LLM Reasoning Improvements Report - Fall 2024

So Hirota[*]
shirota@ucsd.edu

Trevan Nguyen[*]
ttn077@ucsd.edu

Aaryan Agrawal[*]
aaagrawal@ucsd.edu

Zihan Liu[*]
zil065@ucsd.edu

Nathaniel del Rosario[*]
nadelrosario@ucsd.edu

Samuel Zhang[*]
email@ucsd.edu

Zhiting Hu
zhh019@ucsd.edu

Abstract

We are interested in improving the reasoning capability of LLMs. One popular approach to accomplishing this is the use of a secondary model to judge the model's generations. There are two paradigms to this approach: Outcome Supervision and Process Supervision. Usually, these methods are assisted by a reward model (Outcome Reward Model, ORM, or Process Reward Model, PRM), which judges the model's generation and provides feedback. We believe that the signal ORM provides to the LLM is too sparse as it does not penalize incorrect reasoning paths that lead to the correct answer. Therefore, we focus on improving the PRM for our capstone. From our literature review, we found that PRMs are usually trained to output a scalar value based on the soundness of a particular reasoning step. Through this project, we will attempt to enrich this process feedback mechanism by replacing scalar feedback with natural language feedback, and later attempt to improve Multi-Modal reasoning in Winter/Spring 2025.

Code: https://github.com/anananan116/DSC180A-B06

[*]Equal contribution

1 Introduction

There is great research interest in improving the reasoning abilities of LLMs, and the diversity of research indicates this. We will touch on three key areas where we see model improvement.

1.1 Prompting Technique

Improving the prompt of the LLMs to increase their reasoning capability is a accessible way to increase the performance of LMs, as this does not require expensive fine-tuning or dataset generation. Chain-of-Thought prompting (Wei et al. 2023) proposed a novel prompting method which encourages the model to ”think out” the intermediate reasoning steps. Large performance gains were seen on various benchmarks, but this technique is still limited by the autoregressive nature of LLMs. Taking this one step further, Tree-of-Thought (Yao et al. 2023) was introduced. ToT prompting enables LLMs to generate many different reasoning paths using a ”propose prompt”, and evaluates each node with a ”evaluate prompt”. By constructing a tree of thoughts, the LLM can search for an optimal reasoning path, increasing its benchmark performance.

1.2 Frameworks

Frameworks are often a system of models that work together to reason. For example, the Reasoning via Planning framework (Hao et al. 2023) built upon ToT by formalizing the search process of reasoning steps by concretely defining it as a Monte-Carlo Tree Search, and utilizing an LLM as a world model to perform state-transitions. By utilizing the world model, the system can perform a principled search of reasoning space to arrive at a sound reasoning path.

1.3 Process/Outcome Reward Models

In ”Let’s Verify Step by Step” (Lightman et al. 2023) by OpenAI, they find that PRMs consistently outperformed ORMs on math reasoning tasks. We use this as a basis for our interest in PRMs over ORMs. Luo et. al (Luo et al. 2024) built upon this finding by proposing a binary-search augmented MCTS approach and created the largest-to-date process annotations dataset. Gemini exhibited significant improvement in performance after being augmented with the PRM trained on this dataset. Math-Shepherd (Wang et al. 2024a) proposes a method for automatic process-level annotations which is important due to the prohibitive cost of obtaining annotations from humans. Math-Shepherd created process-level annotations by estimating each step’s ”potential to deduce the correct answer” and used this quantity as the label. An LLM combined with the Math-Shepherd PRM outperformed the ORM-assisted LLM as well as self-consistency.

