

Instruct-of-Reflection: Enhancing Large Language Models Iterative Reflection Capabilities via Dynamic-Meta Instruction

Liping Liu^{1*}, Chunhong Zhang¹, Likang Wu³, Chuang Zhao⁴,
Zheng Hu^{1†}, Ming He^{2‡}, Jianping Fan²

^{1‡}Beijing University of Posts and Telecommunications

²AI Lab of Lenovo Research ³Tianjin University

⁴The Hong Kong University of Science and Technology

{liuliping, Huzheng, zhangch}@bupt.edu.cn, heming01@foxmail.com

wulk@tju.edu.cn, czhaobo@connect.ust.hk, jfan1@lenovo.com

Abstract

Self-reflection for Large Language Models (LLMs) has gained significant attention. Existing approaches involve models iterating and improving their previous responses based on LLMs' internal reflection ability or external feedback. However, recent research has raised doubts about whether intrinsic self-correction without external feedback may even degrade performance. Based on our empirical evidence, we find that current static reflection methods may lead to redundant, drift, and stubborn issues. To mitigate this, we introduce **Instruct-of-Reflection (IoRT)**, a novel and general reflection framework that leverages dynamic-meta instruction to enhance the iterative reflection capability of LLMs. Specifically, we propose the instructor driven by the meta-thoughts and self-consistency classifier, generates various instructions, including refresh, stop, and select, to guide the next reflection iteration. Our experiments demonstrate that IoRT achieves an average improvement of 10.1% over established baselines in mathematical and commonsense reasoning tasks, highlighting its efficacy and applicability. Our code is available at <https://github.com/llp635/IoRT>.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Achiam et al., 2023; Anil et al., 2023; Claude, 2024) have demonstrated remarkable capabilities across various natural language processing tasks (Bai et al., 2022a; Wei et al., 2022a; Chu et al., 2023), particularly when leveraging a range of prompting strategies such as Chain-of-Thought (Wei et al., 2022b) which improve the reasoning

*Work done while Liping Liu was an intern at AI Lab of Lenovo Research.

†Corresponding authors.

‡The first affiliation is State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China.

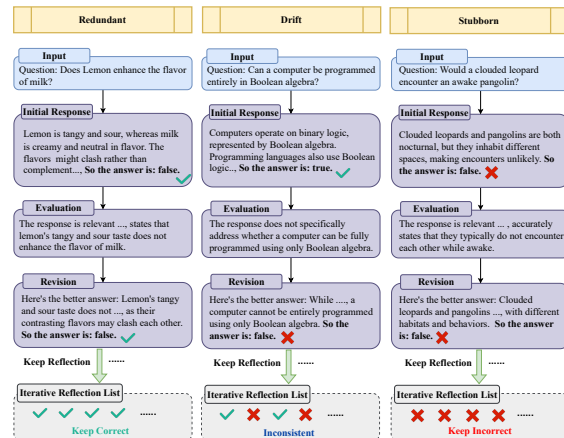


Figure 1: Prior reflection research is primarily based on static iterative strategies, which face issues of redundancy, drift, and stubborn.

ability of LLMs without increasing model parameters (Rae et al., 2021; Srivastava et al., 2023; Yin et al., 2023). Inspired by how humans iteratively refine their written text (Madaan et al., 2024), the concept of "self-correction" (Welleck et al., 2023) has been proposed and garnered extensive attention where LLMs first generate an initial response, then gather feedback to refine previous responses (Ganguli et al., 2023; Xi et al., 2023; Paul et al., 2024a). However, recent studies (Huang et al., 2024; Zhang et al., 2024) raise two doubts about the self-correction capabilities of LLMs: First, (Kim et al., 2024a; Shinn et al., 2023) use oracle labels regarding the answer correctness to guide the self-correction process. However, in practice, the availability of oracle labels seems counter-intuitive because there seems to be little reason to deploy LLMs for problem solving if the ground truth is already known. Second, without any external or human feedback, the performance after LLMs' intrinsic self-correction even deteriorates.

Our research also focuses on the reflective capabilities of LLMs. Unlike previous studies that

primarily investigate performance changes in intrinsic reflection, we explore whether the performance of LLMs in reflection deteriorates when external feedback is available. Therefore, in our exploration experiments, we evaluate the performance of CRITIC (Gou et al., 2024), which leverages the results of tool interactions as external feedback, enabling a systematic investigation of how this feedback impacts the performance of reflection. Our findings (§ 3.1) indicate that neither intrinsic self-correction nor CRITIC performs satisfactorily. To investigate the reasons behind that, we comprehensively analyze the performance across multiple iterations of reflection, which is different from prior research focusing on a single reflection. Figure 1 shows that the i th reflection is derived from the $(i - 1)$ th response, this static iterative reflection leads to three issues: 1) **Redundant** ($\checkmark \Rightarrow \checkmark$): Although it does not alter accuracy in iterations, it does increase overhead. 2) **Drift** ($\checkmark \Rightarrow \times$): Self-correction is highly random, undermining both accuracy and reliability. 3) **Stubborn** ($\times \Rightarrow \times$): LLMs consistently persist in errors, failing to properly identify and correct potential mistakes. Additionally, we found that the prevalence of these three issues varies across different LLMs and tasks.

As a remedy to these challenges, we propose Instruct of Reflection (IoRT), an efficient and dynamic iterative reflection framework designed to continuously optimize the reflection process. The framework including three stages: (i) meta-thinker generates meta-thoughts, (ii) refreshing to generate an initial response, and reflector to self-reflect based on feedback. (iii) instruct the iteration. Notably, aiming to mitigate redundant, drift, and stubborn issues, in the third stage, the instructor, augmented by meta-thought and driven by the self-consistency classifier, generates refresh, stop, or select instructions to direct the next reflection iteration. Specifically, when the Instructor determines that both responses have correctly addressed the problem, it generates a stop instruction to promptly halt the reflection process, effectively reducing redundancy. If the instructor assesses that both answers are incorrect, it generates a refresh instruction, helping the LLMs prevent stubbornness. In cases where answers are inconsistent, the instructor selects the more optimal response as the output of the current iteration and the foundation for the next reflection, thereby reducing drift and improving performance accuracy.

We evaluate the performance of IoRT across var-

ious LLMs and two distinct tasks: mathematical reasoning and commonsense reasoning. The result demonstrates that IoRT outperforms baselines on established strong baselines, underscoring the critical role of the instructor in augmenting the iterative reflection capabilities of LLMs. Our main contributions can be summarized as follows:

- Our exploration experiments reveal that LLMs struggle to correct previous errors. The static iterative reflection suffers from redundant, drift, and stubborn, which may increase overhead or even deteriorate performance.
- We propose Instruct-of-Reflection (IoRT), a dynamic iterative reflection framework that integrates abstract reasoning into the reflection process, adaptively generating different instruction to regulate the reflection process.
- Experimental results on three datasets and five different LLMs underscore the efficacy, overhead efficiency, and generality of IoRT. Additionally, without relying on any oracle labels, IoRT demonstrates its practicality.

2 Related Works

2.1 Prompt-based Reasoning

To exploit the reasoning ability in LLMs, (Wei et al., 2022b) proposes Chain-of-Thought (CoT) prompting which guides the model to generate a series of text describing reasoning steps before reaching the final answer. Subsequently, (Chen et al., 2023) introduces Program-of-Thoughts (PoT) prompting which uses llms to generate programming language statements. In recent years, the self-correction capabilities of LLMs have garnered significant attention, with LLMs expected to iteratively improve their previous responses based on feedback (Bai et al., 2022b). Existing studies on self-correction can be categorized into two types based on the feedback source: external feedback and internal feedback (Pan et al., 2024). Internal feedback relies on the model’s inherent knowledge and parameters, while external feedback involves inputs from humans, other models (Wang et al., 2023b; Paul et al., 2024b), or external tools and knowledge sources (Chen et al., 2024; Olausson et al., 2023; Gao et al., 2023). We comprehensively evaluate the performance of CRITIC (Gou et al., 2024), which leverages specialized tools, such as code executors, to evaluate and reflect on the external feedback obtained through its interactions

with these tools. However, recent studies (Huang et al., 2024; Stechly et al., 2023; Valmeekam et al., 2023; Zhang et al., 2024) cast doubt on the ability of LLMs to correct previous responses based solely on their inherent reflection capacity. Additionally, research on reflection with external feedback (Kim et al., 2024b; Shinn et al., 2023) have been doubted on using oracle labels which are often unavailable in practice to determine when to stop the self-correction loop during the reflection. The improvements vanish when the reflection no longer relies on oracle labels (Huang et al., 2024). Apart from these studies questioning the use of oracle in self-reflection during the iterative reflection process, we also observe that improvements in (Gou et al., 2024) stem from using oracle labels to correct only the incorrect samples during evaluation. Unlike previous findings, we observe that both intrinsic self-correction and external feedback can lead to a deterioration in performance after reflection. Additionally, our proposed Instruct of Reflection Tool (IoRT) diverges from existing reflection methods by emphasizing continuous performance optimization through diverse instructional guidance. We design the instructor to guide the next reflection iteration without relying on oracle labels.

