Learning to Self-Correct through Chain-of-Thought Verification

Bradley Guo [1] **Jinweng Gu** [1] **Jin Peng Zhou** [1] **Wen Sun** [1]

Abstract

Self-correction in Large Language Models (LLMs) has emerged as a promising approach towards enhancing the reasoning capabilities of LLMs at inference-time. In this paper, we study how self-correction can enable an LLM to effectively perform search over multiple sequential turns on 3-SAT problems. We train a self-correcting model with reinforcement learning that verifies an initial solution through chain-of-thought reasoning and uses its own evaluation to provide a new solution. Despite being trained to self-correct once, the model can revise its answers in a sequential loop at inference-time, allowing for multi-turn gains. Our experiments demonstrate that generating strong chain-of-thought evaluations of potential solutions is essential, allowing sequential scaling through refining an initial solution over k turns to surpass even the strongest oracle-guided parallel scaling methods (i.e. pass@k).

1. Introduction

Large language models (LLMs) have demonstrated strong capabilities through pre-training across reasoning and code generation tasks (Feng et al., 2020; Li et al., 2022). A promising direction to further boost their accuracy without spending more compute during pre-training is self-correction: allowing a single model to iteratively refine its own outputs at test time (Welleck et al., 2022; Madaan et al., 2023; Qu et al., 2024; Kumar et al., 2024; Xiong et al., 2025). Instead of independently sampling answers (parallel scaling), self-correction enables the model to sequentially improve its outputs, a key skill for solving complex problems.

Studies such as Huang et al. (2024) have shown that LLMs cannot reliably detect and correct their own mistakes through naive self-correction. In practice, therefore, many self-correction approaches leverage external feedback, usually a separately fine-tuned LLM (Havrilla et al., 2024; Yang et al., 2025). While effective, these methods require multiple models at inference time and do not always outperform simple parallel sampling strategies (e.g. pass@k) when measured under the same compute budget.

To overcome this, Kumar et al. (2024) and Xiong et al. (2025) each train a single model to improve answers without external feedback via reinforcement learning, but they stop short of demonstrating multi-turn gains. While Kumar et al. (2024) achieves this solely through careful reinforcement learning training, Xiong et al. (2025) achieves this by incorporating a chain-of-thought (CoT) verification to improve its internal reward model. However, it is unclear whether the CoT allows the model to pinpoint specific errors to correct as they fail to compare against parallel methods under the same verification budget.

In this work, we reveal the strengths of self-correction by studying the 3-SAT problem. We show that the model's ability to learn extremely accurate CoT evaluations on 3-SAT problems is what enables sequential methods to scale better than parallel methods.

Summary of Contributions

- We demonstrate that a self-correction model trained for single-turn revision can generalize to multiple turns at inference time, yielding a monotonic improvement in solution accuracy.
- We show that strong CoT verification is essential to guide the sequential search of the model and to outperform the pass@k baselines.
- We provide evidence that this self-correction loop generalizes out-of-distribution, retaining its search benefits on larger 3-SAT instances than those seen during training.

2. Related Work

Neural Networks in Boolean Satisfiability. Previous works exploring the use of neural networks to solve the Boolean satisfiability (SAT) problem have either proposed

[1]Cornell University. Correspondence to: Bradley Guo <bzg4@cornell.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

architectures with built-in constraints (Selsam et al., 2019; Serrano et al., 2024; Ghanem et al., 2024), or have incorporated machine learning into traditional SAT solvers (Selsam & Bjørner, 2019; Li et al., 2024). In this work, we use the verification-generation gap of the SAT setting as a testbed to study self-correcting LLMs.