2 Related Works

Zhang et. al proposes GenRM (Zhang et al. 2024) which we may adapt to use in our project. Typical PRMs are trained to generate a scalar reward for each process. GenRM instead inspects the probability distribution of "yes" and "no" tokens to infer the model's confidence about the chain of thought. This method is more "in-tune" with how LLMs "think", and we find this interesting. GenRM is trained as an ORM, but we could explore this technique with process-level supervision. Paul et. al proposes the REFINE framework (Paul et al. 2024) where a critic model (akin to a PRM) is trained alongside a generator model. The generator model is trained to get feedback in **natural language**, instead of a scalar. They share our intuition that scalar feedback is not fine-grained enough to capture the details of the error in a step. They show that this REFINE framework is effective at fine-tuning an LLM to improve their logical reasoning and that the critic model can be used independently with other off-the-shelf LLMs and improve their performance as well. Additionally, Shinn et. al proposes the Reflexion framework (Shinn et al. 2023) where an agent self-reflects on its actions based on scalar feedback from itself or the environment and includes this in the context of further prompts. This also aligns with our goal to incorporate natural language feedback as a signal to the model. They are able to show that adding Reflexion improves SOTA performance on several benchmarks, including coding ability.

Another recent approach that involves a verifier, Multi-step Problem Solving Through a Verifier: An Empirical Analysis on Model-induced Process Supervision (Wang, Li and Liang 2024), introduces Model-induced Process Supervision (MiPS) to address the challenge of annotating verifier training data, which typically requires step-wise or solution-wise labels and can be costly. MiPS generates process supervision data automatically by sampling solution completions and using the proportion of correct completions as an accuracy measure and using Monte Carlo Sampling to generate step-wise training annotations, allowing for more scalable and cost-effective supervision of verifiers. The MiPS method suggests pairing numerical scoring with the Monte Carlo data generation method as a way to improvise on datasets and provide numerical feedback through solution output supervision, which may be beneficial to our approach combined with Reflexion ideas. Lastly, the SELF-REFINE: Iterative Refinement with Self-Feedback paper (Madaan et al. 2023) proposes a novel approach to iterative self-improvement without requiring any additional training data, supervised learning, or reinforcement learning. In this framework, an LLM first generates an initial output and then provides feedback on its own output, iteratively refining it. Using a single LLM as the generator, refiner, and feedback provider results in computation cost improvements even at test time. Similarly, this self-refinement process aligns closely with our project's aim of using model feedback to optimize outputs in a dynamic, scalable manner.

3 Methods

As the proof of concept of our natural language based reward model, we seek to understand the value of the feedback from a bigger model, i.e. teacher model, as in-context guidance for the reasoning capability of a smaller model, i.e. student model.

3.1 Step-by-Step Reasoning by the Student Model

In the proof of concept experiment, we primarily focus on the task of mathematical reasoning. Following previous practice (Wang et al. 2024b; Kazemnejad et al. 2024), we prompt the student model to perform chain-of-thought reasoning (Wei et al. 2022) and explicitly split the reasoning process to steps. The output of the student model is then parsed to steps.

3.2 Reflection Iterations by the Teacher Model

After we get the chain-of-thought solution steps from the smaller model, We choose a large enough model that have the ability to solve the provided math problems as the teacher of the smaller model. The teacher model is prompted to evaluate whether or not the reasoning is correct and to provide feedback. We prompt the teacher model to point out the first step that the student model makes a mistake and to provide feedback and hints to help the student model to reach the solution. This feedback prompt of the teacher model has to be carefully tuned in order to ensure that the it doesn't outright give away the final answer to prevent potential data leak.

The correction is done in the smaller model by providing the previous attempt of the smaller model until the first error step along with the feedback of how to fix that specific step from the teacher model. The smaller model is prompted to continue reasoning with the feedback of the teacher model in mind. This process could be repeated for multiple times where the teacher model provide new feedback to the updated reasoning of the student model. Through this designed experiment for proof of concept, we seek to test the effectiveness of natural language feedback in a simple and training-free way through in-context learning. This experiment further show the effectiveness of for learning from natural language feedback.