2.2 Abstract Reasoning

Instead of addressing the problems directly, abstract reasoning relies on higher-level thoughts to guide thinking (Zheng et al., 2024), making it essential for sophisticated cognitive processing in artificial intelligence systems (Lake et al., 2017; Chollet, 2019; Qiu et al., 2024). For example, consider the question "What are the roots of a quadratic equation?" Abstract reasoning involves extracting the fundamental principle of "the quadratic equation root formula" and applying this principle to this specific question. (Zheng et al., 2024) uses the concepts and principles to guide reasoning by taking a step back. (Fu et al., 2024) guides the model to generate hints such as specific knowledge or key ideas. (Xiong et al., 2024) designs a preliminary study to quantify and delve into the abstract reasoning abilities of LLMs. (Yang et al., 2024) adapts by refining or retrieving high-level thought templates to instantiate specific reasoning structures. In our research, we build on (Fu et al., 2024) to generate meta-thoughts, thereby enhancing the abstract reasoning abilities of the reflective instructor. In contrast to the above works, which obtain an initial response through abstract reasoning, we incorpo-

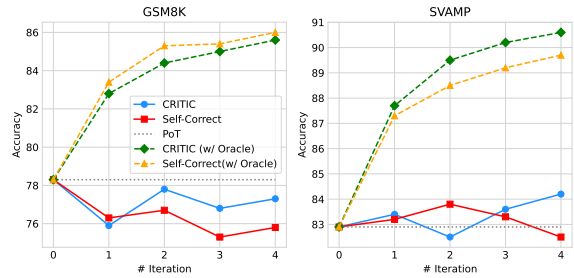


Figure 2: Compare the performance changes of self-correct and CRITIC during iterative reflection across two datasets, evaluating with and without Oracle.

rate abstract reasoning into the reflection process by designing the instructor within our framework.

3 Empirical Evidence for Iterative Reflection

Prior research primarily focuses on performance changes in a single reflection, which fails to fully capture the potential of reflection for long-term improvements. To gain deeper insights into how reflection can gradually refine answers, We comprehensively evaluate the performance changes of self-correct and CRITIC (Gou et al., 2024) across multiple reflection iterations (Huang et al., 2024).

3.1 Performance of Iterative Reflection

We systematically evaluate the performance of self-correct (w/o external feedback) and CRITIC (w/ tool interactions as an available external feedback) across multiple iterative reflections on the GSM8K and SVAMP datasets with GPT-3.5. Additionally, we simulate an evaluation method (w/ oracle) that relies on oracle labels, focusing only on corrections from incorrect to correct answers in each iteration, while disregarding instances where correct answers are modified into incorrect ones.

In Figure 2, evaluating reflection using oracle labels, the accuracy of both self-correct and CRITIC improves steadily with each iteration. In contrast, without oracle labels, the performance is unstable and even deteriorate, regardless of the availability of external feedback. For instance, on GSM8K and SVAMP, the performances of self-correct and CRITIC drop by up to -2.4% and -3.0% , respectively. Notably, compared to self-correct, the performance is more reliable with CRITIC.

3.2 Further Performance Analysis

Why does performance degrade after reflection? To investigate this question, we further conduct the

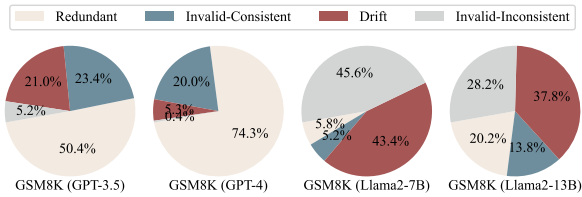


Figure 3: Analyze the different types of iteration of CRITIC on GSM8K.

following analysis:

Step 1: We classify all samples in GSM8K into four categories based on the correctness before and after reflection. Then analyze the performance changes in each iteration using self-correct and CRITIC on GPT-3.5. **Step 2:** We classify all samples into four categories based on the consistency and correctness of the answers throughout all iterations: 1) *Redundant Iteration* means the answers consistently remain correct. 2) *Invalid Consistent Iteration* insists on a consistent wrong answer. 3) *Drift Iteration* generates both incorrect and correct during the iterative reflection. 4) *Invalid Inconsistent Iteration* continuous changes occur in iterations without ever converging to the correct solution. The detailed results are as follows: **Reflection can not only be beneficial but also detrimental.** As shown in Table 5, for both self-correct and CRITIC, the proportion of $\checkmark \Rightarrow \times$ is often comparable to or even exceeds that of $\times \Rightarrow \checkmark$, which explains the performance degradation after reflection. Notably, CRITIC demonstrates a lower ratio of $\checkmark \Rightarrow \times$ compared to self-correct, suggesting that external feedback promotes the stability of the reflection.

Models of different sizes yield varying reflection performance. In Figure 3, significant differences in reflection performance are observed across various LLMs. For instance, GPT-4 shows a 94.3% share of stable iterations, while 89% of LLaMA-2 7B’s iterations are unstable. It implies that larger models like GPT-4, benefiting from vast training data and substantial computational resources (Ouyang et al., 2022; Chowdhery et al., 2023; Chung et al., 2024), exhibit more stable performance compared to smaller models.

Various reflective iterations correspond to different limitations. As shown in Figure 1, LLMs often perform the i _{th} reflection based on the $(i - 1)$ _{th} response, and this static iterative reflection highlights the following three limitations: I. *Redundant Iteration* \Rightarrow *Redundant*: Although re-

dundant iterations do not affect accuracy, they contribute to unnecessary costs and delays. II. *Drift Iteration* \Rightarrow *Drift*: Especially, $\checkmark \Rightarrow \times$ indicates that LLMs often exhibit a high degree of uncertainty, leading to reflection drift and a decline in accuracy. III. *Invalid iteration* \Rightarrow *Stubborn*: LLMs are too stubborn to recognize and correct mistakes. Our method effectively addresses these three issues, achieving optimization of accuracy and efficiency in reflection during iterations.

4 Instruct-of-Reflection (IoRT)

Prior sections illustrate the challenges LLMs encounter in static iterative reflection such as redundancy, drift and stubborn. How to precisely identify effective reflections while mitigating the impact of harmful or redundant reflections? As a remedy, we innovatively propose Instruct-of-Reflection (IoRT), a novel framework that implements dynamic-meta instruction for the iterative reflection process. We can get an overview of the IoRT framework in Figure 4. Specifically, IoRT involves three main steps: generate meta-thoughts, refresh and self-reflect, and instruct the iteration. As the core module, the instructor is augmented by meta-thoughts and driven by self-consistency classifier, generating instructions including refresh, stop, and select, directing the next reflection iteration.

4.1 Generate Meta Thoughts

Human often summarize and abstract higher-level ideas when solving problems (Yang et al., 2024). Inspired by Hint-before-Solving Prompting (HSP) (Fu et al., 2024) and Meta-Reasoning (Zeng et al., 2024), we propose meta-thinker which enables LLMs to explicitly generate meta-thoughts through a few-shot learning for problem solving. Meta-thoughts are high-level knowledge based on abstract reasoning, encompassing analytical methods and fundamental overall strategies. They enhance the instructor’s role as a teacher by providing a comprehensive overview of the reflection process. For instance, for the question "Can a honey bee sting a human more than once?", since the bee’s stinger is closely related to its ability to sting, the meta-thought for this question primarily focuses on the basic principle of the "changes in the stinger". The meta memory module stores meta-thoughts, distilled from various tasks, recorded in the form of $\mathcal{E} = \{(q_i, m_i)\}$, expressing them as a pair relationship $e_i = (q_i, m_i)$, where q_i represents the question

