Pause or Fabricate? Training Language Models for Grounded Reasoning

Yiwen Qiu[1,2], Linjuan Wu[1,2], Yizhou Liu[1], Yuchen Yan[1], Xu Tan[3], Jin Ma[3], Yao Hu[2], Daoxin Zhang[2], Wenqi Zhang[1], Weiming Lu[1], Jun Xiao[1], Yongliang Shen[1]*
[1]Zhejiang University, [2]Xiaohongshu Inc., [3]Tencent.
{qiuyiwen,syl}@zju.edu.cn

Abstract

Large language models have achieved remarkable progress on complex reasoning tasks. However, they often implicitly fabricate information when inputs are incomplete, producing confident but unreliable conclusions—a failure mode we term ***ungrounded reasoning***. We argue that this issue arises not from insufficient reasoning capability, but from the lack of ***inferential boundary awareness***—the ability to recognize when the necessary premises for valid inference are missing. To address this issue, we propose **Grounded Reasoning via Interactive Reinforcement Learning (GRIL)**, a multi-turn reinforcement learning framework for grounded reasoning under incomplete information. GRIL decomposes the reasoning process into two stages: clarify and pause, which identifies whether the available information is sufficient, and grounded reasoning, which performs task solving once the necessary premises are established. We design stage-specific rewards to penalize hallucinations, enabling models to detect gaps, stop proactively, and resume reasoning after clarification. Experiments on GSM8K-Insufficient and MetaMATH-Insufficient show that GRIL significantly improves premise detection (up to 45%), leading to a 30% increase in task success while reducing average response length by over 20%. Additional analyses confirm robustness to noisy user responses and generalization to out-of-distribution tasks.

1 Introduction

Large language models (DeepSeek-AI et al., 2025; OpenAI, 2023; Zhou et al., 2024; Yang et al., 2024) have demonstrated remarkable capabilities in complex reasoning tasks. Reinforcement learning post-training paradigms, including RLHF (Ouyang et al., 2022) and RLVR (DeepSeek-AI et al., 2025), have further improved performance on standard benchmarks where models receive well-formed, information-complete inputs (Shao et al., 2024; Fu et al., 2025; Yan et al., 2025).

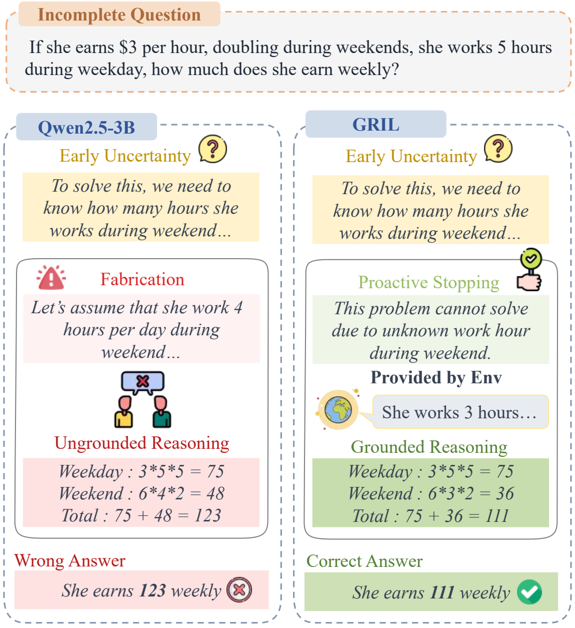

Figure 1: Comparison of reasoning behavior on problems with missing premises. Given an incomplete problem, both models initially detect the information gap. The base model fabricates the missing value and proceeds with ungrounded reasoning, producing an incorrect answer. GRIL-trained model proactively stops and requests clarification, then performs grounded reasoning after receiving the missing premise.

However, this evaluation paradigm rests on a strong implicit assumption: all necessary information is provided in a single turn. In real-world interactions, this assumption rarely holds. Users often provide information incrementally, expressing requirements ambiguously, or omitting critical details altogether (Luo et al., 2025; Gan et al., 2024). In such settings, the challenge is not merely to reason correctly, but to determine *whether reasoning is possible at all* given the available information.

When confronted with problems that lack nec-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 23056–23077
July 2-7, 2026 ©2026 Association for Computational Linguistics

essary premises, current models rather than halting or requesting clarification, they frequently fabricate the missing information and proceed with elaborate reasoning chains built on invented foundations (Fan et al., 2025; Huang et al., 2025). As illustrated in Figure 1, we refer to this phenomenon as *ungrounded reasoning*: the continuation of inference without sufficient grounding information. Importantly, ungrounded reasoning is not a failure of logical capability. The failure occurs because the model reasons without checking whether the premises are valid, resulting in outputs that appear coherent yet are fundamentally unreliable (Sui et al., 2025; Wang et al., 2025a).

Our empirical analysis reveals that ungrounded reasoning follows a characteristic trajectory, as examples shown in Figure 1 (Left). Models often express early uncertainty through phrases such as "we need to know" or "this requires," typically within the first half of their responses. Instead of stopping, however, they rapidly transition to fabrication using phrases like "let us assume" and then continue generating long chains of baseless inference. We capture this behavior using two metrics: *GapRatio*, which measures the proportion of tokens generated after the first uncertainty signal, and *Premise Detection Rate*, which measures how often models correctly identify inputs as incomplete. In preliminary studies, we observe GapRatios exceeding 47.7% and Premise Detection Rates below 41%. This suggests that models often override early uncertainty signals, proceeding with extended inference instead of stopping to request missing information.

Why does this behavior persist? Standard reinforcement learning frameworks reward answer production but provide no mechanism to recognize when no valid answer exists. As a result, models learn to answer rather than abstain, especially when trained on well-formed, single-turn problems (Sharma et al., 2023; Rita et al., 2024). This implies that ungrounded reasoning reflects a failure of *inferential control*: models cannot reliably decide when to stop and seek clarification (Li et al., 2023; Kadavath et al., 2022; Xiong et al., 2024). We thus cast the problem as a choice between two inferential actions: *Grounded Reasoning*, treating the input as sufficient and answering, or *Clarify and Pause*, identifying missing premises and halting until clarification is obtained (Kuhn et al., 2022; Wang et al., 2023b). Building on this insight, we introduce **Grounded Reasoning via Interactive Reinforcement Learning (GRIL)**, a multi-turn RL framework that explicitly trains models to recognize the boundaries of valid inference. GRIL decomposes reasoning into *clarify and pause*, and *grounded reasoning*, with an interactive training environment where correct identification of missing information triggers provision of the missing premise. In the clarify and pause stage, the model interacts with the environment to assess information sufficiency, earning a time-decayed reward for early detection of missing premises. Upon successful detection, the environment supplies the missing information and the model transitions to the grounded reasoning stage, where it solves the task under the completed context and receives a reward only for correct answers. Through stage-specific rewards with temporal decay, GRIL penalizes delayed detection and discourages hallucinated reasoning.

We evaluate GRIL across multiple model scales on mathematical reasoning benchmarks with systematically removed premises. On GSM8K-Insufficient, GRIL yields substantial improvements on Qwen2.5-1.5B, increasing the Premise Detection Rate from 4.6% to 90.8% and the Success Rate from 1.8% to 61.6%. At the same time, response length decreases by over 40%, reflecting earlier termination of ungrounded inference. Notably, GRIL does not degrade performance on complete problems, and achieves gains of up to 7% on standard GSM8K. Further analyses confirm robustness to noisy user responses and generalization beyond mathematical reasoning.

Our contributions are: (1) We identify *ungrounded reasoning* as a distinct failure mode with quantitative metrics for measurement. (2) We reframe reasoning under uncertainty as sequential decision-making, establishing that inferential boundary awareness is distinct from reasoning capability. (3) We propose GRIL, a multi-turn RL framework with stage-specific rewards for premise detection and grounded solving. (4) Through extensive experiments, we show that GRIL substantially reduces ungrounded reasoning while also improving performance on standard benchmarks.

2 Related Work

Reasoning in Large Language Models. Large language models have demonstrated remarkable reasoning capabilities through various prompting and training strategies (Lightman et al., 2023; Wang et al., 2024; Shinn et al., 2023). Chain-of-thought prompting (Wei et al., 2022) and its variants (Yao et al., 2023a; Besta et al., 2024; Wang et al., 2023a) elicit step-by-step reasoning by generating intermediate steps before final answers. Reinforcement learning has further improved reasoning through RLHF (Ouyang et al., 2022; Bai et al., 2022) and process-based reward models (Luo et al., 2024; She et al., 2025; Khalifa et al., 2025). However, recent studies reveal limitations when inputs are incomplete. Fan et al. (2025); Liu et al. (2025) show that reasoning models tend to exhibit overthinking when facing problems with missing premises, often fabricating assumptions rather than recognizing information insufficiency. This tendency appears more pronounced in models trained with RL, where optimization pressure to produce answers may override uncertainty recognition. Our work addresses this by explicitly training models to detect missing premises and request clarification.