LLM Reasoning. Building on recent advances in LLM reasoning (OpenAI, 2024; DeepSeek-AI, 2025), inference-time scaling trades off compute and quality via parallel scaling which involves sampling multiple responses (Wang et al., 2023; Nguyen et al., 2024), or sequential scaling through longer CoT reasoning (Wei et al., 2023; Muennighoff et al., 2025) and iterative refinement (Welleck et al., 2022; Yao et al., 2023; Madaan et al., 2023; Qu et al., 2024; Kumar et al., 2024; Xiong et al., 2025). Recent work has also shown that training a chain-of-thought verifier improves stability over other methods such as DPO verifiers, LLM-as-a-Judge, and discriminative verifiers (Zhang et al., 2025).

Self-correction. Self-correction provides sequential scaling capabilities to LLMs, allowing them to iteratively refine their previous answers. While a variety of works have shown strong self-correction performance when the model is given external feedback (Li et al., 2025), recent work (Huang et al., 2024; Qu et al., 2024; Xiong et al., 2025) found that many intrinsic self-correction methods fail to decide whether to keep its previous answer. While methods using oracle feedback show improvements over multiple sequential turns (Qu et al., 2024), methods that seek to show self-correction behavior using a single model only show improvement over at most two turns (Xiong et al., 2025; Kumar et al., 2024).

3. Experimental Methods

The 3-SAT Problem A 3-SAT formula with n variables and m clauses can be expressed in text in DIMACS format. The DIMACS format for 3-SAT problems consists of the preamble "p cnf n m" followed by a list of m clauses, encoded as a sequence of 3 space-separated integers, terminated by a 0. For example, clause $(x_1 \vee \neg x_3 \vee x_4)$ is written as 1 -3 4 0. An assignment consists of the space-separated sequence of possibly negated integers from 1 to n, with k representing x_k and -k representing $\neg x_k$. A clause is satisfied if at least one of its literals evaluates to true under the given assignment, and an assignment satisfies the 3-SAT formula if every clause is satisfied. The 3-SAT problem in general is an NP-complete problem, which means that while verifying a proposed assignment can be done in polynomial time, finding any satisfying assignment remains NP-complete in the worst case. Our goal is to generate assignments that satisfy the given 3-SAT formula.

Self-correction Model Following the multi-turn Markov Decision Process (MDP) formulation in (Xiong et al., 2025), we isolate the effects of self-correction by training a model denoted by π which takes in a DIMACS 3-SAT problem $s \in \mathcal{S}$ and an initial assignment $a^0 \in \mathcal{A}$, and generates an evaluation y followed by a new assignment a^1:

$$(y^0, a^1) \sim \pi(\cdot|s, a^0)$$

Despite easy access to y^0 at inference-time for the 3-SAT problem, this is not always the case for more complex settings, which is why we train the model to generate y^0 itself. The model is trained in two stages:

1. The model is first trained with supervised fine-tuning (SFT) to follow the correct formatting and to learn a strong and generalizable verification algorithm.
2. Next, it is trained with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to learn how to use the verification feedback in order to provide a new assignment.

An example of the format and CoT for the verification algorithm is shown in Figure 1. For the SFT training phase, we algorithmically generate deterministic evaluations y, which simulate the polynomial-time 3-SAT verification algorithm by using CoT to check each clause one by one. We trained Qwen2.5-1.5B on 3-SAT formulas drawn from the distribution d_0 which contains a random number of variables between 3 and 22, and a random number of clauses between 1 and 8, with unsatisfiable instances discarded. The model was trained using 4000 examples for 1 epoch with learning rate 1e-3. Half of the examples had a correct initial assignment a^0 with $a^1 = a^0$, and half of the examples had an incorrect initial assignment a^0 with an unrelated correct assignment a^1 to allow the model to learn to keep correct assignments and change incorrect assignments.

