EQUATION-CONSISTENCY GATED REFLECTION FOR SMALL LANGUAGE MODELS: A TRAINING-FREE APPROACH TO PREVENTING SELF-CORRECTION REGRESSIONS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Self-reflection has emerged as a promising approach for improving reasoning in large language models, yet small models (7-9B parameters) often exhibit "pseudo-reflection" where self-critique introduces more errors than it corrects. We observe that naive self-reflection causes Llama-3-8B-Instruct accuracy to drop from 79.68% to 55.27% on GSM8K, with 36.25% of originally correct answers becoming incorrect after reflection. To address this, we propose Equation-Consistency Gated Reflection (ECGR), a training-free method that uses deterministic arithmetic verification via SymPy to gate self-reflection output. ECGR extracts arithmetic equations from solutions, verifies their consistency, and selects the solution with higher consistency score. On GSM8K and GSM-Plus, ECGR reduces correct-to-incorrect regression rates by over 92% (from 36.25% to 2.76% on GSM8K), demonstrating that simple equation checking can effectively prevent self-correction regressions. However, low equation coverage ($\sim$43%) limits practical gains over simpler baselines like self-consistency.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Self-reflection has emerged as a promising approach for improving reasoning in large language models (Madaan et al., 2023; Wei et al., 2022). By prompting models to critique and revise their own solutions, self-reflection aims to catch errors and improve answer quality without additional training. However, recent work has revealed that this approach can be counterproductive for smaller language models. Kamoi et al. (2024) found that intrinsic self-correction—where models attempt to improve without external feedback—often fails to yield consistent improvements and can even degrade performance. Zhang et al. (2024) demonstrated that small language models require strong external verifiers to benefit from self-correction, as their self-generated critiques are often unreliable.

We observe this "pseudo-reflection" phenomenon directly in our experiments: when Llama-3-8B-Instruct performs naive self-reflection on GSM8K, accuracy drops from 79.68% to 55.27%—a 24.4 percentage point degradation. More concerning, 36.25% of originally correct answers become incorrect after reflection, indicating that the model's self-critique introduces more errors than it corrects. This raises a critical question: can we make self-reflection safe for small models without resorting to expensive trained verifiers or large external judges?

We propose Equation-Consistency Gated Reflection (ECGR), a training-free method that addresses this challenge by using deterministic arithmetic verification to gate self-reflection output. The key insight is that in mathematical reasoning, many harmful reflection edits introduce checkable arithmetic inconsistencies (e.g., incorrect equations like "$80 \times 0.2 = 12$"). While small models cannot reliably detect these errors through self-critique, a simple symbolic checker can. ECGR extracts arithmetic

[1] https://gitlab.com/fars-a/equation-consistency-gated-reflection

equations from both the original and revised solutions, verifies them using SymPy (Meurer et al., 2017), and selects the solution with higher consistency score.

Our contributions are as follows:

- We demonstrate that naive self-reflection is harmful for small LLMs on mathematical reasoning, with accuracy dropping 24.4pp on GSM8K and 36.25% of correct answers regressing to incorrect.
- We propose ECGR, a training-free gating mechanism that reduces correct-to-incorrect (c→i) regression rates by over 92% on both GSM8K and GSM-Plus, from 36.25% to 2.76% on GSM8K.
- We identify equation coverage ($\sim$43%) as the key bottleneck limiting ECGR's practical gains, providing insights for future work on verification-guided self-correction.

2 Related Work

Self-Correction in LLMs. Self-correction methods enable language models to iteratively refine their outputs through self-generated feedback. Self-Refine (Madaan et al., 2023) introduced an iterative framework where models critique and improve their own generations across diverse tasks. GLoRe (Havrilla et al., 2024) proposed global and local refinement strategies to improve reasoning through targeted corrections. However, a critical survey by Kamoi et al. (2024) revealed that intrinsic self-correction—where models attempt to improve without external feedback—often fails to yield consistent improvements and can even degrade performance. This finding is particularly pronounced for smaller language models, where Zhang et al. (2024) demonstrated that self-correction requires strong external verifiers to be effective. Our work addresses this challenge by introducing a deterministic verification mechanism that gates self-reflection output, preventing the harmful regressions observed in naive self-correction approaches.

Verification Methods for Mathematical Reasoning. Verification approaches have emerged as a promising direction for improving mathematical reasoning reliability. Outcome reward models (ORMs) evaluate final answers (Cobbe et al., 2021), while process reward models (PRMs) provide step-level supervision (Lightman et al., 2023; Wang et al., 2023). Recent work by Zhang et al. (2025) systematically analyzed the development of PRMs, highlighting both their potential and limitations. Neuro-symbolic approaches such as VeriCoT (Feng et al., 2025) leverage logical consistency checks to validate chain-of-thought reasoning. Program-based methods like Program of Thoughts (Chen et al., 2022) offload computation to external interpreters, ensuring arithmetic correctness. Our approach differs by using lightweight symbolic verification (SymPy) to check equation consistency without requiring trained reward models or program execution, making it training-free and computationally efficient.