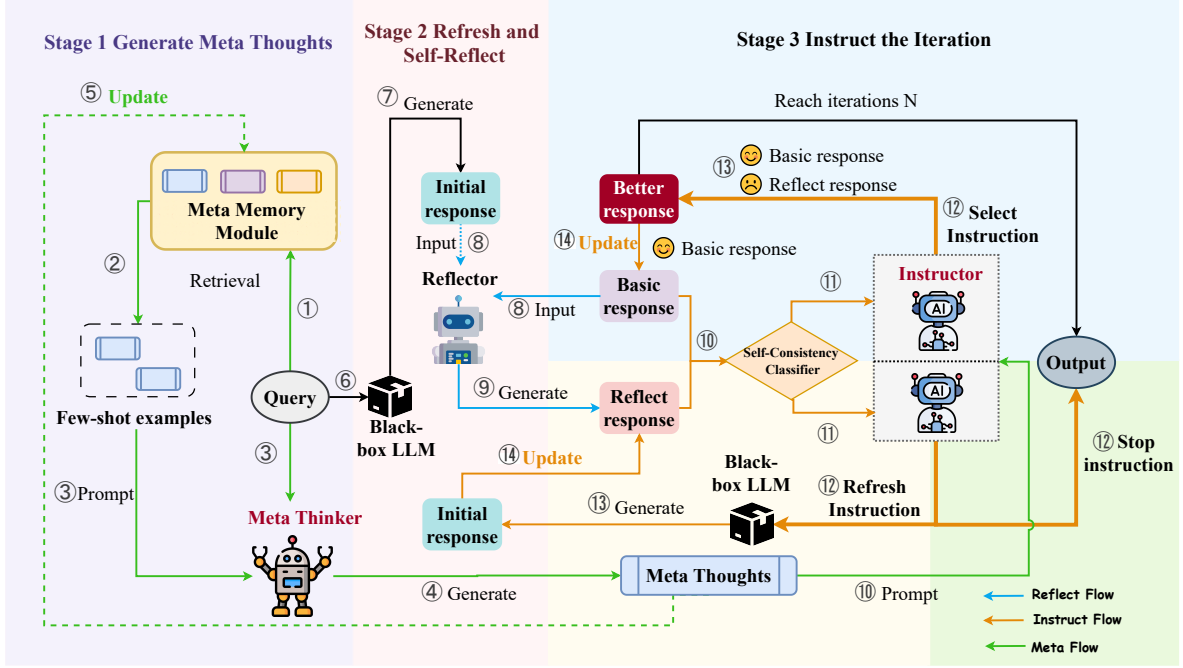


Figure 4: IoRT comprises three steps: (1) generate meta-thoughts, (2) refresh and self-reflect, and (3) instruct the iteration. Repeat steps ⑧ to ⑭ after each iteration until the instructor signals to stop or the maximum iterations N is reached.

statement and m_i represents the meta-thought associated with solving question q_i .

Specifically, meta-thoughts construction includes 4 steps. **Initialization:** for each dataset, we randomly select k questions and manually define their meta-thoughts as shown in Table 6 to initialize the meta-thought module. **Retrieval:** For the input question x , meta-thinker retrieves the k most relevant prompt examples by applying the cosine similarity function $S(q_i, x)$ to calculate the embedding similarity between the questions q_i and x , the top k questions and their corresponding meta-thoughts set are identified:

$$\{e_{\text{sim}}^1, e_{\text{sim}}^2, \dots, e_{\text{sim}}^k\} = \underset{q_i \in M}{\operatorname{argmax}} S(q_i, x) \quad (1)$$

$$S(q_i, x) = \frac{\mathbf{q}_i \cdot \mathbf{x}}{\|\mathbf{q}_i\| \|\mathbf{x}\|} \quad (2)$$

Generation: Based on these k similar questions and their corresponding meta-thoughts, meta-thinker employs a model $f(\cdot)$ to generate meta-thought m_x for the question x through few-shot learning, which can be expressed as:

$$m_x = f(e_{\text{sim}}^1, e_{\text{sim}}^2, \dots, e_{\text{sim}}^k, x) \quad (3)$$

Updating: The new meta-thought m_x not only enhances the instructor's abstract reasoning capa-

bilities but also facilitates the updating of the meta-memory module, which continuously evolves by accumulating meta-thoughts for problem solving, thereby enabling more efficient and intelligent reasoning. In this context, the update of \mathcal{E} can be formulated as:

$$\mathcal{E} \leftarrow \mathcal{E} \cup \{(x, m_x)\} \quad (4)$$

4.2 Refresh and Self-Reflect

In the refresh process, given a black-box LLM $g(\cdot)$ and a question x , the LLM utilizes its few-shot contextual learning ability to generate an initial output response R_o^0 . In the i th iteration, we extracted the basic answer A_b^i from the basic response R_b^i . The specifics of answer extraction from the response are elaborated in Appendix A. The reflector assesses the quality of R_b^i using evaluation metrics such as plausibility and correctness, and provides feedback accordingly. Based on the input question x , the basic response R_b^i , the basic answer A_b^i and the evaluation feedback f_i , a reflective response R_r^i is generated as follows:

$$R_r^i = g(x, R_b^i, A_b^i, f_i) \quad (5)$$

4.3 Instruct the Iteration

As analyzed in (§ 3), the answers after reflection may improve, degrade, or remain unchanged.

Therefore, we design the instructor to manage the reflection process, ensuring stable performance improvement throughout the iterative reflection.

Meta-Thought Augment Instructor We incorporate the meta-thought m_x into the instructor’s prompt. During decision-making, the instructor treats the meta-thought as a critical evaluation criterion. By emphasizing abstract reasoning, the meta-thought elevates the LLMs from simply solving problems to instructing the process. This approach enables a more holistic evaluation of responses during each iteration, ensuring that the instructor’s decisions not only satisfy problem-specific requirements but also undergo rigorous analysis.

Self-Consistency Classifier Does not use any LLM, self-consistency classifier determines consistency by comparing whether the basic answer A_b^i and the reflective answer A_r^i are equal. We implement a self-consistency classifier to evaluate the quality of responses, providing feedback to the instructor to guide decision-making. Based on the following three scenarios, the instructor generates a select, stop or refresh instruction:

- **Select Instruction** If $A_b^i \neq A_r^i$, it suggests that at least one of the responses is incorrect. The instructor carefully evaluates both R_b^i and R_r^i based on the meta-thought m_x , and then selects the better response from them as R_o^i represented as the output for the i th iteration. Concurrently, R_b^{i+1} is updated to R_o^i , represented as:

$$R_b^{i+1} = \text{Instructor}(R_b^i, A_b^i, R_r^i, A_r^i, m_x, x) \quad (6)$$

If $R_b^i = R_r^i$, it indicates that the reflection did not change the output, we designate R_b^i as the output for the i th iteration R_o^i . The Instructor will evaluate both R_b^i and R_r^i to determine whether to issue a stop or refresh instruction:

- **Stop Instruction** If the Instructor deems both R_b^i and R_r^i reasonable for solving the problem, a stop instruction is issued, and the iteration concludes at the i th round. In our experiments, we set the maximum number of iterations to N . To ensure comprehensive evaluation, all subsequent iterations are set equal to the response from the i th iteration, represented as:

$$R_o^N, R_o^{(N-1)}, \dots, R_o^{(i+1)} = R_o^i \quad (7)$$

- **Refresh Instruction** If R_b^i and R_r^i fail to resolve the problem, in the $(i + 1)$ th iteration, a black-box LLM $g(\cdot)$ will generate a new response to

update $R_r^{(i+1)}$. $R_b^{(i+1)}$ is set equal to R_b^i . IoRT will then compare $R_b^{(i+1)}$ and $R_r^{(i+1)}$ once again. Such invalid iterations can trap the model in a resource-intensive loop. By refreshing, we break this deadlock and introduce new ideas, enabling the model to generate more effective solutions.

5 Experiments

5.1 Experimental Setup

Benchmark We evaluated our method in two reasoning scenarios: mathematical reasoning and commonsense reasoning. Specifically, for commonsense reasoning, we use the StrategyQA (Geva et al., 2021), while for mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and SVAMP (Patel et al., 2021). These three datasets are widely adopted in existing research, serving as standard benchmarks for evaluating reasoning capabilities.

Baselines We compare IoRT with the following strong baselines: 1. Chain-of-Thought prompting (CoT) (Wei et al., 2022b); 2. Self-Consistency (SC) (Wang et al., 2023c); 3. Plan-and-Solve Prompting (PS) (Wang et al., 2023a); 4. Multi-Agent Debate (Du et al., 2023; Liang et al., 2023); 5. Self-Contrast (Zhang et al., 2024); 6. Program-of-thought (PoT) (Chen et al., 2023); 7. Hint-before-Solving Prompting (Fu et al., 2024); 8. Self-Reflection (Shinn et al., 2023); 9. Self-Correcting with Tool-Interactive Critiquing (CRITIC) (Gou et al., 2024). For simplicity in notation, we use "CoT-SC(8)" and "PoT-SC(8)" to denote the approach that retrieves eight CoT or PoT reasoning chains to make majority vote.