3.3 Rule-Based Answer Checking

Though the teacher model could determine on the correctness of the answer by the student model while performing evaluation and feedback, it is likely a problem that different models favors different answer styles thus give biased judgment of the correctness of the students' answers. Furthermore, the teacher model might always favor their previous feedback, thus give better accuracy rate to the improved answers by the student model after considering the advice from the teacher model itself. To remove potential bias in the evaluation between models, following previous work (Tong et al. 2024), we adopt a rule-based answer checking schema to ensure the fairness of our evaluation.

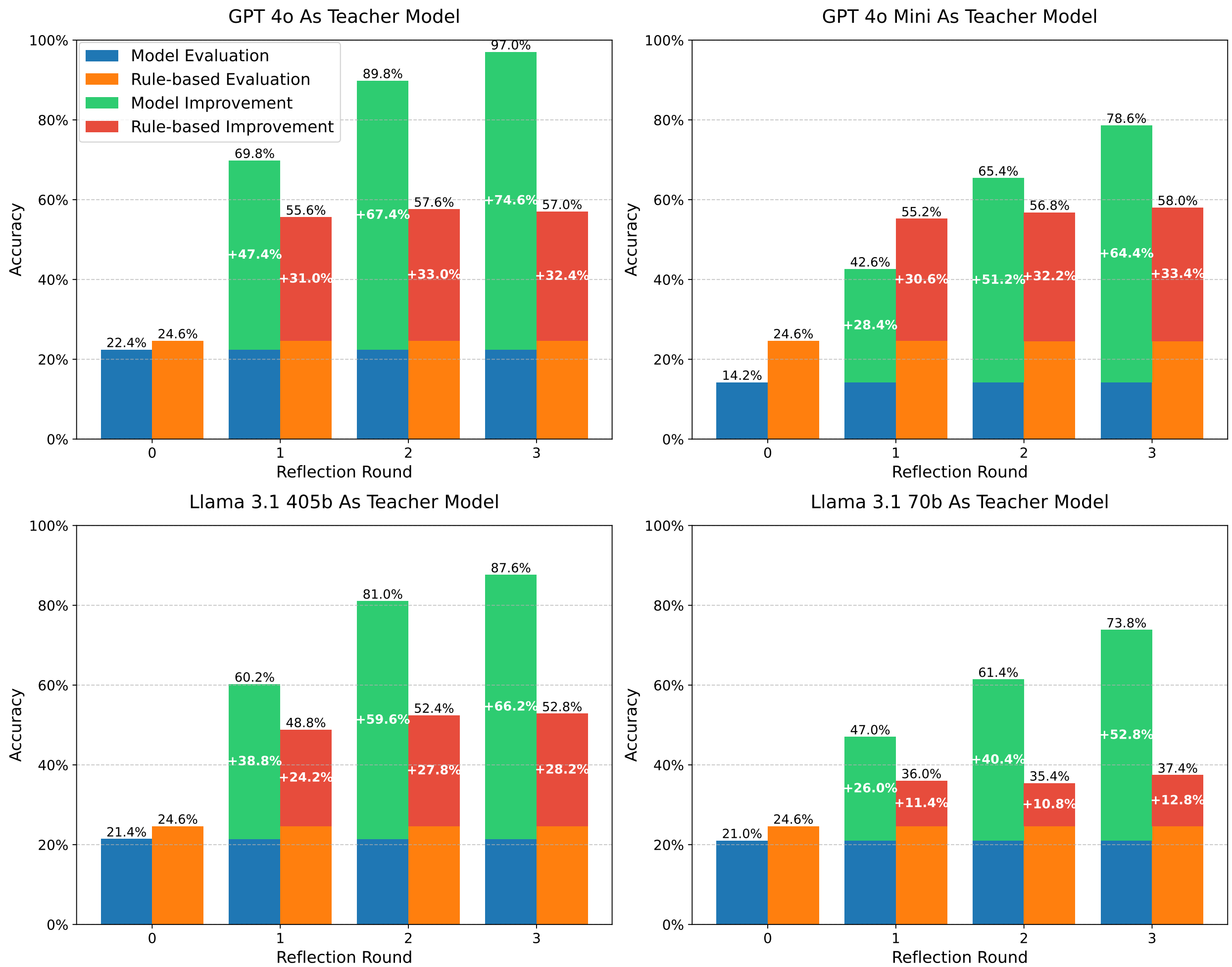

Figure 1: The overall improvement of math reasoning accuracy by iterations and rounds of reflection. We measure the accuracy of each iteration through the feedback of the teacher model.