For the RL training phase, we ran standard GRPO with (s, a^0) as prompts, with s drawn from d_0 and a^0 drawn at random from $\mathcal{A}$, training with 30,000 examples over 3 epochs using 2 A6000s. For the reward, we use the oracle correctness of the second assignment, $r^*(s, a^1)$ with a format penalty $r^{format}(a^1)$ and a penalty $r^{flip}(a^0, a^1)$ on the fraction of signs changed between variables in assignments a^0 and a^1 to ensure that a^0 and a^1 are related to each other. Thus, following the standard RLHF pipelines (Ouyang et al., 2022) we optimize the KL-regularized objective

$$\max_{\phi} \mathbb{E}_{s\sim d_0, a^0\sim\mathcal{A}, (y,a^1)\sim\pi_\phi^{RL}(\cdot|s,a^0)}[$$

$$r^*(s, a^1) - \eta_1 r^{format}(a^1) - \eta_2 r^{flip}(a^0, a^1)$$

$$-\beta \log(\pi_\phi^{RL}(y^0, a^1|s, a^0)/\pi^{SFT}(y, a^1|s, a^0))]$$

During inference, we repeatedly take the a^1 generated by the previous turn and use the pair (s, a^1) as input for the next

Problem:
p cnf 5 3
-5 2 3 0
-5 -3 4 0
2 -4 5 0

First Assignment: -1 2 3 -4 5

Verification:
First, let me parse the SAT formula ... Let me go through each clause one by one.
Clause 1: -5 2 3 0. Literals: -5, 2, 3. Assignment: 5 2 3. So 2, 3 are true. Clause 1 is satisfied.
Clause 2: -5 -3 4 0. Literals: -5, -3, 4. Assignment: 5 3 -4. All literals are false.
Clause 3: 2 -4 5 0. Literals: 2, -4, 5. Assignment: 2 -4 5. So 2, -4, 5 are true. Clause 3 is satisfied.
All literals in clause 2 are false. Instead, a correct assignment that satisfies every clause is

New assignment: -1 2 3 4 5

Figure 1. An example of the self-correction format for a 3-SAT formula with 5 variables and 3 clauses. Part of the verification is truncated for compactness.

Algorithm 1 Sequential inference loop

Input: 3-SAT formula s, initial assignment a^0, number of self-correction turns k, fine-tuned model π.
Output: Final assignment a^n.
for $i = 0$ **to** $k - 1$ **do**
  $(y^i, a^{i+1}) \sim \pi(\cdot|s, a^i)$
end for
return a^n

turn, repeating for k sequential turns. As a pure sequential approach, we evaluate the assignment given at the end of k turns without considering intermediate assignments. We use the same model for each turn as it is trained to keep correct assignments and modify incorrect assignments based on the last attempt. This is shown in Algorithm 1.

To ablate the effects of evaluation inaccuracy, we also provide results for the use of oracle evaluation, i.e. replacing the contents of the for loop in Algorithm 1 with

$$a^{i+1} \sim \pi(\cdot|s, a^i, y^i)$$

since $y^i(a^i)$ can be deterministically generated, and stopping when a correct solution is generated.

Parallel scaling Baseline For a fair parallel baseline, we train a model using SFT followed by GRPO which takes in a 3-SAT formula and outputs an assignment:

$$a \sim \pi(\cdot|s)$$

and this baseline using the pass@k metric. In other words, a parallel sampling of k assignments is considered correct if at least one of the assignments satisfy the 3-SAT formula.

Sequential scaling without verification Baseline To isolate the impact of having the model generate the evaluation y, we trained a separate model in the same format as the self-correction model, the difference being that the model is trained to generate a correct assignment without generating an evaluation given a 3-SAT problem and incorrect assignment:

$$a^1 \sim \pi(\cdot|s, a^0)$$

The sampling procedure is the same as the self-correction model, but unlike the self-correction model which always outputs the k^{th} solution, this baseline uses oracle rewards to select the best generation out of the sequence of k generations.

4. Results

Our main results are reported in Figure 2, where we compare the main self-correction method (orange) against self-correction with oracle evaluation feedback (blue), sequential scaling without verification (green), and the pass@k performance (red). The methods are evaluated using temperature 0.5 in three environments: 3-SAT formulas with (1) 7 variables and 8 clauses, (2) 7 variables and 12 clauses, and (3) 7 variables and 25 clauses. The first environment is in-distribution and the latter two are increasingly out-of-distribution. Since the difficulty of 3-SAT problems rises with the number of clauses, these environments are in increasing orders of difficulty. Each environment consists of 10,000 problems, and the first sample drawn from the parallel scaling baseline is used as the initial assignment for the sequential methods.

