domain evaluation, we test on **GPQA** [35], **MMLU** [14], and **MATH** [15] to assess how well our models generalize to unseen tasks and domains. These datasets cover diverse reasoning scenarios, allowing us to comprehensively evaluate the robustness of our approach. More details about the datasets are provided in Appendix D.

Models and Baselines. We conduct experiments using Qwen2.5 instruct models with 1.5B and 7B parameters. To comprehensively evaluate the effectiveness of our approach, we compare it against verious baselines: **Direct Inference**, where the model generates answers without explicit reasoning steps; **Chain-of-Thought (CoT)** [50], where the model performs all reasoning before generating the final answer; **SFT** [8], where the model is trained with supervised fine-tuning; **Think-answer**, where we train same model with the standard think-answer RL methods proposed in Guo et al. [13]. 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 Section 3.4.3. For fair evaluation, we use the same setup (eg., datasets, RL algorithms, etc.) for think-answer and interleaved training.

Evaluation Metrics. In this work, 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 [32, 19], we use Exact Match (EM) 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 are provided in Appendix E.

Policy Optimization. To train the policy model, we experiment with three policy optimization approaches: the traditional Proximal Policy Optimization (PPO) [37] and it's two variants, Group Relative Policy Optimization (GRPO) [38] and REINFORCE++ [17]. The primary distinction between them lies in their approaches to advantage value estimation. Specifically, PPO utilizes a network to approximate the state value function, leveraging the Generalized Advantage Estimation [36] to derive the advantage. In contrast, GRPO and REINFORCE++ bypass the need for an extra critic network and reduce the resources required during training, In practice, PPO is more stable during training due to the need for a critic model, which requires additional warm-up steps before effective training begins. On the other hand, GRPO and REINFORCE++ are sample efficient but more sensitive to hyperparameters choices. We compare the performance of three optimization methods and discuss the results in detail in Section 5.

Training Details. We use Proximal Policy Optimization (PPO) as our primary training algorithm, as it provides more stable training compared to other RL algorithms. Different RL algrithm results can be found in Table 4. To ensure a fair comparison, we train models up to 2,000 steps and report the checkpoint that has the highest test score for both think-answer and interleaved training. For intermediate reward calculation, we use the Time-Discounted method as it shows better performance in our experiments (Detailed results can be found in Section 5). All experiments are conducted on eight H100 GPU with 80GB memory. More training details are provided in Appendix B.

Main Results. The results in Table 2 demonstrate the benefits of interleaved reasoning. Our base interleaved approach (Interleave), without using intermediate rewards, maintains Pass@1 accuracy comparable to the traditional think-answer baseline while drastically reducing TTFT by an average of 80.3% (1.5B) and 81.4% (7B). This means users receive informative responses nearly *five* times sooner, highlighting that the interleaved structure itself enhances responsiveness by default. The

Table 2: **Main results**: Comparison between proposed *interleaved reasoning* 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 bold for Pass@1, underlined for TTFT. For the non-reasoning baselines (Direct Inference, CoT, SFT) TTFT is naturally 0.

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↓
Qwen2.5-1.5B-Instruct												
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
Qwen2.5-7B-Instruct												
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 3: **Delayed Intermediate Answers**: 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.

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

significant improvement in Pass@1 accuracy occurs when intermediate rewards are introduced (Interleave + IR), leading to an average relative improvement with of 19.3% (1.5B) and 5.7% (7B) with TTFT reductions of 80.7% (1.5B) and 82.2% (7B). Moreover, training on only the datasets with intermediate ground truth, our method exhibits strong out-of-domain generalization across diverse reasoning tasks (GPQA, MMLU, and MATH), maintaining superior accuracy and reduced latency without *any* training data from that domain. These findings clearly indicate the effectiveness of interleaved reasoning in enhancing both model accuracy and utility in practical applications. We present a qualitative analysis of interleaved reasoning in Appendix F and examples in Appendix H.

5 Analysis and Discussions

Impact of Intermediate Answers. Using the Qwen2.5-1.5B-Instruct model, 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 3 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 with and without Intermediate Rewards (IR). The results across multiple datasets (K&K, GPQA, MMLU, MATH) clearly show that delaying the presentation of intermediate answers substantially lowers Pass@1 accuracy and increases TTFT, even when IR is applied. Furthermore, the benefits of IR are diminished in the delayed intermediate setting. This suggests that timely, incremental feedback *throughout the reasoning process* is key to the effectiveness of interleaved reasoning. Additional visualization and discussion in Section 5.