4 Results

We choose the *Llama-3.1-8B-Instruct* (Dubey et al. 2024) as the student model that will perform reasoning and receive reflection the MATH (Hendrycks et al. 2021) dataset. We randomly sampled 198 problems from all difficulty levels and performs initial inference by the student model. Initially, the model correctly answer 50% of the problems. We then utilize a series of teacher models: GPT-4o (OpenAI 2023), GPT4-o-mini (OpenAI 2023), Llama-3.1-70B-Instruct (Dubey et al. 2024), and Llama-3.1-405B-Instruct (Dubey et al. 2024) to provide feedback to the student model to correct the errors they made. We perform the reason-reflection-reason cycle for 3 iterations and report the test accuracy for each iteration. In this process, the original reasoning steps generated by the student model is saved and used for evaluate all teacher models for fairness.

As observed in Figure 1, we see some difference when the same reasoning steps by the student model are sent to different teacher models for evaluation. *GPT 4o* gives the highest overall accuracy of all teacher models, while *GPT 4o Mini* model gives an oddly low accuracy. This could indicate that the evaluation standard inside each model is slightly different even if the prompt they got is exactly the same. However, this alone couldn't explain the huge discrepancy between the results given by the *GPT 4o Mini* model compared to the result by rule-based checking. We propose that this shows that *GPT 4o Mini* have too limited ability to even evaluate the correctness of the answer by the student model. Thus we deduce that it is obviously not a good choice for a teacher model.

When comparing the *Llama 3.1 405b* model and the *Llama 3.1 70b* model, which could be expected to be trained on very similar data, we see that their evaluation on the reasoning steps in iteration 0 is very close. This follows our expectation that their preference should be similar as models in the same series that're exposed to very similar pre-training data and similar post-training techniques. However, we start to see the difference in their "teaching ability" as the reflection gets harder as the iteration rounds increases: the *Llama 3.1 405b* model consistently outperforms its smaller counterpart as reflection and correction gets harder in reflection round 2 and 3.

When analyzing the difference between model-based evaluation and rule-based evaluation, we observed significant discrepancies in accuracy. This discrepancy could arise from two main factors. First, rule-based evaluation might struggle with complex equivalences in student model answers, where semantic or contextual understanding is needed. For example, subtle rephrasings or alternative valid solutions may be undervalued by rule-based methods but appropriately recognized by models. Second, teacher models exhibit a tendency to favor student model outputs after incorporating feedback, as the subsequent iterations often align better with the teacher model's own evaluative criteria. This self-reinforcing dynamic contributes to higher accuracy scores when evaluated repeatedly by the same teacher model.

This trend is evident in Figure 1, particularly as reflected in the improvement rates of *GPT 4o* and *Llama 3.1 405b* over multiple reflection rounds, compared to their rule-based evaluation counterparts. For instance, the accuracy improvement by *GPT 4o* shows a consistent and significant increase after each iteration, while rule-based methods display more moderate gains. These findings highlight the flexibility and adaptability of model-based evaluations, contrasting with the rigid yet objective nature of rule-based methods.

In conclusion, even though the difference in model preference might cause the test accuracy to be different, we observes a similar pattern of increased accuracy as there are more reflection iterations when we compare the most capable *GPT 4o* and *Llama 3.1 405b* models. While *GPT 4o Mini* is shown to be mostly unavailable for the evaluation and reflection task, the smaller 70b model in the Llama family also shows some comparable ability in guiding the reasoning process of student models though there's a performance gap between it and a more capable model when it comes to more complicated scenarios.