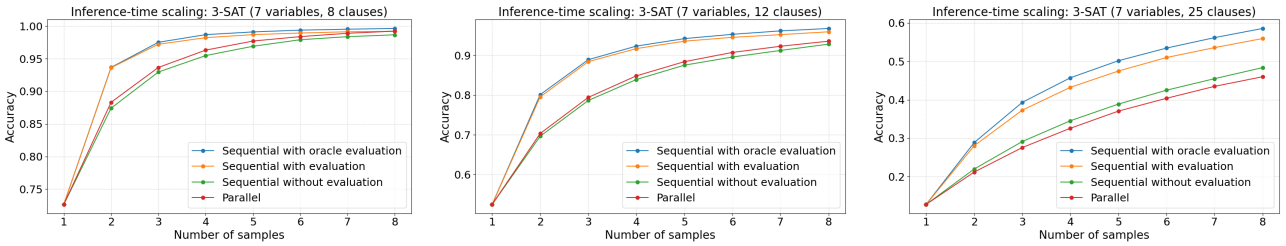

Figure 2. Performance comparison across 10,000 3-SAT problems with 7 variables and varying numbers of clauses: 8 (left), 12 (middle), and 25 (right). Each plot shows success rate as a function of the number of generated samples. Parallel sampling with pass@k is in red, sequential sampling without chain-of-thought verification is in green, self-correction without oracle feedback is in orange, and self-correction with oracle feedback is in blue. Only the orange method operates fully autonomously, without external feedback or pass@k aggregation. For all sequential methods, the initial assignment is shared and drawn from the first sample of the parallel generation model.

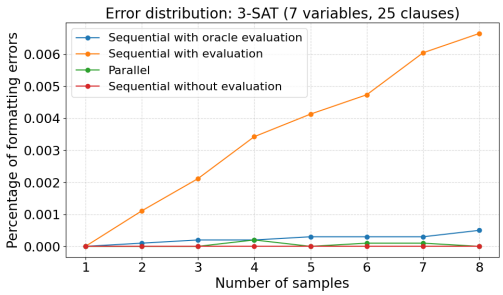

Figure 3. Percentage of formatting errors occurring at each sample number for 3-SAT problems with 7 variables and 25 clauses.

Chain of Thought can verify generalizably. We have shown in Figure 2 that self-correction without external feedback is able to monotonically increase its accuracy over multiple turns even when evaluated in out-of-distribution datasets. This corroborates studies such as (Zhang et al., 2025) in demonstrating the generalizability of CoT verification. Through CoT we were able to teach the model to specifically simulate the standard polynomial-time 3-SAT verification algorithm, enabling out-of-distribution generalization beyond discriminative methods.

Sequential scaling is stronger than parallel scaling. Our sequential method outperforms even the best-case parallel baseline, which assumes access to oracle verification. This shows that the model does more than just resample when incorrect, it learns to leverage its previous attempts to make better-informed guesses. Given that the sequential model without evaluation performs roughly the same as the parallel model across the three environments, we see that the fine-grained CoT evaluation feedback is what allows the model to search more effectively.

Sequential scaling accumulates errors. Figure 2 shows that the accuracy gap between self-correction with oracle evaluation (blue) and without it (orange) widens as both the problem difficulty and the number of sequential steps increase. Since the accuracy of the internal evaluation decreases as the test distribution becomes more out of distribution, the performance difference suggests that strong evaluation plays a large role in the success of sequential scaling. We further investigate why the gap increases with the number of sequential steps, and see in Figure 3 that without oracle evaluation, the percentage of revised answers containing formatting errors accumulates linearly over sequential turns. Because the model cannot revise invalid initial assignments, it cannot recover from formatting errors and accumulates irrecoverable mistakes with each turn.