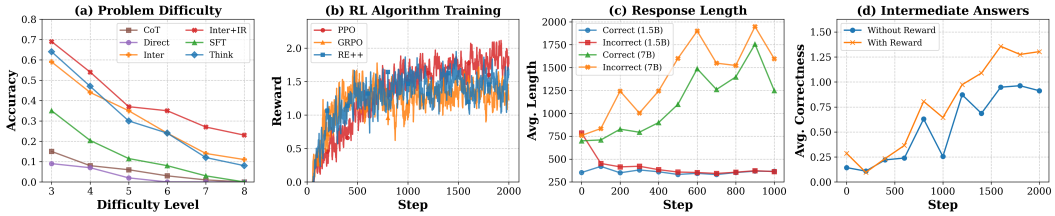

Figure 3: Comparative analysis of interleaved reasoning: (a) Performance gap widens on harder K&K problems as difficulty increases; (b) Training dynamics across different RL algorithms showing convergence patterns; (c) Response length analysis revealing correct answers are typically shorter; (d) Effect of intermediate rewards on model behavior showing increased correct intermediate answers.

Table 4: **RL algorithm performance**: 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.

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↓
GRPO												
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
REINFORCE++												
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
PPO												
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

Scaling to Harder Problems. The K&K dataset naturally contains multiple levels of problem difficulty, with the difficulty increasing as more characters are involved. We train a Qwen2.5-1.5B-Instruct model with datasets involving three, four, and five characters and evaluate on the full range of difficulties (three through eight; see Appendix D 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 practical speedups on TTFT but also improves overall reasoning, especially for harder multi-hop problems.

Different RL Algorithms. The results in Table 4 highlight the performance differences among the three RL algorithms. PPO consistently achieves higher Pass@1 scores across most tasks, though it generally requires more training steps to converge compare to 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 [19]. 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, our method (Interleave + IR) consistently outperforms the "think+answer" baseline, providing further evidence of its effectiveness.

Different Reward Strategies. We investigate the effectiveness of different intermediate reward strategies in Table 5. Results demonstrate 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 [24]. Conditional reward strategies (Section 3.4.3)

Table 5: **Reward strategy analysis**: 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.

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

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. More details about the reward strategies are in Appendix C.

Intermediate Reward Distribution In addition to Figure 3(d), we present Figure 4 to visualize how frequently intermediate rewards are applied during training. Notably, 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 reward is needed to effectively incentivize the model to produce better intermediate steps and ultimately improve final accuracy. 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.

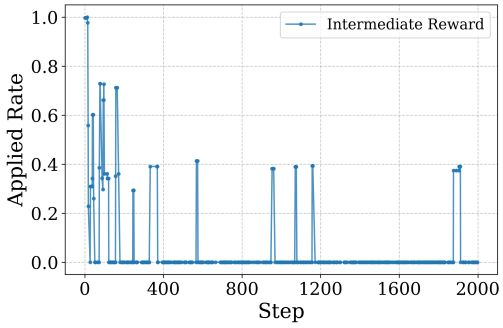

Figure 4: Visualization of intermediate reward application rate during training. The rate decreases as training progresses due to increasing batch accuracy thresholds.

Reasoning Pattern Analysis We analyze the response length of interleaved reasoning and present the results in Figure 3(c). We found that 7B and 1.5B models differ significantly in how their response length changes during training. While both model sizes achieve better performance (Table 2), 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 regarding the relationship between length and performance in reasoning LLMs [52, 47]. However, for both the 1.5B and 7B models, *correct answers are generally shorter than incorrect answers*. Consequently, the correct answers contain fewer thought tokens than the incorrect ones, suggesting that the model finds the correct solution paths more efficiently. Additional analysis in Appendix G.

Comparison with Process Reward Models Our 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.