Multi-Turn RL for Interaction and Reasoning. Multi-turn interaction (Li et al., 2025; Jin et al., 2025) presents challenges for language models. Laban et al. (2025) demonstrate that when information is distributed across multiple turns, model performance degrades significantly compared to single-turn settings, with drops averaging 30%. Missing premise scenarios represent a special case where critical information arrives only after explicit request. Multi-turn RL has been explored for dialogue (Madaan et al., 2023; Gao, 2025; Jiang et al., 2025) and tool-augmented agents (Schick et al., 2023; Yao et al., 2023b; Zhang et al., 2025), with frameworks like RAGEN (Wang et al., 2025b) addressing long-horizon optimization. Clarification behavior has been studied in conversational AI (Hu et al., 2020; Zhang et al., 2024; Hao et al., 2025), while Liu et al. (2025) propose reverse reasoning to detect missing information. Our work differs by framing the problem as learning inferential control through interactive RL, training models to detect, stop, and integrate missing information within a unified framework.

3 Theoretical Analysis

3.1 Modeling the Problem as an MDP

Standard LLM training optimizes next-token likelihood:

$$\max_{\theta} P(y \mid x)$$

Under incomplete information settings (x_{inc}), this objective forces the model to imitate the distribution of answers conditioned on insufficient context. As a result, the model is incentivized to produce plausible completions even when the problem is under-specified, leading to hallucination.

GRIL reformulates reasoning as a Markov Decision Process (MDP):

$$M = (S, A, R, P)$$

and optimizes the expected return:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right]$$

This shifts the objective from imitating answers to maximizing long-term utility. In incomplete contexts, the optimal policy is no longer to guess an answer, but to take an action that increases expected return, namely issuing a clarification when appropriate.

3.2 Reward Function Constraints and Information Gain

The time-decay detection reward introduces an asymmetric structure: early detection yields a higher reward, while delayed detection reduces the reward via the decay factor γ. Furthermore, ungrounded answering incurs an explicit penalty, and unnecessary clarification is also penalized.

We formalize the learning objective as follows. GRIL guarantees the learning of **Inferential Boundary Awareness** if and only if:

$$R(a_c \mid q \in D_{inc}) > R(a_s \mid q \in D_{inc})$$

where a_c denotes the clarify action, a_s denotes the solve action, and D_{inc} denotes the incomplete-query set.

Under incomplete problems, the optimal policy is determined by relative expected returns. By explicitly setting $R(a_s) \leq 0$ (penalizing ungrounded answering) and $R(a_c) > 0$ (positive, possibly time-decayed reward), we obtain the reward ordering:

$$\mathbb{E}[R \mid a_c] > \mathbb{E}[R \mid a_s]$$

Under standard policy gradient updates (e.g., PPO), the objective gradient satisfies:

$$\nabla J(\theta) = \mathbb{E}\left[A(s,a) \nabla \log \pi_\theta(a \mid s)\right]$$

Thus, if $A(s, a_c) > 0$, the probability of a_c increases; conversely, if $A(s, a_s) < 0$, the probability of a_s decreases. As long as the reward gap remains strictly positive, the induced gradient landscape ensures:

$$\pi_\theta(a_c \mid q \in D_{inc}) \to 1$$

at convergence (assuming sufficient exploration and optimization stability).

Importantly, the time-decay factor only scales the magnitude of the reward but does not reverse the reward ordering. Therefore, clarification remains the optimal action under insufficient information, while earlier detection yields higher return.

In this sense, GRIL does not heuristically encourage clarification—it structurally makes clarification the unique optimal action under incomplete contexts.

4 Method

We present **Grounded Reasoning via Interactive Reinforcement Learning (GRIL)**, a multi-turn RL framework that trains language models to recognize inferential boundaries and reason only when sufficient information is available. As illustrated in Figure 2, the core insight is to decompose reasoning into two distinct stages: *clarify and pause*, where the model determines whether the input contains sufficient information, and *grounded reasoning*, where the model produces a solution only when necessary premises are present. We formalize this as an interactive decision process with stage-specific rewards that penalize ungrounded inference and encourage appropriate clarification-seeking behavior.

4.1 Problem Formulation

Consider a reasoning task where the model receives a problem statement that may or may not contain all information necessary for solution. We formalize this as a multi-turn Markov Decision Process $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$.

State Space. At turn t, the state s_t comprises the complete interaction history:

$$s_t = (u_1, a_1, u_2, a_2, \dots, u_t) \tag{1}$$

where u_i denotes environment messages (initial problem, feedback, or clarifying information) and a_i denotes model responses.

Action Space. Although model outputs are natural language sequences, they serve functionally distinct roles in our framework. We abstract model behavior into two inferential actions:

- **Solve** (a_{solve}): The model commits to the current information as sufficient and produces a solution attempt.
- **Clarify** (a_{clarify}): The model identifies that critical information is missing and explicitly requests the needed premise.

This abstraction captures the key insight that *choosing to reason* is itself a decision contingent on available information. Ungrounded reasoning occurs precisely when models select a_{solve} in states where a_{clarify} is appropriate.

Problem Types. We denote the set of incomplete problems (with missing premises) as $\mathcal{D}_{\text{inc}}$ and complete problems (with sufficient information) as $\mathcal{D}_{\text{comp}}$. For $q \in \mathcal{D}_{\text{inc}}$, there exists a missing premise p_q that, when provided, renders the problem solvable. The goal is to learn a policy π_θ that selects appropriate actions based on input completeness.

4.2 Interactive Environment

A central challenge in training inferential control is that action quality cannot be assessed from immediate outputs alone. A model that selects a_{solve} on an incomplete problem may generate internally coherent reasoning, yet the entire inference chain lacks validity. To enable learning from the consequences of inferential choices, we design an interactive environment with asymmetric feedback dynamics.

Transition Dynamics. The environment responds differently based on problem type and model action. For incomplete problems $q \in \mathcal{D}_{\text{inc}}$, selecting a_{clarify} with correct detection triggers provision of the missing premise p_q, while selecting a_{solve} yields negative feedback. For complete problems $q \in \mathcal{D}_{\text{comp}}$, selecting a_{solve} with a correct answer terminates the episode successfully, while selecting a_{clarify} receives indication that no additional information is needed.

Formally, let $\mathcal{T}$ denote the transition function and $\oplus$ denotes concatenation to the dialogue history. For $q \in \mathcal{D}_{\text{inc}}$:

$$\mathcal{T}(s_t, a_t) = \begin{cases} s_t \oplus p_q & a_t = a_{\text{clarify}} \\ s_t \oplus \texttt{[neg.]} & a_t = a_{\text{solve}} \end{cases} \tag{2}$$

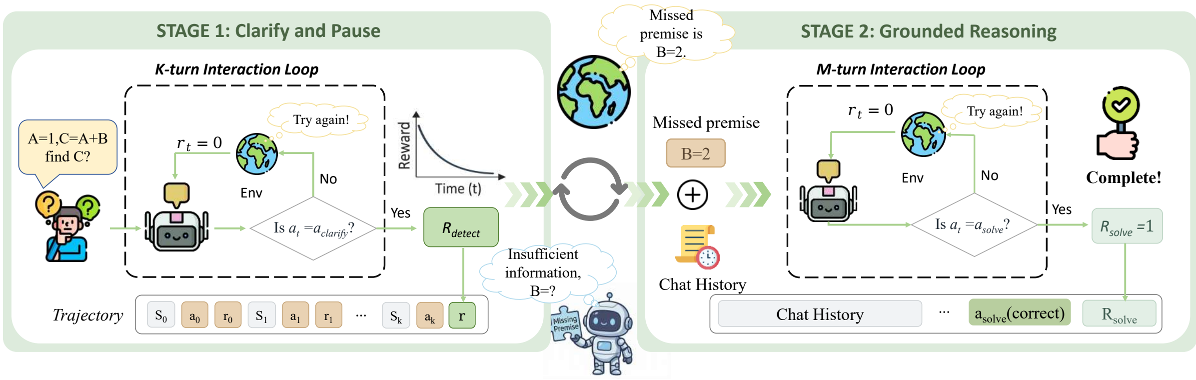

Figure 2: **Overview of the GRIL framework.** The training process consists of two stages. **Stage 1 (Clarify and Pause)**: Given an incomplete problem, the model engages in a multi-turn interaction loop. If the model attempts to solve without sufficient information, it receives negative feedback and zero reward. When the model correctly identifies insufficient information and requests clarification, it receives a detection reward $R_{\text{detect}} = \gamma^t$ that decays with the number of interaction turns, encouraging early detection. **Stage 2 (Grounded Reasoning)**: The environment provides the missing premise, which is concatenated with the chat history. The model integrates this information to produce the final answer. A correct solution yields the solving reward R_{solve}, otherwise the model may retry until maximum turns are reached.