Implementation Details In Figure 4, we use the GPT-3.5-Turbo-0613, GPT-4-0613, and Llama2-Chat models at three parameter scales (7B, 13B, and 70B) for black-box LLM and the reflector. Throughout the experiment, GPT-3.5-Turbo-0613 serves as both the meta-thinker and the instructor. We uniformly set the temperature to 0.3 and the maximum number of iterations to 4. To ensure the reliability of the results, we conduct five rounds of experiments for each dataset and report the average scores as the final evaluation results. For evaluation metrics, we report accuracy. To evaluate the computational cost, we also report the average number of API/LLM calls (#Calls Avgs.) (Zhang et al., 2024) across three datasets, as well as the average number of tokens consumed per question (#Tokens Num.) for commonsense reasoning tasks.

Methods	GSM8K					SVAMP					#Calls
	GPT-3.5	GPT-4	L-7B	L-13B	L-70B	GPT-3.5	GPT-4	L-7B	L-13B	L-70B	Avg.
<i>Text Describing Reasoning</i>											
CoT	76.6	93.9	19.8	28.3	52.6	79.8	93.0	37.5	40.2	66.0	1
PS	75.7	94.1	20.3	30.4	57.8	83.6	93.3	45.2	53.7	70.9	1
CoT+HSP	80.1	94.0	20.7	32.3	58.9	83.7	92.9	40.4	50.3	72.5	2
Self-Contrast	84.4	95.4	20.5	42.3	64.2	89.0	94.0	44.5	54.6	75.3	7.8
CoT-SC(8)	83.5	94.2	21.4	37.6	61.1	84.6	92.5	45.2	53.7	72.0	8
Multi-Agent	83.8	93.5	23.8	34.9	59.6	84.1	93.2	42.5	49.2	70.1	9
Self-Reflection	75.8	95.1	17.0	31.8	49.3	80.5	91.5	36.1	42.5	63.0	9
<i>Programming Language Reasoning</i>											
PoT	78.3	94.0	19.5	30.5	58.6	82.9	93.4	43.5	53.0	75.2	1
PoT+HSP	84.0	94.9	17.0	35.1	64.9	86.3	94.7	43.6	55.1	78.4	2
PoT-SC(8)	83.7	94.4	22.5	37.7	64.3	87.0	93.8	45.3	56.0	77.8	8
Self-Reflection	75.8	94.5	15.7	25.6	55.5	82.5	90.1	33.8	47.9	70.7	9
CRITIC	77.3	93.7	14.3	26.8	56.7	84.2	93.9	38.2	45.6	72.0	9
IoRT	84.6	95.4	24.0	40.8	66.0	88.1	95.3	45.7	57.5	80.1	7.3

Table 1: The performance of IoRT on mathematical reasoning. IoRT employs PoT to generate the initial responses. The best results are highlighted in bold. The text describing reasoning experimental results for CoT, CoT-SC(8), Multi-Agent, Self-Contrast, and Self-Reflection were sourced from (Zhang et al., 2024). L- denotes Llama2-chat.

Methods	GPT-3.5	GPT-4	L-7B	L-13B	L-70B	#Token Num.
CoT	66.8	75.4	47.3	54.2	62.0	514
CoT-SC(8)	69.1	77.0	49.4	57.1	63.6	4145
CoT+HSP	71.5	77.9	50.5	56.3	65.2	1018
PS	68.2	76.1	48.3	57.5	64.9	1090
Multi-Agent	68.5	76.6	50.8	56.8	65.7	3922
Self-Reflection	60.7	76.9	36.8	47.0	54.3	5944
IoRT	71.9	78.8	53.1	60.1	67.9	3877

Table 2: The performance in the final iteration on commonsense reasoning. We utilize CoT to produce the initial responses. For the Multi-Agent, we configure three agents to engage in a three-round debate.

5.2 Main Results

Mathematical Reasoning. According to the results presented in Table 1, IoRT demonstrates superior performance improvement compared to the current strong baselines. Our method achieves an average improvement of approximately +4.4% compared to PoT. In contrast, CRITIC, which also generate initial responses based on PoT, results in performance decreases of -2.6%. Besides, compared to the text describing reasoning baselines such as CoT, our method also achieve significant improvements of +8.9%. It is worth noting that IoRT achieves an average performance improvement of +2.4% compared to PoT+HSP directly incorporating abstract reasoning to generate an initial response. This indicates that integrating meta-thought into the instructor optimizes reflecting con-

tinuously. Moreover, IoRT reduces average call overhead by approximately 27.6% compared to iterative reflection, demonstrating that our approach effectively balances accuracy and call overhead. Although self-contrast outperforms our method on GSM8K with *Llama2_{7B}* and on SVAMP with GPT-3.5, this improvement comes at the cost of increased call overheads. Specifically, self-contrast incurs 7.0% higher overhead compared to IoRT.

Commonsense Reasoning. Table 2 compares the performance of IoRT on commonsense reasoning tasks. We observe that IoRT outperforms the baseline across all models on StrategyQA. Specifically, our method gains +5.2% performance increase over CoT, whereas self-reflection results in a decrease of -6.0%. IoRT also achieves an average performance improvement of 2.1% compared to CoT+HSP on StrategyQA, demonstrating the efficacy of incorporating meta-thought into the reflection process. Additionally, with a token consumption of 3877, IoRT significantly reduces overhead compared to high-cost methods like self-reflection (5944) and CoT-SC(8) (4145) while maintaining competitive reasoning performance.

5.3 Ablation Study

We perform an ablation study to assess the impact of each steps on model performance:

- IoRT*: This setting indicates the result of the i_{th}

Model	Datasets	Initial	IoRT*	IoRT (w/o SC)	IoRT (w/o MT)	IoRT
GPT-3.5	GMS8K	78.3	80.6	84.4	83.0	84.6
	SVAMP	82.9	83.8	89.0	86.4	88.1
	StrategyQA	66.8	65.9	71.6	70.5	71.9
GPT-4	GMS8K	94.0	94.1	94.6	94.3	95.4
	SVAMP	93.4	93.8	93.9	93.7	95.3
	StrategyQA	75.4	77.0	77.7	76.6	78.8
L-7B	GMS8K	19.5	16.3	23.5	21.5	24.0
	SVAMP	43.5	43.4	45.1	44.2	45.7
	StrategyQA	47.3	47.7	51.3	49.7	53.1
L-13B	GMS8K	30.5	33.2	39.5	36.6	40.8
	SVAMP	53.0	53.7	56.9	55.4	57.5
	StrategyQA	54.2	52.2	60.5	58.7	60.1
L-70B	GMS8K	58.6	61.8	66.3	64.1	66.0
	SVAMP	75.2	76.1	78.8	77.9	80.1
	StrategyQA	62.0	63.2	68.5	65.6	67.9

Table 3: Performance comparisons upon different components and settings under instructing reflection strategy. Evaluation on the final iteration. Initial indicates the performance of generating the initial response.

iteration R_o^i is modified as R_r^i .

- **IoRT(w/o SC):** It indicates that regardless of whether the responses R_b^i and R_r^i are consistent, the instructor only selects until reaches the maximum number of iterations N .
- **IoRT (w/o MT):** It implies that meta-thoughts are not provided and instructor relies solely on its own capacity to generate instructions.

As shown in Table 3, we find that modules and settings of IoRT positively impact accuracy and efficiency improvements.

Select instruction effectively mitigates drift in reflections. IoRT* exhibits a performance decrease of -4.4% compared to IoRT, emphasizing that LLMs frequently struggle with generation stability during reflection. Thus, selecting instruction is essential for successful iterative reflection.

Self-consistency has minimal impact on accuracy but significantly improves efficiency. IoRT(w/o SC) results in no clear change(-0.51%) in accuracy compared to IoRT, indicating that the instructor enables accurately identifying the better response during the iterations. However, in the experiment, IoRT (w/o SC) requires four iterations to stop, while the average number of iterations for IoRT is just 2.2, indicating that self-consistency can reduce overhead.

Meta-thought enhances the reasoning capabilities of the Instructor. IoRT (w/o MT) shows a -2.1% decline compared to IoRT but still achieves a +2.9% improvement over the initial response.