6 Conclusion

We introduce interleaved reasoning, a novel reinforcement learning (RL) paradigm enabling reasoning LLMs to alternate thinking with generating structural intermediate answers. Our comprehensive experiments across five diverse datasets and three RL algorithms demonstrate significant practical benefits: an over 80% reduction in time-to-first-token (TTFT) on average and up to a 19.3% improvement in Pass@1 accuracy, without needing any external tools. We found that models are inherently able to perform interleaved reasoning, and we can further enhance this capability via RL. We propose a simple, rule-based conditional reward to incentivize correct intermediate steps and enhance the model's reasoning ability. The interleaved reasoning models, trained solely on logical reasoning and QA, generalize strongly to complex, unseen tasks including MATH, GPQA, and MMLU. Our analysis reveals several practical insights into reward modeling, RL training, and LLM reasoning dynamics. Overall, interleaved reasoning offers a compelling path to build LLMs that are more accurate and interactive.

References

[1] Pranjal Aggarwal and Sean Welleck. L1: controlling how long A reasoning model thinks with reinforcement learning. *CoRR*, abs/2503.04697, 2025. doi: 10.48550/ARXIV.2503.04697. URL https://doi.org/10.48550/arXiv.2503.04697.

[2] Maciej Besta, Niklas Blach, Vít Kubíček, Michael Gerstenberger, Matthew Gianinazzi, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI Conference on Artificial Intelligence*, 2023. URL https://arxiv.org/pdf/2308.09687.pdf.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. Henighan, R. Child, A. Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, I. Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://arxiv.org/pdf/2005.14165.pdf.

[4] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Fan Yang, Zenan Zhou, Weipeng Chen, Haofen Wang, Jeff Z Pan, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

[5] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2022. URL https://api.semanticscholar.org/CorpusId:253801709.

[6] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

[7] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

[8] 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, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling

instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53, 2024. URL https://jmlr.org/papers/v25/23-0870.html.

[9] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[10] Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.

[11] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *ArXiv*, abs/2211.10435, 2022. URL https://arxiv.org/pdf/2211.10435.pdf.

[12] 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. Are we done with mmlu?, 2024.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020. URL https://api.semanticscholar.org/CorpusID:221516475.

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

[16] Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*, 2025.

[17] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

[18] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[19] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

[20] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[21] Tushar Khot, H. Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *ArXiv*, abs/2210.02406, 2022. URL https://arxiv.org/pdf/2210.02406.pdf.

[22] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[23] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Raghavi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling. *CoRR*, abs/2403.13787, 2024. doi: 10.48550/ARXIV.2403.13787. URL https://doi.org/10.48550/arXiv.2403.13787.

[24] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[25] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[26] Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025.

[27] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[28] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[29] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55:1 – 35, 2021. URL http://dl.acm.org/citation.cfm?id=3560815.

[30] Xingyu Bruce Liu, Haijun Xia, and Xiang 'Anthony' Chen. Interacting with thoughtful ai. *ArXiv*, abs/2502.18676, 2025. URL https://api.semanticscholar.org/CorpusId:276617543.

[31] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *ArXiv*, abs/2501.12570, 2025. URL https://api.semanticscholar.org/CorpusId:275790112.

[32] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *ArXiv*, abs/2405.14734, 2024. URL https://api.semanticscholar.org/CorpusID:269983560.

[33] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022. URL https://api.semanticscholar.org/CorpusID:246426909.

[34] Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/e45caa3d5273d105b8d045e748636957-Abstract-Conference.html.

[35] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022, 2023. URL https://api.semanticscholar.org/CorpusID:265295009.

[36] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

[40] David Silver, Satinder Singh, Doina Precup, and R. Sutton. Reward is enough. *Artif. Intell.*, 299:103535, 2021. URL `https://api.semanticscholar.org/CorpusId:236236944`.

[41] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

[42] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[43] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025. doi: 10.48550/ARXIV.2501.12599. URL `https://doi.org/10.48550/arXiv.2501.12599`.

[44] Christoph Treude and Raula Gaikovina Kula. Interacting with ai reasoning models: Harnessing "thoughts" for ai-driven software engineering. *ArXiv*, abs/2503.00483, 2025. URL `https://api.semanticscholar.org/CorpusID:276741340`.

[45] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[46] Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022. doi: 10.48550/ARXIV.2211.14275. URL `https://doi.org/10.48550/arXiv.2211.14275`.

[47] Junlin Wang, Shang Zhu, Jon Saad-Falcon, Ben Athiwaratkun, Qingyang Wu, Jue Wang, Shuaiwen Leon Song, Ce Zhang, Bhuwan Dhingra, and James Zou. Think deep, think fast: Investigating efficiency of verifier-free inference-time-scaling methods. *arXiv preprint arXiv:2504.14047*, 2025.