For $q \in \mathcal{D}_{\text{comp}}$:

$$\mathcal{T}(s_t, a_t) = \begin{cases} \texttt{[done]} & \text{if } a_t = a_{\text{solve}} \\ s_t \oplus \texttt{[unc.]} & \text{if } a_t = a_{\text{clarify}} \end{cases} \tag{3}$$

where `[neg.]` denotes negative feedback, `[done]` denotes successful termination, and `[unc.]` denotes unnecessary clarification. This design creates a learning signal where correct premise detection leads to receiving the missing information, enabling the model to complete valid reasoning. The asymmetry ensures that clarification is beneficial only when genuinely warranted, preventing degeneration into overly conservative strategies.

4.3 Stage-Specific Reward Design

We design a reward function that explicitly decomposes the reasoning process into *clarify and pause*, and *grounded reasoning*, with separate incentives for each stage.

Stage 1: Pause and Clarify. For incomplete problems, we reward early and accurate identification of missing information. Let n_{prior} denote the number of interaction turns generated before the model requests clarification and stops proactively. The detection reward is:

$$R_{\text{detect}} = r_{\text{base}} \cdot \gamma_d^{n_{\text{prior}}} \tag{4}$$

where r_{base} is the base detection reward and $\gamma_d \in (0, 1)$ is a temporal decay factor. This formulation directly addresses the "early suspicion, late action" pattern: models that recognize missing information early receive higher rewards than those that generate extensive ungrounded tokens before requesting clarification. The exponential decay provides strong incentive to minimize n_{prior}.

Stage 2: Grounded Reasoning. After receiving clarifying information, we reward successful problem completion:

$$R_{\text{solve}} = r_{\text{correct}} \cdot \mathbb{1}[\text{answer is correct}] \tag{5}$$

This ensures that premise detection is not an end in itself but a means toward successful task completion. Models must learn to integrate newly provided information with the original context to produce valid solutions.

Complete Problem Handling. For problems in $\mathcal{D}_{\text{comp}}$, we apply standard outcome-based rewards with a penalty for unnecessary clarification:

$$R_{\text{comp}} = r_{\text{correct}} \cdot \mathbb{1}[\text{correct}] - \lambda \cdot \mathbb{1}[\text{unc.}] \tag{6}$$

where $\mathbb{1}[\text{unc.}]$ indicates unnecessary clarification on complete problems. The penalty coefficient λ prevents models from adopting a trivial strategy of always requesting clarification regardless of input completeness.

Overall Objective. The total reward for a trajectory τ on problem q is:

$$R(q, \tau) = \begin{cases} \alpha \cdot R_{\text{detect}} + \beta \cdot R_{\text{solve}} & \text{if } q \in \mathcal{D}_{\text{inc}} \\ R_{\text{comp}} & \text{if } q \in \mathcal{D}_{\text{comp}} \end{cases} \tag{7}$$

where α and β control the relative importance of detection versus solving. We optimize using Proximal Policy Optimization (PPO) (Schulman et al., 2017) with KL regularization:

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\tau\sim\pi_\theta}\left[R(q,\tau)\right] - \beta_{\mathrm{KL}} \cdot D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}) \tag{8}$$

where π_{ref} is the reference model and β_{KL} controls deviation from the initial policy.

4.4 Training Protocols

Data Construction. We construct incomplete problems by systematically removing critical premises from well-formed mathematical problems. For each problem, we identify sentences containing numerical values, remove one such sentence, and verify via automated checking that the modified problem becomes unsolvable without the removed information. The removed sentence is retained as p_q for use during interactive training. We also randomly sampled a subset of questions for manual annotation verification. Details are provided in Appendix.

Data Composition. Training data comprises a balanced mixture of incomplete and complete problems with ratio 1:1. This balance is critical: training exclusively on incomplete problems would bias models toward over-detection, while training only on complete problems provides no signal for learning clarification behavior.

Output Format. We enforce structured generation with <think>...</think> tags for reasoning traces and <answer>...</answer> tags for final responses. When requesting clarification, models output "insufficient information" in the answer field . This structured format enables unambiguous identification of inferential actions.

5 Experiments

5.1 Experimental Setup

Models and Training. We evaluate across four model scales: Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct (Yang et al., 2024), Qwen3-0.6B, and Qwen3-1.7B (Yang et al., 2025). All models are trained using the verl (Sheng et al., 2025) framework with maximum turns $T = 4$, reward weights $\alpha = 0.3$, $\beta = 0.7$, temporal decay $\gamma_d = 0.5$, KL coefficient $\beta_{\mathrm{KL}} = 0.01$, and penalty coefficient$\lambda = 2$. Training data consists of incomplete and complete problems mixed at 1:1 ratio.

Datasets. We construct two evaluation benchmarks: **GSM8K-Insufficient** and **MetaMATH-Insufficient**, derived from GSM8K (Cobbe et al., 2021) and MetaMATH (Yu et al., 2024) by systematically removing critical premises. Notably, there is no overlap between the training and test sets. We also evaluate on standard GSM8K and MATH500 to assess reasoning capability on complete problems. Details are in Appendix A.6.

Baselines. We compare against three settings: (1) **Base**: the pretrained model with zero-shot prompting; (2) **Prompt**: the base model with explicit instructions indicating that problems may contain missing information; (3) **SFT**: supervised fine-tuning on the same data distribution as GRIL. Detailed SFT training process is displayed in Appendix A.4.1

Metrics. We report four metrics: *Success Rate (SR)*, the proportion of problems solved correctly after clarification if needed; *Premise Detection (PD)*, the proportion of incomplete problems correctly identified; *Numbers of Iteraction Turns (NT)*, the average interaction turns required; and *Response Length*, the average tokens generated.

5.2 Main Results

Table 1 presents results on GSM8K-Insufficient and MetaMATH-Insufficient across model scales. GRIL consistently outperforms all baselines on both benchmarks.

GRIL Dramatically Improves Premise Detection and Task Success. Across all model scales, GRIL achieves substantial gains in both identifying incomplete inputs and ultimately solving problems. On Qwen2.5-1.5B, Success Rate improves from 1.8% (Base) to 61.6%, a 34× improvement, while Premise Detection rises from 4.6% to 90.8%. The gains persist at larger scales: Qwen2.5-3B achieves 73.5% SR compared to 62.7% for SFT, and Qwen3-1.7B reaches 72.8% SR with 96.5% detection rate.

GRIL Outperforms Supervised Learning on Identical Data. GRIL consistently outperforms SFT across various settings. On Qwen2.5-1.5B, the SR gap is 30 percentage points (61.6% vs 31.6%); on MetaMATH-Insufficient, GRIL achieves 72.5% compared to 56.1% for SFT on Qwen2.5-3B. This shows that multi-turn RL lets the model learn from feedback after it makes a choice, and this kind of learning signal can't be obtained from imitation learning