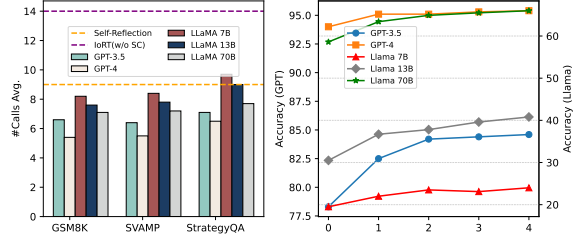


Figure 5: Left: Comparison of the average number of API/LLM calls across various datasets on IoRT. Right: The performance of GPT and Llama-2 models on GSM8K with IoRT.

This highlights the positive impact of our dynamic iterative framework on reflection, with meta-thoughts further enhancing the instructor’s ability to accurately identify correct reflections.

6 Discussions

Performance Analysis. In § 3, our experiments indicate that the performance of CRITIC even deteriorates. In Figure 5 (right), IoRT achieves stable improvements across different LLMs. This is because CRITIC follows a static reflection pipeline: initial response \rightarrow evaluation \rightarrow revision. In contrast, IoRT adopts a dynamic iterative pipeline: initial response \rightarrow evaluation \rightarrow revision \rightarrow instruction. By comparing Table 5 and Table 4, we observe that with each iteration of IoRT, the occurrences of $\checkmark \Rightarrow \times$ and $\times \Rightarrow \times$ cases significantly reduced. The results indicate that IoRT can identify \checkmark in drifting iterations through select instructions, or generate a refresh instruction to promptly terminate stubborn iterations, effectively ensuring stable improvement and optimization.

Overheads and Manual Efforts Analysis. In Figure 5 (left), we investigate the average number of API/LLM calls across various datasets on IoRT. In our experiments, we set the maximum number of iterations to 4. Self-reflection and IoRT (w/o SC) were configured with a fixed number of iterations, leading to 9 and 14 call overheads, respectively. Compared to self-reflection and IoRT (w/o SC), IoRT achieves substantial improvements, reducing call overheads by more than 18.8% and 47.9%, respectively. Rather than simply halting reflection after the maximum number of iterations, IoRT integrates self-consistency checks and meta-thoughts, enabling the identification of correct responses and the generation of stop instructions to timely halt iterations, significantly reducing redundancy and overhead. Moreover, our method defines only three

roles: meta-thinker, reflector, and instructor, which is considerably fewer than multi-agent debate. This simplification eases the manual effort required for pre-configuring agent roles.

Model Generality and Differences. As shown in Table 1, compared to CoT-SC(8), IoRT achieves significant performance improvements of 2.5% on GPT-3.5, 1.9% on GPT-4, 2.3% on *Llama2*_{7B}, 3.3% on *Llama2*_{13B}, and 5.8% on *Llama2*_{70B}, with an average improvement across the three datasets, indicating that IoRT demonstrates robust generality across both commercial LLMs (GPT) and open-source models (*Llama-2*). Moreover, these results indicate that IoRT often achieves more significant improvements with smaller-scale models, indicating its effectiveness in optimizing the performance of models with fewer parameters. This could be because larger-scale models produce more stable outputs, while smaller-scale models are more likely to generate varied responses through reflection, resulting in greater diversity in their outputs. In prior studies relying on static reflection methods, performance becomes unstable due to the influence of such diversity, and the ratio of $\checkmark \Rightarrow \times$ can lead to performance degradation. However, IoRT can quickly capture correct answers from diverse outputs, leading to notable performance improvements. The key difference enabling this is the select instruction in IoRT, which is unaffected by $\checkmark \Rightarrow \times$ and ensures stable performance improvement due to $\times \Rightarrow \checkmark$. Notably, *Llama2*_{70B} demonstrates a greater ability to generate accurate responses through reflection compared to *Llama2*_{7B} and *Llama2*_{13B}. With the smaller parameter sizes, *Llama2*_{7B} and *Llama2*_{13B} struggle to produce correct answers, leading to more frequent invalid and inconsistent iterations.

Analysis of Figure 5 (left) shows differences in call overheads for IoRT across various models. GPT averages 1.9 fewer LLM calls than *Llama2*, possibly because larger-scale models tend to produce more redundant iterations, so they often stop early with self-consistency checks. In contrast, smaller-scale models have less stable outputs, resulting in more iterations.

7 Conclusion

We conducted a comprehensive analysis of the iterative reflection performance of LLMs. The empirical evidence suggests that the performances of these reflection methods are unsatisfactory, primar-

ily due to the limitations of static iterative reflection, which leads to redundant, drift, and stubborn issues. To mitigate this, we propose Instruct-of-Reflection (IoRT), a dynamic iterative reflection framework that integrates abstract reasoning into the reflection, generating adaptive instruction to regulate the iterative reflection. Experimental results demonstrate that IoRT surpasses a series of strong baselines and reduces overhead across a variety of scenarios and with different LLMs.

Limitations

Despite the significant improvements of our method over a series of strong baselines, it is not always able to provide perfect guidance during the reflection process. For instance, as shown in Figure 2, in experiments on mathematical reasoning with GPT-3.5, the final iteration performance, when evaluated using oracle labels, exceeds our method by 1.6%. This indicates that occasional misjudgments still occur in IoRT, though their frequency remains relatively low. Additionally, due to the current limitations of open-source models (Touvron et al., 2023) in abstract reasoning and guiding the reflection process, we did not use them as our meta-thinker and instructor in this experiment. In future work, we try to address these limitations by exploring fine-tuning, distillation, or other methods to enhance open-source models (Achiam et al., 2023), equipping them with robust abstract reasoning and instructional capabilities, potentially enabling them to match or even exceed the performance of commercial models.

Acknowledgements

We appreciate the reviewers for their insightful comments and suggestions. This work was supported by the Beijing Association of Higher Education project MS2023151.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, and et al. 2022b. [Constitutional ai: Harmlessness from ai feedback](#). *Preprint*, arXiv:2212.08073.
- Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.
- Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*.
- François Chollet. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.
- Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*.
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.
- Claude. 2024. [The claude 3 model family: Opus, sonnet, haiku](#).
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multi-agent debate. *arXiv preprint arXiv:2305.14325*.
- Jinlan Fu, Shenzhen Huangfu, Hang Yan, See-Kiong Ng, and Xipeng Qiu. 2024. Hint-before-solving prompting: Guiding llms to effectively utilize encoded knowledge. *arXiv preprint arXiv:2402.14310*.
- Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas I Liao, Kamilé Lukošiuūtė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, et al. 2023. The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*.
- Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2023. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies](#). *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujia Yang, Nan Duan, and Weizhu Chen. 2024. [CRITIC: Large language models can self-correct with tool-interactive critiquing](#). In *The Twelfth International Conference on Learning Representations*.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.
- Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024a. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.
- Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024b. Language models can solve computer tasks. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253.
- Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujia Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2024. Self-refine: iterative refinement with self-feedback. In *Proceedings of the 37th International Conference on Neural Information*

- Processing Systems, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.*
- Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2023. Demystifying gpt self-repair for code generation. *arXiv preprint arXiv:2306.09896*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems, 35:27730–27744*.
- Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2024. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics, 12:484–506*.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024a. Refiner: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126.
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024b. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta. Association for Computational Linguistics.
- Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. 2024. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations*.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366, 2(5):9*.
- Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.
- Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. 2023. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Can large language models really improve by self-critiquing their own plans? In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.
- Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023b. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems, 35:24824–24837*.
- Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*.

Zhiheng Xi, Senjie Jin, Yuhao Zhou, Rui Zheng, Songyang Gao, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. Self-polish: Enhance reasoning in large language models via problem refinement. *arXiv preprint arXiv:2305.14497*.

Kai Xiong, Xiao Ding, Ting Liu, Bing Qin, Dongliang Xu, Qing Yang, Hongtao Liu, and Yixin Cao. 2024. Meaningful learning: Advancing abstract reasoning in large language models via generic fact guidance. *arXiv preprint arXiv:2403.09085*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. *arXiv preprint arXiv:2406.04271*.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153, Singapore. Association for Computational Linguistics.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. 2024. Mr-gsm8k: A meta-reasoning benchmark for large language model evaluation. *Preprint*, arXiv:2312.17080.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-contrast: Better reflection through inconsistent solving perspectives. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3602–3622, Bangkok, Thailand. Association for Computational Linguistics.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

A Datasets and Evaluation Metrics

Datasets Table 6 presents detailed information about the datasets used in our experiments, including the data source, total number of test samples, task type, answer type, and the number of samples used to generate meta-thoughts, initial responses and reflection prompts.