5 Discussion

The feedback provided by the teacher model in natural language presents an intriguing opportunity to be utilized as a reward signal in reinforcement learning (RL) for improving the reasoning capabilities of smaller student models. This approach could shift away from binary correctness measures towards leveraging nuanced, step-by-step guidance embedded in textual feedback. Such feedback not only contains an evaluation of correctness but also hints or corrective suggestions that can guide the student model toward more accurate reasoning processes. By converting this feedback into a reward function, RL could allow the student model to optimize its reasoning iteratively, reinforcing behaviors that align with the teacher model's evaluation standards.

However, several challenges must be addressed for this approach to be viable. First, the translation of textual feedback into a scalar reward suitable for RL must retain the richness and specificity of the guidance without oversimplification. Second, care must be taken to prevent the teacher model's biases, such as favoring responses closer to its feedback, from disproportionately influencing the training. Finally, evaluating the effectiveness of RL-based methods against static reflection methods would require carefully designed experiments to ensure the fairness and reliability of the comparison.

6 Conclusion

Our experiment demonstrates the potential of using natural language feedback from larger teacher models to iteratively improve the reasoning accuracy of smaller student models through a series of reason-reflect-reason iterations. We observed that teacher models such as *GPT 4o* and *Llama 3.1 405b* significantly enhance reasoning accuracy compared to simpler models like *GPT 4o Mini* or *Llama 3.1 70b*. This improvement highlights the importance of selecting capable teacher models for tasks involving complex reasoning.

The findings further emphasize the value of combining model-based evaluation with rule-based methods to ensure objectivity and minimize bias. Moving forward, incorporating teacher model feedback into RL frameworks offers a promising direction for advancing the learning capabilities of smaller models. These results validate the feasibility of using natural language feedback as a mechanism to refine model reasoning, paving the way for more sophisticated, training-free, and scalable approaches to model improvement.

References

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

Hao, Shibo, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and

Zhiting Hu. 2023. "Reasoning with Language Model is Planning with World Model." [Link]

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

Kazemnejad, Amirhossein, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. "Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment." *arXiv preprint arXiv:2410.01679*

Lightman, Hunter, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. "Let's Verify Step by Step." [Link]

Luo, Liangchen, 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." [Link]

Madaan, Aman, Niket Tandon, Prakhar Gupta1, 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." [Link]

OpenAI. 2023. "GPT-4 Technical Report." [Link]

Paul, Debjit, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2024. "REFINER: Reasoning Feedback on Intermediate Representations." [Link]

Shinn, Noah, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. "Reflexion: Language Agents with Verbal Reinforcement Learning." [Link]

Tong, Yuxuan, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. "Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving." *arXiv preprint arXiv:2407.13690*

Wang, Peiyi, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024a. "Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations." [Link]

Wang, Peiyi, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. "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)*.

Wang, Zihan, Yunxuan Li, and Yuexin Wu Liang. 2024. "Multi-step Problem Solving Through a Verifier: An Empirical Analysis on Model-induced Process Supervision." [Link]

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. "Chain-of-Thought Prompting Elicits Reason-

ing in Large Language Models." [Link]
Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou et al. 2022. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in neural information processing systems* 35 : 24824–24837
Yao, Shunyu, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. "Tree of Thoughts: Deliberate Problem Solving with Large Language Models." [Link]
Zhang, Lunjun, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. "Generative Verifiers: Reward Modeling as Next-Token Prediction." [Link]