Model	GSM8K-Insufficient				Metamath-Insufficient			
	SR ↑	PD ↑	NT ↓	Length ↓	SR ↑	PD ↑	NT ↓	Length ↓
Qwen2.5-1.5B-Instruct								
Base Model	1.8	4.6	3.828	810	2.2	4.2	3.792	937
w/ Prompt	21.0	45.7	3.303	579	19.3	48.1	3.426	743
w/ SFT	31.6	64.6	3.202	750	31.1	60.4	3.248	999
GRIL(Ours)	**61.6**	**90.8**	**2.913**	**479**	**58.4**	**88.2**	**2.941**	**581**
Qwen2.5-3B-Instruct								
Base Model	20.6	28.0	3.665	887	15.5	20.9	3.722	1091
w/ Prompt	54.9	81.5	2.919	635	51.5	77.1	2.985	824
w/ SFT	62.7	**90.3**	2.706	575	56.1	83.9	2.800	750
GRIL(Ours)	**73.5**	88.0	**2.481**	**448**	**72.5**	**86.6**	**2.473**	**624**
Qwen3-0.6B								
Base Model	16.0	24.2	3.737	1668	14.7	25.6	3.793	1907
w/ Prompt	44.6	79.2	**2.763**	1112	40.6	75.7	**2.869**	1345
w/ SFT	48.4	82.4	2.971	**832**	41.3	**81.8**	3.077	**1003**
GRIL(Ours)	**52.3**	**84.6**	2.850	1269	**45.2**	79.2	3.025	1616
Qwen3-1.7B								
Base Model	41.3	52.6	3.271	1271	62.0	90.5	2.585	929
w/ Prompt	66.7	88.9	2.481	744	64.2	86.1	2.609	966
w/ SFT	63.6	92.8	2.482	704	63.0	90.3	2.531	874
GRIL(Ours)	**72.8**	**96.5**	**2.348**	**376**	**65.8**	**95.2**	**2.488**	**502**

Table 1: Results on GSM8K-Insufficient and MetaMATH-Insufficient across model scales. SR: Success Rate (%), PD: Premise Detection (%), NT: Numbers of Interaction Turns, Length: Response Length in tokens.

Reduced Ungrounded Reasoning Yields Efficiency Gains. GRIL reduces inference cost alongside accuracy gains. Response length drops substantially—Qwen2.5-1.5B by 41% (810→479 tokens) and Qwen3-1.7B by 70% (1271→376 tokens)—with fewer interaction turns, indicating earlier premise detection. The exception is Qwen3-0.6B, whose response length increases despite accuracy gains, suggesting a capacity threshold around 1B parameters below which models require more tokens to articulate clarifications and integrate multi-turn context.

6 Analysis

We conduct detailed analyses to understand the mechanisms behind GRIL's improvements. Our investigation covers five aspects: reduction in ungrounded reasoning, performance on complete problems, discrimination between complete and incomplete inputs, decomposition of performance gains, and generalization capabilities.

GRIL Substantially Reduces Ungrounded Reasoning. To quantify the severity of ungrounded reasoning, we measure the *GapRatio*[1]: the proportion of tokens generated after the model first expresses uncertainty. A higher ratio indicates more severe ungrounded inference. We evaluate on three public datasets from Fan et al. (2025). As shown in Figure 3, GRIL reduces this metric across all evaluation datasets. On SVAMP, the GapRatio drops from 0.63 to 0.27; on Formula, from 0.43 to 0.40; on GSM8K, from 0.37 to 0.30. Simultaneously, Premise Detection Rate increases substantially: Formula rises from 30% to 99%, SVAMP from 49% to 88.7%, and GSM8K from 45% to 71.8%. These results confirm that GRIL trains models to recognize inferential boundaries early and act on that recognition, rather than suppressing uncertainty signals in favor of continued generation.

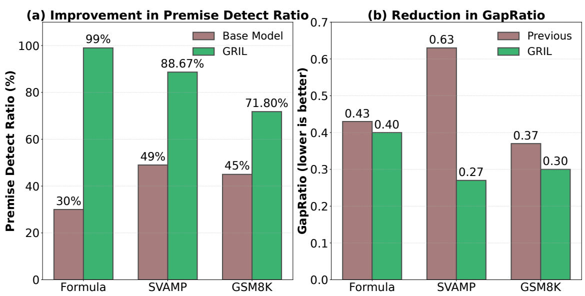

Figure 3: Premise Detection Rate and GapRatio before and after GRIL training on three public benchmarks from Fan et al. (2025).

GRIL Improves Performance on Complete Problems. A natural concern is whether training for premise detection compromises reasoning ability on well-formed inputs. Figure 4 demonstrates

[1] Formally, GapRatio $= (T_{\text{total}} - T_{\text{suspect}})/T_{\text{total}}$, where T_{total} is the total tokens generated and T_{suspect} is the token position where the model first expresses uncertainty.

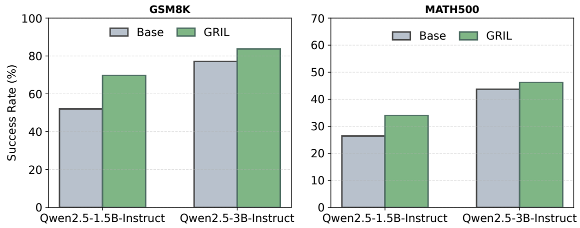

Figure 4: Success Rate on standard complete benchmarks (GSM8K and MATH500) comparing base models with GRIL-trained variants.

Model	Recall	Precision	F1
Qwen2.5-1.5B-Instruct	0.419	0.951	0.581
+ GRIL	**0.861**	**0.961**	**0.908**
Qwen2.5-3B-Instruct	0.756	0.968	0.847
+ GRIL	**0.886**	**0.981**	**0.931**

Table 2: Premise detection performance (Recall, Precision, F1) on a mixed dataset containing both complete and incomplete queries.

that GRIL not only preserves but enhances standard reasoning performance. On GSM8K, Qwen2.5-1.5B improves from 52.0% to 69.7% (+17.7 points), while Qwen2.5-3B rises from 77.1% to 83.7%. On the more challenging MATH500 benchmark, gains are also consistent: 26.4% to 34.0% for the 1.5B model, 43.7% to 46.2% for the 3B model. We hypothesize that learning to distinguish sufficient from insufficient information induces more careful reasoning: models that must decide whether to proceed learn to assess problem structure more thoroughly, benefiting downstream solving even when all information is present.

GRIL Robustly Distinguishes Complete and Incomplete Inputs. Effective inferential control requires high sensitivity to missing premises (Recall) without false alarms on complete problems (Precision). We construct a mixed evaluation set containing equal proportions of complete and incomplete queries and measure standard classification metrics. As shown in Table 2, GRIL achieves substantial Recall gains: Qwen2.5-1.5B improves from 0.419 to 0.861 (+105%), Qwen2.5-3B from 0.756 to 0.886. Crucially, Precision remains above 0.96 and even improves slightly, indicating that GRIL does not achieve sensitivity by becoming overly conservative. The resulting F1 scores (0.908 and 0.931) demonstrate that GRIL learns a precise decision boundary, effectively filtering incomplete queries while maintaining grounded reasoning.

Both Detection and Integration Capabilities Improve. To decompose performance gains, we design an experiment where models first respond normally, then receive forced environmental feedback regardless of their initial action. We analyze two conditional success rates: *Detected Correct Ratio (DCR)*[2], measuring success after correct premise identification, and *Non-detected Correct Ratio (NCR)*[3], measuring recovery via forced feedback despite initial detection failure. As shown in Table 3, GRIL improves both metrics. DCR for Qwen2.5-1.5B nearly doubles (37.3% $\rightarrow$ 66.9%), validating the effectiveness of the clarify-then-solve cycle. NCR also improves substantially (36.9% $\rightarrow$ 48.2%), indicating that GRIL enhances general context integration capabilities beyond just detection. This dual improvement explains why GRIL outperforms baselines by large margins: it strengthens both the ability to pause appropriately and the ability to reason effectively once sufficient information is available.

Model	GSM8K-Insufficient		MetaMath-Insufficient	
	DCR (%)	NCR (%)	DCR (%)	NCR (%)
Qwen2.5-1.5B-Instruct	37.3	36.9	36.0	37.3
+ GRIL	**66.9**	**48.2**	**65.7**	**57.9**
Qwen2.5-3B-Instruct	63.7	54.2	67.3	52.8
+ GRIL	**80.8**	**63.6**	**83.7**	**57.4**

Table 3: Decomposition of Success Rate into Detected Correct Ratio (DCR) and Non-detected Correct Ratio (NCR) on GSM8K-Insufficient and MetaMATH-Insufficient.