5. Discussion

Our experiments on the 3-SAT problem show that chain-of-thought evaluation enables LLMs to identify errors in its solutions, and thereby iteratively refine its solutions more efficiently than independent sampling. In an environment where verification is known to be significantly easier than generation, we demonstrate that sequential scaling without oracle information scales better than the best parallel sampling methods (pass@k) over multiple turns even without external feedback and displays generalization to out-of-distribution problems as well as to more turns of sequential scaling than seen in training.

Much of the success of sequential scaling in the 3-SAT setting can be attributed to the strength of the learned evaluation. This suggests future work focused on improving the evaluation quality of self-correcting models in more complex problem settings such as mathematical reasoning and code generation, where strong chain-of-thought verification is harder to achieve. Additionally, our model only considers the last response when revising, due to computational limitations. A model that can consider a larger response history could provide further improvements and may develop more complex search strategies similar to depth-first search.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., and Zhou, M. Codebert: A pre-trained model for programming and natural languages, 2020. URL https://arxiv.org/abs/2002.08155.

Ghanem, M., Schmitt, F., Siber, J., and Finkbeiner, B. Learning better representations from less data for propositional satisfiability, 2024. URL https://arxiv.org/abs/2402.08365.

Havrilla, A., Raparthy, S., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., and Raileanu, R. Glore: When, where, and how to improve llm reasoning via global and local refinements, 2024. URL https://arxiv.org/abs/2402.10963.

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet, 2024. URL https://arxiv.org/abs/2310.01798.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., Zhang, L. M., McKinney, K., Shrivastava, D., Paduraru, C., Tucker, G., Precup, D., Behbahani, F., and Faust, A. Training language models to self-correct via reinforcement learning, 2024. URL https://arxiv.org/abs/2409.12917.

Li, C., Liu, C., Chung, J., Lu, Z., Jha, P., and Ganesh, V. A reinforcement learning based reset policy for cdcl sat solvers, 2024. URL https://arxiv.org/abs/2404.03753.

Li, C., Xue, M., Zhang, Z., Yang, J., Zhang, B., Wang, X., Yu, B., Hui, B., Lin, J., and Liu, D. Start: Self-taught reasoner with tools, 2025. URL https://arxiv.org/abs/2503.04625.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., Hubert, T., Choy, P., de Masson d'Autume, C., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Sutherland Robson, E., Kohli, P., de Freitas, N., Kavukcuoglu, K., and Vinyals, O. Competition-level code generation with alphacode. *Science*, 378 (6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Nguyen, A., Mekala, D., Dong, C., and Shang, J. When is the consistent prediction likely to be a correct prediction?, 2024. URL https://arxiv.org/abs/2407.05778.

OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Qu, Y., Zhang, T., Garg, N., and Kumar, A. Recursive introspection: Teaching language model agents how to self-improve, 2024. URL https://arxiv.org/abs/2407.18219.

Selsam, D. and Bjørner, N. Guiding high-performance sat solvers with unsat-core predictions, 2019. URL https://arxiv.org/abs/1903.04671.

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. Learning a sat solver from single-bit supervision, 2019. URL https://arxiv.org/abs/1802.03685.

Serrano, C. R., Gallagher, J., Yamada, K., Kopylov, A., and Warren, M. A. Self-satisfied: An end-to-end framework for sat generation and prediction, 2024. URL https://arxiv.org/abs/2410.14888.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo,

D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. Generating sequences by learning to self-correct, 2022. URL https://arxiv.org/abs/2211.00053.

Xiong, W., Zhang, H., Ye, C., Chen, L., Jiang, N., and Zhang, T. Self-rewarding correction for mathematical reasoning, 2025. URL https://arxiv.org/abs/2502.19613.

Yang, W., Chen, J., Lin, Y., and Wen, J.-R. Deepcritic: Deliberate critique with large language models, 2025. URL https://arxiv.org/abs/2505.00662.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction, 2025. URL https://arxiv.org/abs/2408.15240.