[48] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

[49] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[51] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. *https://arxiv.org/abs/2410.23123*, 2024.

[52] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

[53] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *CoRR*, abs/2502.18600, 2025. doi: 10.48550/ARXIV.2502.18600. URL https://doi.org/10.48550/arXiv.2502.18600.

[54] Yuhui Xu, Hanze Dong, Lei Wang, Doyen Sahoo, Junnan Li, and Caiming Xiong. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025.

[55] Danlong Yuan, Tian Xie, Shaohan Huang, Zhuocheng Gong, Huishuai Zhang, Chong Luo, Furu Wei, and Dongyan Zhao. Efficient rl training for reasoning models via length-aware optimization, 2025. URL https://arxiv.org/abs/2505.12284.

[56] Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, D. Schuurmans, O. Bousquet, Quoc Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625, 2022. URL https://arxiv.org/pdf/2205.10625.pdf.

A Think-answer Template

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 6: Template for think-answer reasoning from Guo et al. [13]. prompt will be replaced with the specific reasoning question during training.

B Additional Training Details

All experiments were conducted using VERL [39], 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 evluate 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.

Table 7: Training hyperparameters used for our experiments.

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

C Reward Calculation

C.1 Individual Reward

Given the generated sequence y and the ground truth answer $g = \{g_1, g_2, ..., g_N\}$, 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:

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

   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:

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

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 3.4.3. 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. We present the full algorithm for intermediate reward calculation in Algorithm 1. Our evaluation in Section 5 shows that the Time-Discounted strategy performs best overall, balancing the need for early correct answers with maintaining coherent reasoning during the reasoning process.

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

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

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:

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

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 appears correct answers. Formally,

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

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 $|g|$ 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.

Algorithm 1 Intermediate Reward Calculation

Input: Generated sequence y, ground truth intermediate answers $g = \{g_1, g_2, ..., g_N\}$, current training batch B, reward strategy S
Parameters: Base reward value R_{base}, stable training threshold ϵ
Output: Intermediate reward value
Parse y to extract all intermediate answers $y_{\text{answer}} = \{y_{\text{answer}}^{(1)}, \ldots, y_{\text{answer}}^{(N)}\}$, where $y_{\text{answer}}^{(N)}$ is the final answer
is_final_correct $\leftarrow \text{Correct}(y_{\text{answer}}^{(N)})$
is_format_valid $\leftarrow \text{FormatCheck}(y)$
is_progressing $\leftarrow (\text{Acc}(B) > \text{Acc}(B-1) - \epsilon)$
if is_final_correct AND is_format_valid AND is_progressing **then**
    reward_sum $\leftarrow 0$
    if $S =$ "All-or-None" **then**
        all_correct $\leftarrow$ TRUE
        for $k = 1$ to $N - 1$ **do**
            if NOT $\text{Correct}(y_{\text{answer}}^{(k)})$ **then**
                all_correct $\leftarrow$ FALSE
                break
            end if
        end for
        if all_correct **then**
            reward_sum $\leftarrow R_{\text{base}}$
        end if
    else if $S =$ "Partial Credit" **then**
        for $k = 1$ to $N - 1$ **do**
            if $\text{Correct}(y_{\text{answer}}^{(k)})$ **then**
                reward_sum $\leftarrow$ reward_sum $+ R_{\text{base}}/N$
            end if
        end for
    else if $S =$ "Time-Discounted" **then**
        correct_step $\leftarrow \{\}$ {Track all correct steps}
        for $k = 1$ to $N - 1$ **do**
            for each required answer g_k in g **do**
                if g_j not in correct_step AND $\text{Correct}(y_{\text{answer}}^{(k)})$ **then**
                    correct_step$[g_j] \leftarrow i$
                end if
            end for
        end for
        if |correct_step| $= |g|$ **then**
            reward_sum $\leftarrow R_{\text{base}}$
        else
            sum_weights $\leftarrow \sum_{step \in correct\_step} 1/step$
            reward_sum $\leftarrow$ (sum_weights$/|g|) \cdot R_{\text{base}}$
        end if
    end if
    return reward_sum
else
    return 0
end if

D Dataset Details

D.1 In-Domain Datasets

Knights and Knaves (K&K). K&K is a logical reasoning dataset that requires multi-step reasoning to identify the correct characters [51]. 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.