GRIL Generalizes to Out-of-Distribution Domains. To assess whether learned inferential control transfers beyond mathematical reasoning, we evaluate on modified HotpotQA (multi-hop reasoning) and CommonsenseQA (commonsense reasoning) datasets (Yang et al., 2018; Arora et al., 2023), where we apply the same premise removal procedure used for mathematical benchmarks. As shown in Figure 5, GRIL consistently improves both Premise Detection and Success Rate in these unseen domains. Smaller models (1.5B) show particularly pronounced gains, with Success Rate improvements exceeding 18 percentage points on CQA. The improvement trajectories reveal that GRIL pushes models toward the upper-right region of the detection-success space across both in-domain and out-of-domain tasks. This transfer

[2] DCR=$P(\text{success} \mid \text{detected})$
[3] NCR=$P(\text{success} \mid \text{not detected})$

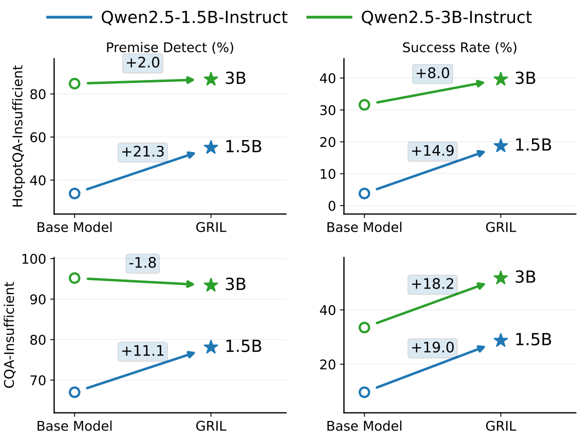

Figure 5: Out-of-distribution generalization on HotpotQA-Insufficient and CQA-Insufficient. Arrows indicate improvement trajectories from base models to GRIL-trained variants.

Model	Noisy Feedback	Uninformative Response
Qwen2.5-1.5B-Instruct	12.8	62.5
+ GRIL	**47.2**	**86.7**
Qwen2.5-3B-Instruct	51.4	89.1
+ GRIL	**69.7**	**89.5**

Table 4: Success Rate (%) under noisy feedback and uninformative user response conditions.

suggests that GRIL instills a general capacity for inferential boundary awareness rather than domain-specific heuristics.

GRIL Is Robust to Noisy User Interactions. Real-world interactions are imperfect: users may provide irrelevant information or fail to supply requested premises. We simulate two challenging conditions: *Noisy Feedback*, where clarifications are mixed with irrelevant daily conversation or off-topic remarks, and *Uninformative Response*, where the user replies with evasive answers such as "I don't know" or "Don't ask me." As shown in Table 4, GRIL substantially improves resilience in both settings. Under noisy feedback, Qwen2.5-1.5B improves from 12.8% to 47.2%, indicating learned ability to filter distractions and extract relevant information. Under uninformative responses, performance rises from 62.5% to 86.7%, demonstrating graceful termination instead of hallucination when clarification is unavailable. These results confirm that GRIL's benefits extend beyond idealized settings.

7 Conclusion

In this paper, we identify ungrounded reasoning as a critical failure mode where language models fabricate missing premises and proceed with structurally invalid inference rather than recognizing inferential boundaries. We propose GRIL, a multi-turn reinforcement learning framework that explicitly trains models to distinguish between situations requiring reasoning versus clarification. Through stage-specific rewards that penalize delayed detection and encourage effective information integration, GRIL teaches models when to pause and when to proceed. Extensive experiments demonstrate substantial improvements in both premise detection and task success, with gains generalizing to out-of-distribution domains and noisy interactions.

8 Limitations

The current implementation relies on structured output formats to parse actions, and efficiency is evaluated mainly via turn/length proxies rather than directly measuring clarification quality and user burden. Future work should move beyond one-shot, information-complete problem solving toward realistic multi-turn conversations where users reveal constraints gradually, requiring models to proactively elicit missing details while handling refusals, uncertainty, and inconsistency. For preference-sensitive open-ended tasks, "missing premises" are often subjective goals and trade-offs rather than a single correct constraint, calling for evaluations that directly measure clarification utility and user burden, and training frameworks robust to noisy, ambiguous, preference-driven feedback.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 62506332), National Natural Science Foundation of China (No. 62436007), CCF-Tencent Rhino-Bird Open Research Fund, and ZJU Kunpeng&Ascend Center of Excellence.

References

Simran Arora, Patrick S. H. Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. 2023. Reasoning over public and private data in retrieval-based systems. *Trans. Assoc. Comput. Linguistics*, 11:902–921.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *CoRR*, abs/2504.06514.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *CoRR*, abs/2508.15260.

Yujian Gan, Changling Li, Jinxia Xie, Luou Wen, Matthew Purver, and Massimo Poesio. 2024. Clarq-llm: A benchmark for models clarifying and requesting information in task-oriented dialog. *arXiv preprint arXiv:2409.06097*.

Zhenyu Gao. 2025. Modeling reasoning as markov decision processes: A theoretical investigation into nlp transformer models.

Qianyue Hao, Sibo Li, Jian Yuan, and Yong Li. 2025. Rl of thoughts: Navigating llm reasoning with inference-time reinforcement learning. *arXiv preprint arXiv:2505.14140*.

Xiang Hu, Zujie Wen, Yafang Wang, Xiaolong Li, and Gerard de Melo. 2020. Interactive question clarification in dialogue via reinforcement learning. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020 - Industry Track, Online, December 12, 2020*, pages 78–89. International Committee on Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2):42:1–42:55.

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

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *CoRR*, abs/2505.06120.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

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

Yuxin Liu, Chaojie Gu, Yihang Zhang, Bin Qian, and Shibo He. 2025. Reverse thinking enhances missing information detection in large language models. *Preprint*, arXiv:2512.10273.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *CoRR*, abs/2406.06592.

Sichun Luo, Yi Huang, Mukai Li, Shichang Meng, Fengyuan Liu, Zefa Hu, Junlan Feng, and Qi Liu. 2025. Clarifymt-bench: Benchmarking and improving multi-turn clarification for conversational large language models. *Preprint*, arXiv:2512.21120.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Mathieu Rita, Florian Strub, Rahma Chaabouni, Paul Michel, Emmanuel Dupoux, and Olivier Pietquin. 2024. Countering reward over-optimization in llm with demonstration-guided reinforcement learning. *arXiv preprint arXiv:2404.19409*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

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

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, and 1 others. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*.

Shuaijie She, Junxiao Liu, Yifeng Liu, Jiajun Chen, Xin Huang, and Shujian Huang. 2025. R-PRM: reasoning-driven process reward modeling. *CoRR*, abs/2503.21295.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yanbo Wang, Yongcan Yu, Jian Liang, and Ran He. 2025a. A comprehensive survey on trustworthiness in reasoning with large language models. *arXiv preprint arXiv:2509.03871*.

Zekun Wang, Ge Zhang, Kexin Yang, Ning Shi, Wangchunshu Zhou, Shaochun Hao, Guangzheng Xiong, Yizhi Li, Mong Yuan Sim, Xiuying Chen, and 1 others. 2023b. Interactive natural language processing. *arXiv preprint arXiv:2305.13246*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025b. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can llms express their uncertainty. *An Empirical Evaluation of Confidence Elicitation in LLMs. arXiv*.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Xin Xu, Mengdi Zhang, Jian Shao, and Yueting Zhuang. 2025. Mathfimer: Enhancing mathematical reasoning by expanding reasoning steps through fill-in-the-middle task. *Preprint*, arXiv:2502.11684.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Michael JQ Zhang, W Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788*.

Xuan Zhang, Yongliang Shen, Zhe Zheng, Linjuan Wu, Wenqi Zhang, Yuchen Yan, Qiuying Peng, Jun Wang, and Weiming Lu. 2025. Asktoact: Enhancing llms tool use via self-correcting clarification. *CoRR*, abs/2503.01940.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. Archer: Training language model agents via hierarchical multi-turn RL. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

A Appendix

A.1 Data construction

Here give the algorithm to generate training data and test data. This algorithm constructs supervision for *missing-information detection* via **premise masking**. For each problem $Q \in \mathcal{D}$, it segments the text into sentences $S = \{s_1, \ldots, s_n\}$ and filters candidate premise sentences $S_{\text{num}} = \{\, s \in S \mid \texttt{contains\_digit}(s) \land \neg\texttt{is\_question}(s) \,\}$. It then samples a sentence $s_{\text{mask}} \sim \text{Unif}(S_{\text{num}})$, removes it to form $Q_{\text{masked}} = S \setminus \{s_{\text{mask}}\}$, and queries a pretrained LLM to predict whether Q_{masked} is `Solvable` or `Unsolvable`. If Q_{masked} is `Unsolvable`, s_{mask} is labeled *essential* and the pair $(Q_{\text{masked}}, s_{\text{mask}})$ is added as a missing-premise example; otherwise s_{mask} is labeled *redundant*. The output is a labeled dataset of *essential* vs. *redundant* premises.