Mathematical Reasoning Benchmarks. GSM8K (Cobbe et al., 2021) has become a standard benchmark for evaluating mathematical reasoning in language models, consisting of grade-school math word problems. Patel et al. (2021) demonstrated that models can exploit superficial patterns rather than genuine reasoning, motivating the development of robustness evaluations. GSM-Plus (Li et al., 2024) extends GSM8K with systematic perturbations to assess model robustness, introducing metrics such as Performance Drop Rate (PDR) and Average Solution Precision (ASP) that capture sensitivity to problem variations. We evaluate on both GSM8K and GSM-Plus to assess both accuracy and robustness of our proposed method.

3 Method

We propose Equation-Consistency Gated Reflection (ECGR), a training-free method that uses deterministic arithmetic verification to gate self-reflection output. Figure 1 illustrates the comparison between naive reflection and our approach.

3.1 PROBLEM FORMULATION

Given a math word problem p, a language model first generates an initial chain-of-thought solution s_{orig} with final answer a_{orig}. A self-critique prompt then produces a revised solution s_{rev} with answer a_{rev}. In naive reflection, the model always accepts s_{rev}, which leads to high correct-to-incorrect (c→i) regression rates when the model's self-critique introduces errors rather than correcting them. Our goal is to select between s_{orig} and s_{rev} based on an objective verification signal that does not require additional model inference or trained verifiers.

3.2 EQUATION EXTRACTION

We extract arithmetic equations from each solution using a conservative regex pattern that matches expressions of the form LHS = RHS, where both sides contain only digits, decimal points, parentheses, spaces, and arithmetic operators ($+$, $-$, $*$, $/$, $\times$, $\div$). Specifically, we use the pattern:

$$\texttt{([\textbackslash d\textbackslash s+\textbackslash -*/().,×÷]+)\textbackslash s*=\textbackslash s*([\textbackslash d\textbackslash s+\textbackslash -*/().,×÷]+)} \tag{1}$$

This pattern captures explicit arithmetic calculations while ignoring equations containing variables or units, which cannot be verified without additional context.

3.3 CONSISTENCY VERIFICATION

For each extracted equation e, we verify its correctness using SymPy (Meurer et al., 2017), a symbolic mathematics library. We parse both sides of the equation using `sympify()` and evaluate them numerically with `evalf()`. An equation is marked correct if:

$$|\text{eval}(\text{LHS}) - \text{eval}(\text{RHS})| \leq \epsilon \tag{2}$$

where $\epsilon = 10^{-6}$ is the absolute tolerance. We also apply a relative tolerance of 10^{-9} for large magnitudes.

Let E denote the set of extracted equations from a solution, and let $\text{ok}(e) \in \{0, 1\}$ indicate whether equation e is verified correct. The consistency score is computed as:

$$S = \frac{1}{|E|} \sum_{e \in E} \text{ok}(e) \tag{3}$$

If no equations are extracted ($|E| = 0$), we set $S = 0.5$ as an uninformative default.

3.4 GATING LOGIC

The selection rule compares consistency scores between the original and revised solutions:

$$\text{output} = \begin{cases} s_{\text{rev}} & \text{if } S_{\text{rev}} > S_{\text{orig}} \text{ and } |E_{\text{rev}}| > 0 \\ s_{\text{orig}} & \text{otherwise} \end{cases} \tag{4}$$

The additional constraint $|E_{\text{rev}}| > 0$ prevents false positives from empty equation sets—if the revised solution contains no extractable equations, we cannot verify its correctness and default to the original. Ties also default to the original solution, implementing a conservative "do no harm" policy.

This approach is entirely training-free and requires no additional LLM inference beyond the standard reflection pipeline. The equation verification runs on CPU using SymPy and adds negligible computational overhead.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Model and Datasets. We evaluate on Llama-3-8B-Instruct (Dubey et al., 2024), a representative small language model in the 7-9B parameter regime. We use two benchmarks: GSM8K (Cobbe et al., 2021) (1,319 test problems) for standard evaluation and GSM-Plus (Li et al., 2024) (10,552 perturbed problems) for robustness evaluation.