Musique. Musique is a multi-hop question answering dataset that requires retrieving and combining information from multiple sources [45]. 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.

D.2 Out-of-Domain Datasets

GPQA. We use the GPQA-diamond version [35], which consists of 198 data points. GPQA (Graduate-level Physics Questions and Answers) is a challenging benchmark designed to evaluate LLMs on graduate-level physics problems, requiring advanced domain knowledge and multi-step reasoning.

MMLU. We use MMLU-redux-2.0, a cleaned and reannotated version of MMLU from [12]. 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.

MATH. We use 198 data points from the level 5 subset of MATH [15], 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.

E Evaluation Metrics

E.1 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 [50, 13, 19], 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 (removing punctuation, converting to lowercase, and standardizing numerical formats). A prediction is considered correct only if it exactly matches the normalized ground truth.

E.2 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:

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

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.

E.3 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 [52], which is as known as reward hacking. Therefore, we use EM as our main evaluation metric.

F Qualitative Analysis of Interleaved Reasoning

To complement our quantitative findings on significant time-to-first-token (TTFT) reduction, we conduct a qualitative evaluation using an LLM-based judge (`gpt-4o-mini-2024-07-18`) to assess the value of interleave reasoning. Specifically, we compared two versions of the interleaved method (with and without intermediate rewards) against the standard think-answer method. 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 in Appendix F.1.

We calculate the win rates for each method, as shown in Table 8. 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 in terms of both win rate and qualitative scores, highlighting the importance of intermediate rewards in enhancing the user experience.

Table 8: LLM-based qualitative evaluation: average win rates and average scores by domain.

	Think-Ans vs. Interleave		**Think-Ans vs. Inter+IR**	
Dataset Group	Think-Ans Win (%)	Inter Win (%)	Think-Ans Win (%)	Interleave+IR Win (%)
In-domain	36.7	**63.4**	43.4	**56.7**
Out-of-domain	**70.1**	29.9	**52.1**	47.9
Overall	**53.4**	46.7	48.6	**51.4**

F.1 LLM-Judge Evaluation Prompt

The following prompt was used to instruct the LLM judge for qualitative evaluation:

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>`

Table 9: Average response length for each reasoning method.

Dataset / Setting	Think-Answer	Interleave	Interleave+IR
GPQA	414.55	314.95	433.22
K&K	710.58	316.05	401.08
MATH	491.24	297.49	500.32
MMLU	277.60	191.07	329.35
Musique	110.89	113.21	165.10
Overall average	400.97	252.79	368.73
Correct answers	198.11	207.56	308.31
Incorrect answers	445.75	259.09	380.27

G Additional Response Length Analysis

Table 9 shows the average number of tokens produced by each method on every dataset, as well as the overall averages and the split between correct and incorrect responses. Interleaved reasoning produces the longer traces after applying intermediate rewards (Interleave+IR), though the traditional think-answer method generates longest answers in general. Pure interleaving (Interleave) is the most concise across the board. We also observe that shorter reasoning does not hurt accuracy: although

interleave reasoning produces the fewest tokens, it matches or exceeds Think-answer on every dataset (see main results in Table 2). Additionally, for every training recipe and model size we evaluated, correct solutions are consistently shorter than incorrect ones. This suggests that once the model has found a reliable path it stops generating unnecessary thoughts, whereas failure cases often involve extra turns spent exploring.

H Case Studies of Interleaved Reasoning

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

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>

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>

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.
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>

MMLU

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) $\nabla_w f(w) = X^T X w - X^T y + \lambda$
C) $\nabla_w f(w) = X^T X w - X^T y + \lambda w$
D) $\nabla_w f(w) = X^T X w - X^T y + (\lambda + 1)w$
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 cm^{-1}, strong sharp peak at 1720 cm^{-1}

^{1}H 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}Cl/^{37}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 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 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 cm^{-1}) and the C=O stretch (1720 cm^{-1}).</answer>
<think>Third, analyze the ^{1}H 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 ^{1}H 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 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 cm^{-1}), but the broad O-H from 3500-2700 cm^{-1} points to a carboxylic acid, not typically a phenol (phenols are usually sharper, 3200-3600 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 ^{1}H 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>

GPQA

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>