A.2 Param ablation

A.2.1 K turns ablation

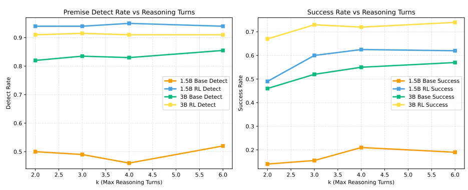

Figure 6: Performance changes via deifferent iteration turns

We evaluate premise detection and success rate under varying reasoning turn limits k. The base model shows unstable performance at small k and improves gradually as k increases. In contrast, GRIL consistently improves robustness and performance across all k. Smaller RL-trained models achieve high premise detection even with limited

Algorithm 1 Missing Information Detection via Premise Masking

Input: Mathematical dataset $\mathcal{D}$; sentence tokenizer; pretrained LLM
Output: Labeled dataset of essential and redundant premises
foreach $Q \in \mathcal{D}$ **do**
  Segment Q into sentences $S = \{s_1, \ldots, s_n\}$
  $S_{\text{num}} \leftarrow \{ s \in S \mid \text{contains\_digit}(s) \wedge \neg\text{is\_question}(s) \}$ **if** $S_{num} = \emptyset$ **then**
    continue
  end
  Sample $s_{\text{mask}} \sim \text{Unif}(S_{\text{num}})$ $Q_{\text{masked}} \leftarrow S \setminus \{s_{\text{mask}}\}$ $V \leftarrow \text{LLM}(\texttt{Solvable?}, Q_{\text{masked}})$ **if** $V = \textit{Unsolvable}$ **then**
    Label s_{mask} as *essential* $\mathcal{D}_{\text{neg}} \leftarrow \mathcal{D}_{\text{neg}} \cup \{(Q_{\text{masked}}, s_{\text{mask}})\}$
  end
  else
    Label s_{mask} as *redundant*
  end
end
return Labeled dataset

turns, while larger models reach near-saturated performance within a few turns, indicating that RL promotes early detection and efficient reasoning. As shown in Figure 6, we evaluate premise detection and success rate under varying reasoning turn limits k. The base model shows unstable performance at small k, whereas GRIL consistently improves robustness and achieves near-saturated performance with fewer turns.

A.2.2 Reward weight analysis

Table 5: Sensitivity analysis of stage weight combinations (k_1, k_2) in Multi-turn RL.

(k_1, k_2)	Success Rate	Premise Detect
$(0.2, 0.8)$	0.624	0.918
$(0.3, 0.7)$	0.618	0.908
$(0.4, 0.6)$	0.627	0.931

We conduct a parameter sensitivity study to examine the impact of stage weight combinations (k_1, k_2) in GRIL As shown in Table 5, the Success Rate remains largely stable as k_1 increases from 0.2 to 0.4. In contrast, Premise Detect shows a slight improvement when a larger weight is assigned to the first stage, indicating that emphasizing the early stage facilitates earlier detection of missing premises without degrading overall problem-solving performance. These results suggest that GRIL is robust to stage weight variations and maintains effective reasoning and premise awareness across different configurations.

A.3 Time decay analysis

To examine the effect of the decay factor in the first-stage policy, we compare $\gamma = 0.5$ and $\gamma = 1.0$. As shown in Table 6, a stronger decay ($\gamma = 0.5$) significantly improves Premise Detect, indicating higher sensitivity to missing premises and a stronger tendency to initiate clarification early. However, this

γ	Success Rate	Premise Detect	Response Length
0.5	0.617	0.936	472
1.0	0.659	0.894	838

Table 6: Effect of the decay factor γ in the first-stage policy.

aggressive querying behavior also leads to a slight drop in Success Rate, suggesting a more conservative strategy that may prematurely abandon complex problems. In addition, the stronger decay results in shorter average responses, while the non-decayed setting ($\gamma = 1.0$) produces longer responses due to more extended reasoning trajectories.

A.3.1 Data ratio analysis

Ratio	Success Rate	Premise Detect
3 : 7	0.650	0.873
4 : 6	0.601	0.973
5 : 5	0.622	0.955

Table 7: Effect of data ratio between incomplete-premise and complete queries.

We study the impact of different data ratios between incomplete-premise queries and complete queries. As shown in Table 7, the model achieves competitive Success Rate and Premise Detect even with a smaller proportion of incomplete-premise data (e.g., 3:7). As the ratio increases to 4:6 or 5:5, Premise Detect improves, while Success Rate remains relatively stable. These results indicate that GRIL is robust to the amount of incomplete-premise data and can effectively learn premise detection and problem solving even when such data is limited.

A.4 Baseline and Test Data

A.4.1 Baselines

We design three categories of baseline models to systematically analyze the roles of prompting, supervised fine-tuning (SFT), and reinforcement learning (RL) in handling missing-premise scenarios.

Base Model. The base model refers to a pre-trained language model without any task-specific training. It is evaluated on all benchmarks using a standard zero-shot reasoning prompt, without explicitly informing the model that the input question may contain missing premises. This baseline measures the model's intrinsic ability to handle incomplete information relying solely on knowledge acquired during pretraining.

Prompt-based Model. The prompt-based model shares the same pretrained backbone as the base model, but augments the input with an explicit task instruction. Specifically, the prompt informs the model that the problem may contain missing or incomplete premises and requires the model to first assess solvability before attempting to produce a solution. No parameter updates are performed for this model, allowing us to isolate and evaluate the performance gains introduced purely by explicit task prompting.

Supervised Fine-Tuned (SFT) Model. The SFT model is trained on the same data source as the reinforcement learning stage, but optimized using supervised learning. We construct high-quality, multi-turn missing-premise dialogue data using GPT-4o-mini. Each dialogue trajectory typically consists of: (1) an initial question with insufficient information, (2) intermediate premise detection or clarification steps, and (3) a final outcome, such as requesting additional information or determining that the problem is unsolvable under the current conditions. The model is fine-tuned by imitating these trajectories, serving as a strong supervised baseline aligned with the RL model in terms of data distribution.Example of training loss are shown in Figure 7

A.4.2 Evaluation Datasets

We evaluate all models on a diverse set of benchmark datasets, including both our constructed missing-premise datasets and publicly available benchmarks used in prior work.

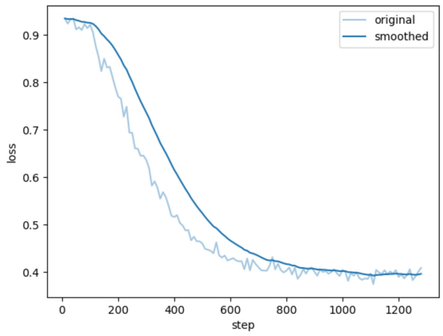

Figure 7: Example loss plot of SFT Methods

GSM8K-Insufficient. GSM8K_Insufficient is constructed from the GSM8K test set by systematically removing essential premises from the original problems, rendering them unsolvable without additional information. This dataset is designed to evaluate a model's ability to detect missing premises and avoid overconfident reasoning under incomplete inputs.

MetaMath-Insufficient. MetaMath_Insufficient is constructed using the same procedure as GSM8K_Insufficient, but based on the MetaMath dataset. Compared to GSM8K, this dataset contains more diverse mathematical structures and longer reasoning chains, posing greater challenges for missing-premise detection in complex scenarios.

Public Missing-Premise Benchmarks. In addition, we evaluate our models on publicly available benchmarks from prior work, including GSM8K, SVAMP, and Formula. These datasets contain manually or semi-automatically constructed missing-premise problems and have been widely used to assess model robustness under incomplete inputs. Evaluating on these benchmarks ensures the comparability of our approach with existing methods.

A.5 System prompt example

We provide system prompt example for better detail, just displayed in Table 8, all prompt stays the same during traning and test.

A.6 Detailed Datasets Info Used in Analysis

This appendix provides a detailed description of the datasets used in §5, including their sources, construction procedures, and sizes.

Main experiments. We evaluate on two premise-missing benchmarks. GSM8K-Insufficient is constructed from the GSM8K standard test set by

System Prompt: You are a helpful assistant.
User Prompt: Please solve the following problem, strictly adhering to this format:
1. **<think>...</think>**: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process.
2. **<answer>...</answer>**: Must ONLY contain the final answer enclosed in ‘
boxed‘. If information is insufficient, the answer must be ’insufficient information’.

Input Format:

```
Turn X:
State:
{problem input}
```

Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format.