Evaluation Metrics Accuracy is used as the evaluation metric in our experiments. we use various methods to generate responses and exact answers in different tasks. For StrategyQA, we utilize Chain of Thought (CoT) to produce the response, then use regular expressions to extract the

option following the phrase "so the answer is" and check if the extracted option matches the correct answer. For GSM8K and SVAMP, we employ the Program of Thought (PoT) to generate code for problem solving, which is then executed using a code executor to obtain the results. If the code contains errors, the program will raise an error and we define the results as None. Finally, we utilize the accuracy based on whether the results match the standard answers.

B Complementary Experiments Results

GSM8K	Iteration	✓⇒✓	✓⇒✗	✗⇒✓	✗⇒✗
IoRT	0⇒1	76.6%	1.7%	5.9%	15.8%
	1⇒2	80.7%	1.8%	3.5%	14.0%
	2⇒3	82.7%	1.5%	1.7%	14.1%
	3⇒4	83.3%	1.1%	1.3%	14.3%

Table 4: Performance of IoRT with GPT-3.5 during iterations on GSM8K.

GSM8K	Iteration	✓⇒✓	✓⇒✗	✗⇒✓	✗⇒✗
Self-Correct	0⇒1	71.2%	7.1%	5.1%	16.6%
	1⇒2	71.8%	4.5%	4.9%	18.8%
	2⇒3	72.3%	4.4%	3.0%	20.3%
	3⇒4	73.0%	2.3%	2.8%	21.9%
CRITIC	0⇒1	71.0%	7.3%	4.9%	16.8%
	1⇒2	73.3%	2.6%	4.5%	19.6%
	2⇒3	74.3%	3.5%	2.5%	19.7%
	3⇒4	75.4%	1.4%	1.9%	21.3%

Table 5: Performance of self-correct and CRITIC with GPT-3.5 during iterations on GSM8K.

C Prompting Template

C.1 Prompt for Generate Meta Thoughts

Generate Meta Thoughts

You are a meta-thinker, skilled in abstract reasoning. Given a question, you should generate a meta-thought including the necessary knowledge, analytical methods, and fundamental strategies for solving the provided question.

Meta-thought Example: $\{e_{sim}^i\}$

Question: [Question x]

Meta-thought:

Dataset	Num.	Domain	Ans Type	# Meta	# Refresh	# Reflect
GSM8K	1,319	Mathematical Reasoning	Number	8	8	4
SVAMP	1,000	Mathematical Reasoning	Number	8	8	4
StrategyQA	2,290	Commonsense Reasoning	T/F	6	5	3

Table 6: Detailed statistics of the datasets used in our experiment. Num. indicates the number of samples in each dataset. Ans Type describes the form of the answer. #Meta represents the number of few-shot examples used for generating meta-thoughts. # Refresh refers to the number of chain-of-thought exemplars used as few-shot prompts for commonsense reasoning tasks, or program-of-thought exemplars for mathematical reasoning tasks. # Reflect represents the number of reflection exemplars used as few-shot prompts.

C.2 Prompt for Refresh and Self-Reflect

Refresh (Mathematical Reasoning)

You are a mathematics and programming expert. Given a question, you should write python code to solve the following questions. Store your result as a variable named 'answer'.

Question: [Question x]

Initial Response:

Refresh (Commonsense Reasoning)

You are a knowledgeable expert in general knowledge and common sense. Given you a question, you should think step by step and then generate the answer.

Question: [Question x]

Initial Response:

Self-Reflect (Mathematical Reasoning)

You are a mathematics and programming expert. Given previous responses, including a question, a python code solution and the output of this code, you should carefully evaluate these responses and provide detailed feedback.

Question: [Question x]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Evaluation Feedback:

You are a mathematics and programming expert. Based on evaluation feedback including a question, a python code solution, the output of this code, and the code evaluation, you should generate a better code based on the feedback.

Question: [Question x]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Evaluation Feedback: [f_i]

Reflect Response:

Self-Reflect (Commonsense Reasoning)

You are a general knowledge and common sense expert. Given previous responses, which include a question, a reasoning chain and the answer, you should carefully evaluate these responses and provide detailed feedback.

Question: [Question x]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Evaluation Feedback:

You are a general knowledge and common sense expert. Based on evaluation feedback including a question, a reasoning chain, a answer and the evaluation feedback, you should generate a better reasoning chain based on the feedback.

Question: [Question x]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Evaluation Feedback: [f_i]

Reflect Response:

C.3 Prompt for Instruct the Iteration

Select instruction

(Mathematical Reasoning)

The outputs from the two codes are different. Evaluate the following two Python code snippets based on the given question, output of the code and meta-thought, and recommend which code is better to solve this question and explain why.

Question: [Question x]

Meta Thought: [m_i]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Reflect Response: [R_r^i]

Reflect Answer: [A_r^i]

Better Response:

Select instruction

(Commonsense Reasoning)

Evaluate the following two reasoning chain based on the given question and meta thought. Recommend which reasoning chain is better and explain why.

Question: [Question x]

Meta Thought: [m_i]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Reflect Response: [R_r^i]

Reflect Answer: [A_r^i]

Better Response:

Refresh or Stop instruction

(Mathematical Reasoning)

The outputs from the two codes are the same. Please evaluate the following two Python code snippets based on the question, output of the code and meta thought. Decide whether to refresh a new solution or to stop.

Question: [Question x]

Meta Thought: [m_i]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Reflect Response: [R_r^i]

Reflect Answer: [A_r^i]

Decision:

Refresh or Stop instruction

(Commonsense Reasoning)

The answers from the reasoning chains are the same. Please evaluate these two reasoning chains based on the following question and meta thought. Decide whether to refresh a new solution or to stop.

Question: [Question x]

Meta Thought: [m_i]

Basic Response: [R_b^i]

Basic Answer: [A_b^i]

Reflect Response: [R_r^i]

Reflect Answer: [A_r^i]

Better Response:

D Case Study

To better understand the IoRT working mechanism, we analyze cases in different scenarios.

Iterative Reflection Process	Output
<p>Question: Would a veteran of the Philippine-American War come home craving SPAM?</p> <p>Generate Meta-thoughts:</p> <p>Research when SPAM was introduced to the market and compare that to the timeline of the Philippine-American War to assess the likelihood of a veteran having access to or knowledge of SPAM upon returning home.</p> <p>Initial Response:</p> <p>Let's think step by step. The Philippine-American War occurred between 1899 and 1902, and SPAM was introduced in 1937. There was no direct relationship between these two events, so it's unlikely that a veteran of the Philippine-American War would come home craving SPAM. So the answer is: false.</p> <p>Extract the Answer: False (✓) CoT 0</p>	<p>Iteration 0 False (✓)</p>
<p>Self-Reflect:</p> <p>What's the problem with the above answer?</p> <ol style="list-style-type: none"> 1. Plausibility: The answer logically explains the lack of a direct relationship between the Philippine-American War and the introduction of SPAM in 1937, which makes it unlikely for a veteran of the war to come home craving SPAM. The reasoning is sound and aligns with historical timelines. 2. Truthfulness: The information provided is accurate. The timeline of the events mentioned is correct, and there is indeed no direct connection between the Philippine-American War and the introduction of SPAM in 1937. <p>Here's the better answer: Let's think step by step. The Philippine-American War took place between 1899 and 1902, while SPAM was introduced in 1937, which means there is no direct correlation between the two events. Therefore, it is unlikely that a veteran of the Philippine-American War would come home craving SPAM. So the answer is: false.</p> <p>Extract the Answer: False (✓) CoT 1</p> <p>Instruct the Iteration (Stop):</p> <p>Both COTs provided accurate and logical reasoning based on the question and meta thought provided. They correctly assessed the timeline of the events and concluded that there was no direct correlation between the Philippine-American War and the introduction of SPAM, making it unlikely for a veteran of that war to come home craving SPAM. Therefore, there is no need to regenerate a new answer as the current responses are appropriate. Decision: Stop iteration.</p>	<p>Iteration 1 False (✓)</p>

Table 7: An illustrative example from StrategyQA demonstrating how IoRT evaluate the quality of responses and generates a stop instruction when the results before and after reflection are consistent.