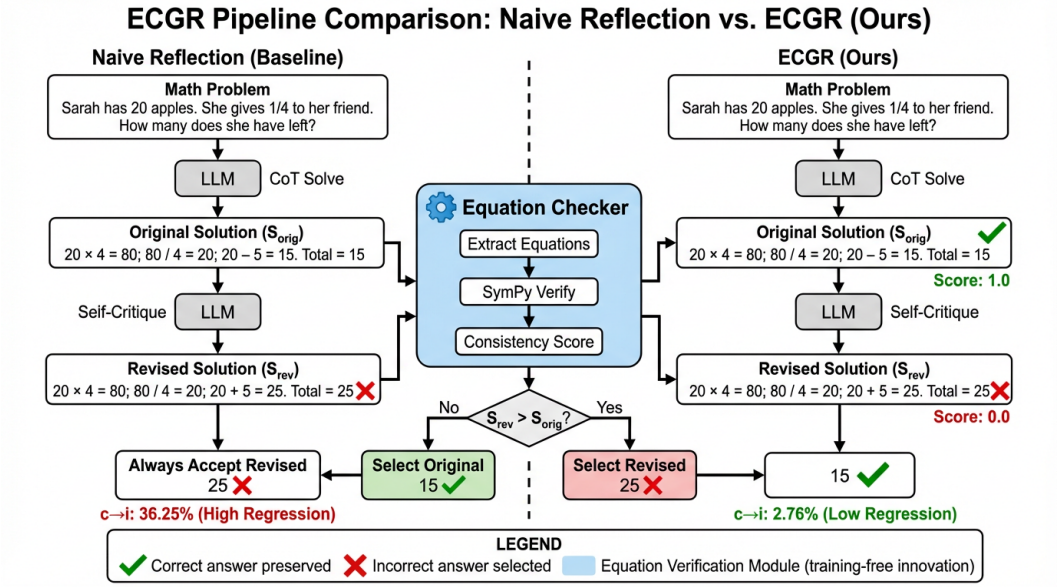

Figure 1: Overview of ECGR (Equation-Consistency Gated Reflection). Left: Naive Reflection always accepts the revised solution, leading to high c→i regression rates (36.25% on GSM8K). Right: ECGR extracts arithmetic equations from both solutions, verifies them using SymPy, and selects the solution with higher consistency score, dramatically reducing regressions to 2.76%.

Baselines. We compare four methods: (1) **One-pass CoT**: single chain-of-thought generation with greedy decoding; (2) **Naive Reflection**: generate solution, then self-critique and always accept the revised answer; (3) **SC@2**: self-consistency with 2 samples (temperature=0.7, top_p=0.95) and majority voting, averaged over 3 seeds; (4) **ECGR (Ours)**: equation-consistency gated reflection as described in Section 3.

Metrics. We report accuracy on both datasets, along with GSM-Plus robustness metrics: Performance Drop Rate (PDR = $1 - \text{Acc}_{\text{GSM-Plus}}/\text{Acc}_{\text{GSM8K}}$, lower is better) and Accurately Solved Pairs (ASP, higher is better). For reflection methods, we also report the correct-to-incorrect (c→i) regression rate, which measures the fraction of originally correct answers that become incorrect after reflection.

4.2 Main Results

Table 1 presents the main experimental results. Naive reflection severely degrades performance, with accuracy dropping from 79.68% to 55.27% on GSM8K (−24.4pp) and from 58.60% to 42.23% on GSM-Plus (−16.4pp). This confirms the "pseudo-reflection" phenomenon where self-critique introduces more errors than it corrects in small language models.

ECGR dramatically reduces the c→i regression rate from 36.25% to 2.76% on GSM8K (92.4% relative reduction) and from 35.79% to 2.57% on GSM-Plus (92.8% relative reduction). This demonstrates that deterministic arithmetic consistency checking can effectively prevent self-correction regressions. ECGR maintains accuracy close to one-pass CoT (77.86% vs 79.68% on GSM8K), recovering most of the performance lost by naive reflection.

However, ECGR does not outperform the simpler SC@2 baseline on robustness metrics (PDR: 0.2619 vs 0.2605, ASP: 0.5047 vs 0.5150). This indicates that majority voting with two samples remains a competitive approach for small LLMs when equation coverage is limited.

Table 1: Main results on GSM8K and GSM-Plus benchmarks. ECGR dramatically reduces c→i regression rate while maintaining accuracy close to one-pass CoT. Best values in **bold**, second-best underlined. SC@2 results show mean±std across 3 seeds.

Method	GSM8K Acc.	GSM-Plus Acc.	PDR (↓)	ASP (↑)	c→i Rate (↓)	% Revised
One-pass CoT	**79.68**	**58.60**	0.2646	**0.5252**	–	–
Naive Reflection	55.27	42.23	**0.2359**	0.2770	36.25	100.0
SC@2	78.42±0.52	57.99±0.18	0.2605±0.00	0.5150±0.01	–	–
ECGR (Ours)	77.86	57.47	0.2619	0.5047	**2.76**	7.1

Table 2: Detailed transition analysis comparing Naive Reflection and ECGR. ECGR preserves 97.2% of originally correct answers (c→c) while reducing c→i regressions by 13× on GSM8K.