Table 8: Prompt Example

removing essential premises, resulting in 865 premise-missing instances. MetaMath-Insufficient is constructed from MetaMath in the same manner, resulting in 1130 premise-missing instances.

Analysis experiment 1. The data used in Analysis Experiment 1 is taken from the publicly available dataset released in Ref. (Fan et al., 2025).

Analysis experiment 2. Analysis Experiment 2 uses standard information-complete test sets, including the GSM8K standard test set and the MATH500 test set.

Analysis experiment 3. Analysis Experiment 3 uses a mixed dataset containing both complete problems and premise-missing problems, with a total of 3814 instances.

Analysis experiment 4. Analysis Experiment 4 evaluates multi-domain QA transfer on premise-missing variants. HotpotQA-Insufficient is constructed from HotpotQA, containing 1055 instances. CQA-Insufficient is constructed from CommonsenseQA, containing 1170 instances.

A.7 Llama-3B-Instruct training

As shown in Table 9 on Llama3-3B-Instruct (Dubey et al., 2024), prompting and multi-turn RL consistently outperform the base model on both missing-premise benchmarks, with GRIL achieving the best overall performance.

	GSM8K-Insufficient		MetaMath-Insufficient	
Model	SR	PD	SR	PD
Base Model	17.0	65.0	15.0	64.0
w/ Prompt	43.8	80.8	42.9	73.6
w/ Multi-turn RL	49.3	80.4	44.2	79.6

Table 9: Performance on GSM8K-Insufficient and MetaMath-Insufficient

B The Use of Large Language Model(LLMs)

To elevate the overall quality of this paper, a large language model was utilized to refine the manuscript—specifically, to enhance its clarity, streamline its conciseness, and ensure strict grammatical correctness.

C Algorithm pipeline

Algorithm 2 GRIL Training Framework

Input : Dataset $\mathcal{D} = \mathcal{D}_{inc} \cup \mathcal{D}_{comp}$, Initial Policy π_θ, Reference Policy π_{ref}, Max turns K, Hyperparameters $\gamma_d, \alpha, \beta_{\text{KL}}, \lambda$

for *iteration* $= 1, \ldots, I$ **do**
  Sample batch of queries $Q \sim \mathcal{D}$ **foreach** *query* $q \in Q$ **do**
    Initialize state $s_0 \leftarrow (q)$, turn $t \leftarrow 0$
    ▷ `Stage 1: Clarify and Pause`
    while $t < K$ ***and*** *not terminated* **do**
      Generate inferential action $a_t \sim \pi_\theta(\cdot \mid s_t)$
      if $q \in \mathcal{D}_{inc}$ **then**
        if $a_t = a_{clarify}$ **then**
          Obtain missing premise p_q from Environment $R_{\text{detect}} \leftarrow \gamma_d^t \cdot r_{\text{base}}$ $s_{t+1} \leftarrow s_t \oplus a_t \oplus p_q$
          ▷ `Stage 2: Grounded Reasoning`
          Generate solution $a_{\text{solve}} \sim \pi_\theta(\cdot \mid s_{t+1})$ $R_{\text{solve}} \leftarrow r_{\text{correct}} \cdot \mathbb{I}[a_{\text{solve}} \text{ is correct}]$ Terminate episode
        else
          Receive negative feedback [neg.] from Environment $s_{t+1} \leftarrow s_t \oplus a_t \oplus [\text{neg.}]$
        end
      end
      else if $q \in \mathcal{D}_{comp}$ **then**
        if $a_t = a_{solve}$ **then**
          $R_{\text{comp}} \leftarrow r_{\text{correct}} \cdot \mathbb{I}[a_t \text{ is correct}]$ Terminate episode
        else
          Receive unnecessary feedback [unc.] from Environment Apply penalty $R_{\text{comp}} \leftarrow -\lambda \cdot \mathbb{I}[\text{unc.}]$ $s_{t+1} \leftarrow s_t \oplus a_t \oplus [\text{unc.}]$
        end
      end
      $t \leftarrow t + 1$
    end
    ▷ `Reward Assignment`
    Compute trajectory reward $R(q, \theta)$ by combining R_{detect}, R_{solve}, and R_{comp} (Eq. 7)
  end
  Update policy π_θ by maximizing the PPO objective with KL penalty $-\beta_{\text{KL}} D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}})$
end

D Case Study

D.1 Case Study 1

Case Study 1 (Before Training): the model fails to proactively stop under insufficient information.

System Prompt: You are a helpful assistant.

User Prompt: Please solve the following problem, strictly adhering to this format: 1. <think>...</think>: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process. 2. <answer>...</answer>: Must ONLY contain the final answer. If information is insufficient, the answer must be insufficient information. Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format.

Interaction Log:

```
system
You're a helpful assistant.
user
Please solve the following problem, strictly adhering to this format:
1. **<think>...</think>**: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process.
2. **<answer>...</answer>**: Must ONLY contain the final answer . If information is insufficient, the answer must be 'insufficient information'.

Turn 1:
State:
Find the sum of the remainders when the same integer is divided by 3 and by 5.
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format.

assistant
<think></think><answer>To solve the problem of finding the sum of the remainders when the same integer is divided by 3 and by 5, we need to understand the concept of modular arithmetic.

When an integer \( n \) is divided by 3, the remainder can be any of the following: 0, 1, or 2. Similarly, when \( n \) is divided by 5, the remainder can be any of the following: 0, 1, 2, 3, or 4.

The sum of the remainders when \( n \) is divided by 3 and by 5 will be the sum of all possible remainders for each divisor. Since there are 3 possible remainders for each divisor, we can calculate the total sum as follows:

Sum = (Remainder when divided by 3) + (Remainder when divided by 5)

There are 3 remainders when divided by 3 (0, 1, 2) and 5 remainders when divided by 5 (0, 1, 2, 3, 4). Therefore, the total sum of the remainders is:

Sum = 0 + 1 + 2 + 0 + 1 + 2 + 3 + 4

Sum = 15

So, the sum of the remainders when the same integer is divided by 3 and by 5 is 15.

Final answer: \(\boxed{15}\)</answer>
user
Reward:
0.0

Turn 2:
State:
That is incorrect. Please try again.
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format.

assistant
<think>It appears there was a misunderstanding. Let me clarify the problem and provide the correct solution.</think><answer>When trying to find the sum of the remainders when the same integer is divided by 3 and by 5, we need to consider the possible remainders for each divisor. As previously stated, the remainders when divided by 3 are 0, 1, and 2, and the remainders when divided by 5 are 0, 1, 2, 3, and 4. The sum of these remainders is 15. Therefore, the correct answer is 15.</answer>
user
Reward:
0.0

Turn 3:
State:
That is incorrect. Please try again.
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text. Strictly follow this format.

assistant
```

```
<think>Let me rephrase the question and provide the correct solution.</think><answer>When a
    number is divided by 3, the possible remainders are 0, 1, and 2. When the same number is
    divided by 5, the possible remainders are 0, 1, 2, 3, and 4. The sum of these remainders is
    15. Therefore, the correct answer is 15.</answer>
user
Reward:
0.0

Turn 4:
State:
That is incorrect. Please try again.
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra
    text. Strictly follow this format.

assistant
<think>Let me restate the problem clearly and provide the correct solution.</think><answer>The
    sum of the remainders when a number is divided by 3 and 5 is 15. This is because the
    possible remainders when divided by 3 are 0, 1, and 2, and when divided by 5, they are 0, 1,
     2, 3, and 4. Adding these together gives 15.</answer>
```

Observation. The problem is underdetermined since the integer is unspecified. However, the model repeatedly attempts to solve the problem instead of detecting the missing premise and stopping or requesting clarification.

Case Study 1 (After Training): The model proactively stops, requests missing premises, and solves after clarification.

System Prompt:
You're a helpful assistant.

User Prompt:
Please solve the following problem, strictly adhering to this format:

- <think>...</think>: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process.
- <answer>...</answer>: Must ONLY contain the final answer . If information is insufficient, the answer must be `insufficient information`.

Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra text.