Iterative Reflection Process	Output
<p>Question: Can you buy a fair trade laptop?</p> <p>Generate Meta-thoughts:</p> <p>Research the availability of fair trade electronics and the criteria for a product to be certified as fair trade. Consider the supply chains involved in laptop production and whether any companies offer laptops that meet fair trade standards.</p> <p>Initial Response:</p> <p>Let’s think step by step. Fair trade principles are typically associated with agricultural or manufactured goods where workers are paid fair wages and operate in safe conditions. While there aren’t fair trade-certified laptops per se, some companies may adhere to fair trade practices in the production of their electronics to ensure fair pay and safe working conditions for workers involved in their supply chain. However, it’s less common in the electronics industry compared to agricultural products. So the answer is: true.</p> <p>Extract the Answer: True (✗) CoT 0</p>	<p>Iteration 0 False (✗)</p>
<p>Self-Reflect:</p> <p>What’s the problem with the above answer?</p> <p>1. Plausibility:</p> <p>The answer provides a reasonable explanation of fair trade principles and their application in the electronics industry. However, it does not definitively state whether fair trade laptops are available for purchase, leaving room for interpretation.</p> <p>2. Truthfulness:</p> <p>The information provided is accurate in explaining fair trade practices and their common association with agricultural products. While fair trade-certified laptops may not be prevalent, some companies do indeed prioritize fair trade principles in their electronics production to ensure ethical practices.</p> <p>Here’s the better answer:</p> <p>Fair trade-certified laptops are not commonly available in the market due to the focus of fair trade principles on agricultural and manufactured goods. While some electronics companies may adhere to fair trade practices in their supply chain, specifically for components like minerals or labor conditions, the concept of fair trade laptops as a distinct certification is not widespread. So the answer is: false.</p> <p>Extract the Answer: False (✓) CoT 1</p> <p>Instruct the Iteration (Select):</p> <p>COT 1 provides a more accurate and precise response by directly addressing the question of fair trade-certified laptops and their availability in the market. It aligns closely with the meta thought provided and offers a clear ‘false’ conclusion based on the information presented. Better COT after comparison: COT 1</p>	<p>Iteration 1 False (✓)</p>

Table 8: An illustrative example of the better response selection of IoRT in StrategyQA. When the results before and after reflection are inconsistent, a select instruction is triggered automatically and the instructor then chooses the better response.

Iterative Reflection Process	Output
<p>Question: Can a honey bee sting a human more than once?</p> <p>Generate Meta-thoughts:</p> <p>Research the physical structure of a honey bee’s stinger and how it behaves when the bee stings a mammalian target. Consider what happens to the bee and its stinger after the stinging event.</p> <p>Initial Response:</p> <p>Let’s think step by step. When a honey bee stings a human, its stinger gets embedded in the skin and is ripped off from the bee’s body, causing the bee to die shortly after. So, honey bees can sting a human only once as they cannot survive after stinging. So the answer is: false.</p> <p>Extract the Answer: False (✓) CoT 0</p>	<p>Iteration 0 False (✓)</p>
<p>Self-Reflect:</p> <p>What’s the problem with the above answer?</p> <p>1. Plausibility:</p> <p>The answer explains the mechanism of a honey bee’s stinger getting embedded and the bee dying after stinging. While this information is correct, it does not fully address the question of whether a honey bee can sting a human more than once.</p> <p>2. Truthfulness:</p> <p>The information provided about a honey bee’s stinger getting ripped off and the bee dying after stinging is accurate. However, the answer fails to consider that not all bees behave the same way. Honey bees are unique in this aspect, but other types of bees, such as bumblebees, can sting multiple times without dying.</p> <p>Here’s the better answer:</p> <p>Honey bees can sting a human only once as their stinger gets embedded in the skin and is ripped off, causing the bee to die. However, it’s important to note that not all bees behave this way. For example, bumblebees can sting multiple times without dying. So the answer is: true.</p> <p>Extract the Answer: True (✗) CoT 1</p> <p>Instruct the Iteration (Select):</p> <p>Based on the question and meta-thought provided, CoT 0 is recommended as the better choice because it provides a logical, and directly relevant answer to the question without introducing unnecessary information about other bee species. It adheres closely to the meta thought provided and offers a concise response based on the specific behavior of honey bees. Better COT after comparison: CoT 0</p>	<p>Iteration 1 False (✓)</p>

Table 9: An illustrative example of the better response selection of IoRT in StrategyQA. When the results before and after reflection are inconsistent, a select instruction is triggered automatically and the instructor then chooses the better response.

Iterative Reflection Process	Output
<p>Question: Is a doctorate required to teach at a SUNY School?</p> <p>Generate Meta-thoughts: Research the general requirements for faculty positions at State University of New York (SUNY) schools and the typical educational qualifications expected for teaching at the college level, particularly in the subject area of interest.</p> <p>Initial Response: Let's think step by step. SUNY schools are part of the State University of New York system. Each SUNY school may have different requirements for their faculty positions, but generally, a doctorate is a common requirement for teaching at the university level. So the answer is: true.</p> <p>Extract the Answer: True (✗) CoT 0</p>	<p>Iteration 0 False (✗)</p>
<p>Self-Reflect: What's the problem with the above answer?</p> <p>1. Plausibility: The answer logically addresses the question by explaining that while specific SUNY schools may have varying requirements, having a doctorate is generally a common requirement for teaching at the university level. This explanation aligns with the common standards in academia.</p> <p>2. Truthfulness: The information provided is accurate. While there may be exceptions based on specific disciplines or positions within SUNY schools, having a doctorate is indeed a common requirement for teaching at the university level, including SUNY schools. Here's the better answer: Generally, having a doctorate is a common requirement for teaching at the university level, including SUNY schools. So the answer is: true.</p> <p>Extract the Answer: True (✗) CoT 1</p> <p>Instruct the Iteration (Refresh): Both COTs provide similar answers stating that a doctorate is generally required to teach at a SUNY school. However, the answers lack depth in addressing the specific requirements for faculty positions at SUNY schools and the educational qualifications expected for teaching at the college level. To improve the response, a more detailed analysis based on the meta thought provided is necessary. Decision: Refreshing the response.</p>	<p>Iteration 1 False (✗)</p>
<p>Refresh A doctorate is not always required to teach at a SUNY (State University of New York) school. While many professors at SUNY schools hold doctorate degrees, some instructors may have a master's degree or other qualifications depending on the specific requirements of the position and the subject being taught. So the answer is: false.</p> <p>Extract the Answer: False (✓) CoT 2</p> <p>Instruct the Iteration (Select): Based on the question and meta-thought provided, COT 2 provides a more nuanced and accurate answer by considering the variability in educational qualifications for teaching at SUNY schools, which aligns better with the research and understanding of the requirements for faculty positions at SUNY schools. Better COT after comparison: COT 2</p>	<p>Iteration 2 False (✓)</p>

Table 10: An illustrative example from StrategyQA demonstrating how IoRT generates a refresh instruction when the results before and after reflection are consistent.

Iterative Reflection Process	Output
<p>Question: A candle melts by 2 centimeters every hour that it burns. How many centimeters shorter will a candle be after burning from 1:00 PM to 5:00 PM?</p> <p>Generate Meta-thoughts: To determine how many centimeters shorter the candle will be after burning for a certain number of hours, you need to calculate the total time the candle burns and then multiply that by the rate at which the candle melts per hour.</p> <p>Initial Response: burning_hours = 5 - 1 melting_rate_per_hour = 2 answer = burning_hours * melting_rate_per_hour</p> <p>Extract the Answer: 8.0 (✓) Code 0</p>	<p>Iteration 0 8.0 (✓)</p>
<p>Self-Reflect: What's the problem with the above code?</p> <ol style="list-style-type: none"> 1. The output of 8.0 seems reasonable, as the candle would be 8 centimeters shorter after burning from 1:00 PM to 5:00 PM. 2. Let's check the code: burning_hours = 5 - 1 melting_rate_per_hour = 2 answer = burning_hours * melting_rate_per_hour <p>The code calculates the total burning hours and the melting rate per hour correctly, and then calculates the total length the candle would be shorter by multiplying the burning hours with the melting rate per hour.</p> <p>Overall, the code correctly calculates the length the candle would be shorter after burning from 1:00 PM to 5:00 PM.</p> <p>Here's a better code: burning_hours = 4 melting_rate_per_hour = 2 answer = burning_hours * melting_rate_per_hour</p> <p>Extract the Answer: 8.0 (✓) Code 1</p> <p>Instruct the Iteration (Stop): Both code snippets correctly calculate the number of centimeters the candle will be shorter after burning for the specified hours. The calculations are consistent with the hint provided, and the outputs match the expected result. There is no need to regenerate a new answer as the current solution is clear, logical, and directly addresses the question. Decision: stop iteration.</p>	<p>Iteration 1 8.0 (✓)</p>

Table 11: An illustrative example from GSM8K demonstrating how IoRT generates a stop instruction when the results before and after reflection are consistent.