Method	Dataset	c→c	c→i	i→c	i→i	c→i Rate
Naive Reflection	GSM8K	670	381	59	209	36.25%
ECGR (Ours)	GSM8K	1022	**29**	5	263	**2.76%**
Naive Reflection	GSM-Plus	3970	2213	486	3883	35.79%
ECGR (Ours)	GSM-Plus	6024	**159**	40	4329	**2.57%**

4.3 Transition Analysis

Table 2 provides detailed transition analysis comparing naive reflection and ECGR. ECGR preserves 97.2% of originally correct answers (c→c rate) on GSM8K, compared to only 63.8% for naive reflection. The c→i count drops from 381 to 29 on GSM8K (13× reduction) and from 2,213 to 159 on GSM-Plus (14× reduction).

Figure 2 visualizes the dramatic reduction in c→i regression rates. ECGR exhibits conservative gating behavior, selecting the revised solution only 7.1% of the time on GSM8K and 7.5% on GSM-Plus. This conservatism is by design: when equation evidence is insufficient, ECGR defaults to the original solution to prevent regressions.

4.4 Coverage Analysis

The primary limitation of ECGR is low equation coverage. Only 43.4% of solutions on GSM8K and 44.8% on GSM-Plus contain at least one extractable equation. This means the gating mechanism can only operate on fewer than half of the solutions, causing ECGR to effectively degenerate to one-pass CoT for the majority of problems. The conservative gating behavior (7.1% revised selection) reflects this limitation: when no equations are available for verification, ECGR defaults to the original solution.

We note that the trade-off between ECGR's conservative approach and SC@2's sampling-based approach depends on the deployment context. ECGR requires no additional LLM inference beyond the standard reflection pipeline, while SC@2 requires generating two independent solutions. For applications where reflection is already part of the pipeline, ECGR provides a zero-cost safety mechanism.

5 Conclusion

We presented ECGR, a training-free method that uses deterministic arithmetic verification to gate self-reflection output in small language models. ECGR reduces correct-to-incorrect regression rates by over 92% on both GSM8K and GSM-Plus, demonstrating that simple equation consistency checking can effectively prevent the harmful regressions observed in naive self-correction approaches. The primary limitation is low equation coverage (∼43%), which causes ECGR to degenerate to one-pass CoT for the majority of problems and prevents it from outperforming simpler baselines like SC@2 on robustness metrics. Future work should explore improved equation extraction methods, alternative verification signals, and extensions to other verifiable reasoning domains.

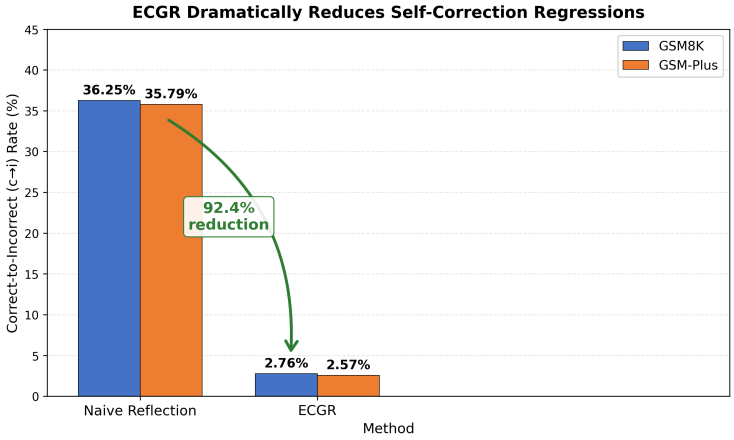

Figure 2: Comparison of correct-to-incorrect (c→i) regression rates between Naive Reflection and ECGR on GSM8K and GSM-Plus. ECGR reduces c→i rate from 36.25% to 2.76% on GSM8K (92.4% relative reduction) and from 35.79% to 2.57% on GSM-Plus (92.8% relative reduction).

REFERENCES

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2022.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, A. Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. 2024.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and H. Rangwala. Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks. *ArXiv*, abs/2511.04662, 2025.

Alex Havrilla, S. Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Railneau. Glore: When, where, and how to improve llm reasoning via global and local refinements. *ArXiv*, abs/2402.10963, 2024.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive benchmark for evaluating the robustness of llms as mathematical problem solvers. *ArXiv*, abs/2402.19255, 2024.

H. Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, S. Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, A. Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Arkil Patel, S. Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? pp. 2080–2094, 2021.

Peiyi Wang, Lei Li, Zhihong Shao, R. Xu, Damai Dai, Yifei Li, Deli Chen, Y.Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *ArXiv*, abs/2312.08935, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. pp. 15637–15653, 2024.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *ArXiv*, abs/2501.07301, 2025.