Interaction Log:

```
Turn 1:
State:
Find the sum of the remainders when the same integer is divided by 3 and by 5.
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra
    text. Strictly follow this format.

assistant
<think>Let's break down the problem step by step. We need to find the sum of the remainders when
     the same integer is divided by 3 and by 5. However, without knowing the specific value of
    the integer, it is impossible to determine the remainders. Therefore, the problem does not
    provide enough information to determine the remainders.</think><answer>Insufficient
    information</answer>

user
Reward:
1.0

Turn 2:
State:
```

```
Successfully detected a missing premise. Here is the missed information: When an integer is
    divided by 15, the remainder is 7.. Please solve the problem now.You should give detailed
    reasonig steps
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no extra
    text. Strictly follow this format.

assistant
<think>Let's break down the problem step by step. We know that the remainder when an integer is
    divided by 15 is 7. Therefore, the integer can be written in the form \( N = 15k + 7 \) for
    some integer \( k \). We need to find the remainders when \( N \) is divided by 3 and by 5.
    The remainder when \( N \) is divided by 3 is \( 7 \mod 3 = 1 \). The remainder when \( N \)
     is divided by 5 is \( 7 \mod 5 = 2 \). Therefore, the sum of the remainders is \( 1 + 2 = 3
     \). Therefore, the sum of the remainders is 3.</think><answer>3</answer>
```

Observation. After training, the model correctly identifies the underdetermined input and stops with an `insufficient information` response. Once the missing premise is provided, it successfully integrates the new condition and solves the problem in the next turn.

D.2 Case Study 2

Case Study 2 (Before Training): The model repeatedly attempts to solve an ill-defined problem without identifying the missing premise.

System Prompt:
You're a helpful assistant.

User Prompt:
Please solve the following problem, strictly adhering to this format:

- `<think>...</think>`: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process.
- `<answer>...</answer>`: Must ONLY contain the final answer. If information is insufficient, the answer must be `insufficient information`.

Always output: `<think> [Your thoughts] </think> <answer> [your answer] </answer>` with no extra text.

Interaction Log:

```
Turn 1:
State:
Barbara got a great deal on a new chest of drawers, but she has to take a lot of paper
out of the drawers to be able to use it. If a bunch holds 4 sheets of paper, a bundle
holds 2 sheets of paper, and a heap holds 20 sheets of paper, how many sheets of paper
did Barbara remove from the chest of drawers? If we know the answer to the above
question is 114, what is the value of unknown variable x?

assistant
<think>To solve this problem, we need to determine the value of the unknown variable x,
which represents the number of drawers in the chest of drawers.

From the given information:
- A bunch holds 4 sheets of paper.
- A bundle holds 2 sheets of paper.
- A heap holds 20 sheets of paper.

We also know that Barbara removed a total of 114 sheets of paper.

We can set up an equation:
4x + 2x + 20x = 114

Now, we can solve this equation.</think>
<answer>x = 5</answer>
```

```
Reward:
0.0

Turn 2:
State:
That is incorrect. Please try again.

assistant
<think>I apologize for the mistake. Let's re-examine the problem.

We can set up the equation:
4x + 2x + 20x = 114</think>
<answer>5</answer>

Reward:
0.0

Turn 3:
State:
That is incorrect. Please try again.

assistant
<think>I apologize for the mistake. Let's re-examine the problem.

We can set up the equation:
4x + 2x + 20x = 114</think>
<answer>5</answer>

Reward:
0.0

Turn 4:
State:
That is incorrect. Please try again.

assistant
<think>I apologize for the mistake. Let's re-examine the problem.

We can set up the equation:
4x + 2x + 20x = 114</think>
<answer>5</answer>
```

Observation. Despite repeated negative feedback, the model fails to identify the missing or ill-defined premise regarding the variable x and persistently attempts to solve the problem using an incorrect formulation, illustrating overconfident reasoning and the absence of a proactive stopping strategy.

Case Study 2 (After Training): The model proactively detects missing premises and successfully solves the task after clarification.

System Prompt:
You're a helpful assistant.

User Prompt:
Please solve the following problem, strictly adhering to this format: 1. `<think>...</think>`: Must contain the complete, step-by-step reasoning process. If information is missing, state why here and terminate your thinking process. 2. `<answer>...</answer>`: Must ONLY contain the final answer. If information is insufficient, the answer must be `insufficient information`.
Always output: `<think> [Your thoughts] </think> <answer> [your answer] </answer>` with no extra text. Strictly follow this format.

Interaction Log:

```
system
You're a helpful assistant.
user
Please solve the following problem, strictly adhering to this format:
1. **<think>...</think>**: Must contain the complete, step-by-step reasoning process. If
    information is missing, state why here and terminate your thinking process.
2. **<answer>...</answer>**: Must ONLY contain the final answer . If information is insufficient,
     the answer must be 'insufficient information'.

Turn 1:
State:
Barbara got a great deal on a new chest of drawers, but she has to take a lot of paper out
of the drawers to be able to use it. If a bunch holds 4 sheets of paper, a bundle holds 2
sheets of paper, and a heap holds 20 sheets of paper, how many sheets of paper did Barbara
remove from the chest of drawers? If we know the answer to the above question is 114, what
is the value of unknown variable x?
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no
extra text. Strictly follow this format.

assistant
<think>Let's break down the total number of sheets of paper in the chest of drawers. We know
the total number of sheets removed, which is 114. We need to determine the total number of
sheets of paper in the chest of drawers. Since each bunch holds 4 sheets, each bundle holds
2 sheets, and each heap holds 20 sheets, we can express the total number of sheets as the sum
of the number of bunches, bundles, and heaps. Since the problem does not provide the number of
bunches, bundles, and heaps, we cannot determine the exact value of the unknown variable x.
Therefore, the problem does not provide enough information to determine the value of the unknown
variable x.</think><answer>Insufficient information</answer>

user
Reward:
1.0

Turn 2:
State:
Successfully detected a missing premise. Here is the missed information: She found x bundles
of colored paper, 2 bunches of white paper, and 5 heaps of scrap paper.. Please solve the
problem now.You should give detailed reasonig steps
Always output: <think> [Your thoughts] </think> <answer> [your answer] </answer> with no
extra text. Strictly follow this format.

assistant
<think>Let's break down the total number of sheets of paper in the chest of drawers. We know
the total number of sheets removed, which is 114. We need to determine the total number of
sheets of paper in the chest of drawers. The total number of bundles is x, each bundle holds
2 sheets, so the total number of sheets in bundles is 2x. The total number of bunches is 2,
each bunch holds 4 sheets, so the total number of sheets in bunches is 2 * 4 = 8. The total
number of heaps is 5, each heap holds 20 sheets, so the total number of sheets in heaps is
5 * 20 = 100. The total number of sheets is 2x + 8 + 100 = 2x + 108. The total number of
sheets removed is 114. Therefore, 2x + 108 = 114. Solving for x, we get 2x = 114 - 108 = 6.
Therefore, x = 6 / 2 = 3.</think><answer>3</answer>
```

Observation. After training, the model correctly stops when the initial question is under-specified and requests missing premises. Once the additional information is provided, it integrates the new constraints and solves for the unknown variable in the subsequent turn.

D.3 Case of GapRatio

Okay, so Lani baked 55 cookies, and the question is asking how many cookies were in each jar. Hmm, let me think about this. First off, I need to figure out what information is given and what's missing. We know that Lani baked 55 cookies, but we don't know anything else about the jars. Maybe she put them in jars, but how many jars? Was each jar filled equally? Is there a specific number of jars mentioned?I don't think so. The problem doesn't specify how many jars there are or if the cookies were divided equally among them. Without that information, it's hard to determine how many cookies are in each jar. If there were, say, 5 jars, then each jar would have 11 cookies. But since we don't know the number of jars, we can't make that assumption.Maybe the question expects me to assume something, like maybe it's divided into jars with the same number of cookies, but even then, without knowing the total number of jars, I can't calculate the exact number per jar. It's like trying to split something without knowing how many parts it's being split into.Wait, maybe the question is trying to test if I recognize that information is missing? So, maybe the answer is that there's insufficient information to determine the number of cookies per jar. Because without knowing how many jars there are, we can't figure out how the cookies are distributed.Yeah, that makes sense. So, the problem is underspecified because it doesn't provide enough details about the number of jars or any other factors that could help determine the distribution. Without that, it's impossible to know how many cookies are in each jar.So, I think the answer is that there's not enough information provided to determine the number of cookies in each jar.</think>The problem states that Lani baked 55 cookies but does not specify the number of jars or any other details about how the cookies were distributed. Without this information, we cannot determine how many cookies were in each jar.

Observation. By the third sentence, the model has already begun to suspect that we do not know any information about the jars. However, after recognizing this issue, the model does not choose to stop; instead, it repeatedly engages in doubt and speculation, for example by assuming that there are five jars. This example illustrates the evidence-free reasoning exhibited by an untrained model when confronted with missing-premise problems, which in turn leads to a high GapRatio.