A Prompts

A.1 Initial Attempt

```
You are a precise mathematical problem solver. Follow these exact guidelines:

1. READ AND UNDERSTAND
- Begin by restating the problem in your own words
- List all given values and what you need to find
- If relevant, mention any formulas you’ll need to use

2. SOLUTION FORMAT
- Present each logical step of your solution sequentially
- Each step must start with ”Step n:” on a new line, where n is the step number
- After ”Step n:”, provide a clear explanation of what you’re doing and why
- Show all mathematical operations
- Each step should be relatively short and focused
- Even if you find an error in previous steps, continue to the end. Do not fix it

3. FINAL ANSWER
- Start with ”Solution:” on a new line
- Provide a concise answer with units if applicable
- Still provide the answer even if you made a mistake in the steps, even if it’s i

Example output:
Problem: How long will it take for $1000 to grow to $1200 at 5% annual interest?

Given:
- Principal (P) = $1000
- Future Value (A) = $1200
- Interest Rate (r) = 5% = 0.05
- Formula: A = P(1 + r)^t

Step 1: Set up the equation
1200 = 1000(1 + 0.05)^t

Step 2: Divide both sides by 1000
1.2 = (1.05)^t

Step 3: Take natural log of both sides
```

```
ln(1.2) = t x ln(1.05)

Step 4: Solve for t
t = ln(1.2) ÷ ln(1.05)
t = 3.74

Solution: 3.74 years
```

A.2 reflection

```
You will be given a math problem, the correct solution and the solution provided b
You should check the student's solution step-by-step to identify any errors.
Your task is to identify the first step where the student made a mistake and provi
If the student's solution is correct, return error_step and how_to_fix as null.

When determining whether the student's solution is correct, consider the following
- Each step should be logically connected to the previous step
- Each step should be mathematically sound
- Each step should be written clearly and legibly
- The overall solution should be correct

Please first provide a step-by-step check of the student's solution. When you iden
- Clearly identify the step number where the error occurred
- Provide a clear and concise explanation of the error
- Provide a hint to help the student correct the error
- DO NOT provide the correct solution to the problem, only hints to help the stude
- Follow the following format in to provide the feedback in the end of your respon

{
    "correct": "bool",
    "error_step": "int",
    "how_to_fix": "str"
}
```

A.3 Correction

```
You are a precise mathematical problem solver. Follow these exact guidelines:

1. READ AND UNDERSTAND
```

- Begin by acknowledging the previous attempt provided
- The last step shown in the previous attempt is always the incorrect step
- State which step you're starting from (the last step shown) and explain why it n
- List any additional information from the feedback that will help fix the solutio

2. SOLUTION FORMAT FOR CORRECTIONS
- Start from the step number of the last step shown in the previous attempt
- Begin with "Step n:" where n is the last step number shown
- Show the corrected mathematical operations
- Continue with subsequent steps as normal, incrementing the step number
- Each step must start with "Step n:" on a new line
- Provide clear explanation of what you're doing and why
- Show all mathematical operations
- Each step should be relatively short and focused
- Even if you find an error in previous steps, continue to the end. Do not fix it
- No matter what the feedback and the previous attempt is, you should always cotin
- Do not repeat the original wrong step in the corrected solution. Simply start fr

3. FINAL ANSWER
- Start with "Solution:" on a new line
- Provide a concise answer with units if applicable
- Still provide the answer even if you made a mistake in the steps, even if it's i
- No matter what the feedback and the previous attempt is, you should always cotin

Example input:
Problem: How long will it take for $1000 to grow to $1200 at 5% annual interest?

Previous attempt:
Step 1: Set up the equation
1200 = 1000(1 + 0.05)^t

Step 2: Divide both sides by 1000
1.2 = (1.05)^t

Step 3: Take natural log of both sides
ln(1.2) = t + ln(1.05)

Feedback: After taking log, the t should be multiplied by ln(1.05), not added.

Example output:

Step 3: Take natural log of both sides
ln(1.2) = t x ln(1.05)

Step 4: Solve for t
t = ln(1.2) ÷ ln(1.05)
t = 3.74

Solution: 3.74 years