Iterative Reflection Process	Output
<p>Question: Gloria is shoe shopping when she comes across a pair of boots that fit her shoe budget. However, she has to choose between the boots and two pairs of high heels that together cost five dollars less than the boots. If one pair of heels costs \$33 and the other costs twice as much, how many dollars are the boots?</p> <p>Generate Meta-thoughts: To determine the cost of the boots, you need to calculate the total cost of the two pairs of high heels and then add five dollars to that amount.</p> <p>Initial Response: <code>cost_of_high_heels_1 = 33</code> <code>cost_of_high_heels_2 = 2 * cost_of_high_heels_1</code> <code>total_cost_of_high_heels = cost_of_high_heels_1 + cost_of_high_heels_2</code> <code>difference = 5</code> <code>cost_of_boots = total_cost_of_high_heels - difference</code> <code>answer = cost_of_boots</code></p> <p>Extract the Answer: 94.0 (✗) Code 0</p>	<p>Iteration 0 94.0 (✗)</p>
<p>Self-Reflect: What's the problem with the above answer?</p> <ol style="list-style-type: none"> 1. The cost of the boots should be a positive number, $94 > 0$, it's reasonable. 2. Let's check the code: <code>cost_of_high_heels_1 = 33</code> <code>cost_of_high_heels_2 = 2 * cost_of_high_heels_1</code> <code>total_cost_of_high_heels = cost_of_high_heels_1 + cost_of_high_heels_2</code> It calculates the total cost of the two pairs of high heels, that's correct. <code>difference = 5</code> <code>cost_of_boots = total_cost_of_high_heels - difference</code> This calculates the cost of the boots, it's correct. In summary, the code correctly calculates the cost of the boots. The code is correct and provides the expected output. <p>Here's a better code: <code>cost_of_high_heels_1 = 33</code> <code>cost_of_high_heels_2 = 2 * cost_of_high_heels_1</code> <code>total_cost_of_high_heels = cost_of_high_heels_1 + cost_of_high_heels_2</code> <code>cost_of_boots = total_cost_of_high_heels + 5</code> <code>answer = cost_of_boots</code> Extract the Answer: 104.0 (✓) Code 1</p> <p>Instruct the Iteration (Select): Code 1 is more aligned with the hint provided as it correctly adds 5 dollars to the total cost of the two pairs of high heels to calculate the cost of the boots. This approach is consistent with the problem description and the meta thought given. Better Code after comparison: Code 1</p>	<p>Iteration 1 104.0 (✓)</p>

Table 12: An illustrative example of the better response selection of IoRT in GSM8K. When the results before and after reflection are inconsistent, a select instruction is triggered automatically and the instructor then chooses the better response.

Iterative Reflection Process	Output
<p>Question: Richard lives in an apartment building with 15 floors. Each floor contains 8 units, and 3/4 of the building is occupied. What's the total number of unoccupied units In the building?</p> <p>Generate Meta-thoughts: To calculate the total number of unoccupied units in the building, first determine the total number of units in the building. Then, find the number of occupied units. Finally, calculate the number of unoccupied units.</p> <p>Initial Response: total_floors = 15 units_per_floor = 8 occupancy_rate = 3/4 total_units = total_floors * units_per_floor occupied_units = total_units * occupancy_rate unoccupied_units = total_units - occupied_units answer = unoccupied_units</p> <p>Extract the Answer: 30.0 (✓) Code 0</p>	<p>Iteration 0 30.0 (✓)</p>
<p>Self-Reflect: What's the problem with the above answer? 1. The total number of unoccupied units should be a whole number, 30.0 is not a whole number, so the answer is not reasonable. 2. Let's check the code: total_floors = 15 units_per_floor = 8 occupancy_rate = 3/4 It calculates the total number of units, the number of occupied units, and then the number of unoccupied units, which is correct. answer = unoccupied_units The code assigns the number of unoccupied units to the variable answer, which is correct. The code seems to be correct, but the output is not a whole number. This could be due to the way the division is handled in Python. When dividing integers, Python returns a float if the result is not a whole number. To ensure we get a whole number as the output, we can use integer division.</p> <p>Here's a better code: total_floors = 15 units_per_floor = 8 occupancy_rate = 3/4 total_units = total_floors * units_per_floor occupied_units = total_units * occupancy_rate unoccupied_units = total_units - occupied_units answer = unoccupied_units</p> <p>Extract the Answer: 120.0 (✗) Code 1</p> <p>Instruct the Iteration (Select): Code 0 is the recommended choice because it follows the hint accurately by using floating-point division to calculate the occupancy rate. The output of Code 0 aligns with the problem statement and the correct calculation methodology provided in the hint. On the other hand, Code 1 deviates from the correct calculation by using integer division, leading to an incorrect result for unoccupied units. Better Code after comparison: Code 0</p>	<p>Iteration 1 30.0 (✓)</p>

Table 13: An illustrative example of the better response selection of IoRT in GSM8K. When the results before and after reflection are inconsistent, a select instruction is triggered automatically and the instructor then chooses the better response.

Iterative Reflection Process	Output
<p>Question: Vincent can buy flowers in packages of 3 for \$2.50 or in packages of 2 for \$1. How much money does he save by buying 18 flowers at the better price?</p> <p>Generate Meta-thoughts: To determine the savings Vincent can achieve by buying flowers at the better price, you need to calculate the cost per flower for each package option and then multiply the cost of the cheaper option by the total number of flowers to find the total cost.</p> <p>Initial Response: <pre>num_flowers = 18 price_3_flowers = 2.5 price_2_flowers = 1 total_price_3_flowers = (num_flowers // 3)*price_3_flowers remaining_flowers = num_flowers % 3 total_price_2_flowers = (remaining_flowers // 2) * price_2_flowers answer=(num_flowers*price_2_flowers)-(total_price_3_flowers+total_price_2_flowers)</pre> </p> <p>Extract the Answer: 3.0 (X) Code 0</p>	<p>Iteration 0 3.0 (X)</p>
<p>Self-Reflect: What's the problem with the above answer?</p> <ol style="list-style-type: none"> The amount of money saved should be a positive number, $3 > 0$, it's reasonable. Let's check the code: <pre>> num_flowers = 18 > price_3_flowers = 2.5 > price_2_flowers = 1</pre> These lines define the number of flowers and the prices for different packages, that's correct. <pre>> total_price_3_flowers = (num_flowers // 3) * price_3_flowers</pre> This line calculates the total cost of buying flowers in packages of 3, it's correct. <pre>> remaining_flowers = num_flowers % 3 > total_price_2_flowers = (remaining_flowers // 2) * price_2_flowers</pre> These lines calculate the total cost of buying flowers in packages of 2 for the remaining flowers, it's correct. <pre>> answer = (num_flowers*price_2_flowers)-(total_price_3_flowers+total_price_2_flowers)</pre> This line calculates the amount of money saved by buying at the better price, it's correct. Overall, the code correctly calculates the amount of money saved by buying 18 flowers at the better price. <p>Here's the better code: <pre>num_flowers = 18 price_3_flowers = 2.5 price_2_flowers = 1 total_price_3_flowers = (num_flowers // 3)*price_3_flowers remaining_flowers = num_flowers % 3 total_price_2_flowers = (remaining_flowers//2)* price_2_flowers money_saved = (num_flowers // 3) * price_3_flowers + (num_flowers % 3) * price_2_flowers - num_flowers * price_2_flowers answer = abs(money_saved)</pre> </p> <p>Extract the Answer: 3.0 (X) Code 1</p> <p>Instruct the Iteration (Refresh):</p>	<p>Iteration 1 3.0 (X)</p>

The error lies in both code segments not correctly implementing the meta-thought's guidance to compare the total costs of each purchasing option, ultimately resulting in incorrect calculations of Vincent's savings. Decision: Refreshing the response.

Refresh

```

num_flowers = 18
price_3_flowers = 2.5
price_2_flowers = 1
cost_using_3_flower_packages = (num_flowers / 3) * price_3_flowers + (num_flowers
mod 3) * price_2_flowers
cost_using_2_flower_packages = (num_flowers / 2) * price_2_flowers + (num_flowers
mod 2) * (price_2_flowers / 2)
money_saved = cost_using_2_flower_packages - cost_using_3_flower_packages
answer = money_saved

```

Extract the Answer: 6.0 (✓) Code 2

Instruct the Iteration (Select):

COT 2 aligns better with the meta-thought of accurately calculating total costs for each buying strategy. Code 2 indicates a greater benefit from selecting the optimal option, ensuring Vincent maximizes his savings on flowers. Better Code after comparison: Code 2

:
Iteration 2
6.0 (✓)

Table 14: An illustrative example from GSM8K demonstrating how IoRT generates a refresh instruction when the results before and after reflection are consistent.