

ADVERSARIAL TRAINING FOR PROCESS REWARD MODELS

Anonymous authors

Paper under double-blind review

ABSTRACT

Process Reward Models (PRMs) enhance reasoning ability of LLMs by providing step-level supervision. However, their widespread adoption is limited due to expensive manual step-level annotation and poor generalization of static training data to novel errors. We introduce Adversarially Trained PRMs (APRM), where a Generator (G) learns to produce reasoning errors to deceive a PRM (R), while R concurrently learns to detect them. This interaction yields progressively harder negatives for R , improving its robustness and generalization to novel errors without requiring manual step-level labels. Averaged across diverse mathematical reasoning benchmarks, APRM improves solver accuracy by +3.4 percentage points (pp) over the strongest PRM baseline. APRM achieves gains of +5.3 pp on out-of-distribution tasks.

“It is not the strongest of the species that survives, nor the most intelligent, but the one most adaptable to change”
- Charles Darwin

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated impressive capabilities in complex reasoning, even achieving gold-medal-level performance on the International Mathematical Olympiad questions, when augmented with specialized systems (Lu et al., 2024; Luong et al., 2025). However, their reasoning remains fundamentally unreliable, often producing factually incorrect outputs despite appearing coherent (Chu et al., 2023; Bommasani et al., 2021; Uesato et al., 2022). This unreliability poses significant risks, particularly in high-stakes domains such as scientific discovery, finance, or medicine, where subtle errors can have severe consequences. As these models become better, their errors become more nuanced (Lightman et al., 2023c; Zhang et al., 2024; Guan et al., 2025; OpenAI, 2023). Detecting such errors is essential for trustworthy reasoning, necessitating the need for Process Reward Models (PRMs) that can identify very subtle errors.

Current PRM training techniques predominantly rely on static datasets (Lightman et al., 2023a; Wang et al., 2024b; Lightman et al., 2023b). These methods are inherently limited as they provide a fixed error distribution that cannot adapt to more nuanced errors. On the other hand, synthetic data generation techniques like Zhang et al. (2024); Lu et al. (2024b); Cobbe et al. (2021a); Yu et al. (2024b); Uesato et al. (2022) rely on the flawed assumption that correct final answers imply correct intermediate steps. These paradigms lack mechanisms to actively mine harder negatives.

To address this, we require a PRM training paradigm that provides an adaptive curriculum, where negative sample hardness dynamically increases while aligning with the PRM’s evolving capabilities. This makes the PRM robust to increasingly subtle errors and also ensures optimal and smooth learning. In this paper, we introduce Adversarially Trained Process Reward Models (APRM). We formulate PRM training as a two-player, general-sum, non-cooperative game. In APRM (Fig 1), a Generator (G) actively learns to produce plausible but incorrect reasoning steps to deceive a Reward Model (R), which concurrently learns to detect these increasingly subtle errors. This competitive interaction creates the desired adaptive curriculum, directly addressing the limitations of static or passively generated training data.

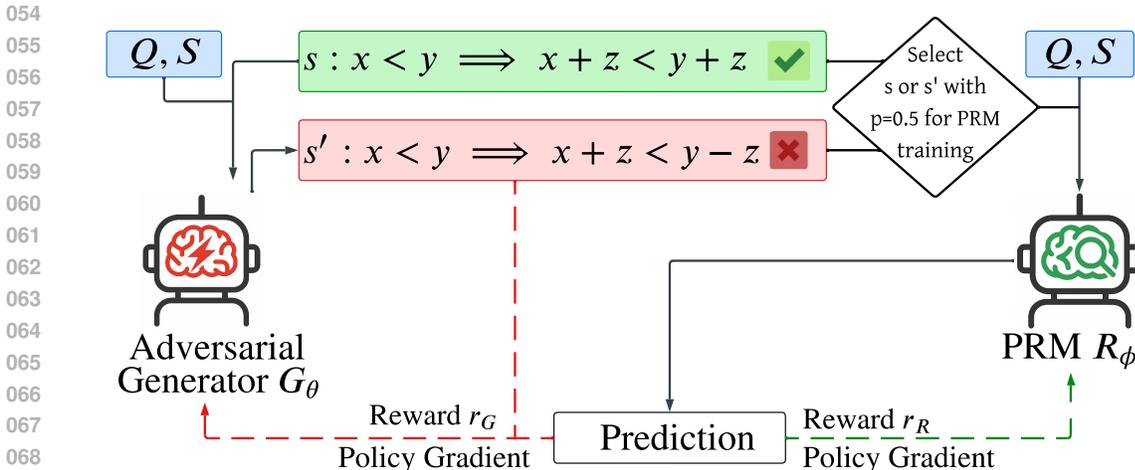


Figure 1: **APRM Overview.** The Generator G_θ perturbs the correct step s into an incorrect step s' . The PRM R_ϕ evaluates a step for its correctness given the question Q and partial solution S . Both G_θ and R_ϕ optimize for rewards given based on the perturbed step and R_ϕ 's prediction, details in section 2.1.

APRM is both theoretically and empirically grounded. We employ game-aware optimizers and symmetric policy regularization to ensure stable training dynamics with provable linear-rate convergence to a Nash Equilibrium. Unlike prior adversarial approaches (Goodfellow et al., 2014; Madry et al., 2017; Wu et al., 2024) that were framed as zero-sum games, we model the Generator-PRM interaction as a general-sum game. This yields a stationary-point Nash Equilibrium rather than a mini-max saddle point. Experiments on mathematical reasoning benchmarks show that APRM outperforms state-of-the-art PRM training and prompting methods by up to 5.3 pp, with the largest gains on out-of-distribution problems, highlighting its enhanced robustness and generalization.

We summarize our contribution as follows:

- **Adversarial training for PRMs.** We introduce PRM training as a two-player adversarial game between a generator and a PRM. The generator learns to produce harder negatives, leading to stronger and more generalizable Process Reward Models.
- **Theoretical Analysis.** We provide theoretical analysis of the game, showing the existence of Nash Equilibria. We establish linear convergence guarantees under regularized game utilities and use game-aware optimizers to enable stable training.
- **Experimental validation.** We experimentally validate the proposed adversarial training setup. Across math and science reasoning benchmarks, APRM yields an average improvement of 3.4 pp over state-of-the-art PRM training methods, consistently observed across solvers of different sizes.

2 ADVERSARIAL PROCESS REWARD MODELS

We formulate the problem of learning a robust process reward model (PRM) as a two-player adversarial game. This game consists of a Generator G , tasked with perturbing a correct reasoning steps to make it incorrect, and a Reward Model R , whose objective is to predict the correctness of a given reasoning step. G aims to deceive R into classifying the corrupted step as correct. R , on the other hand, aims for accurate classification, detecting even the most nuanced errors generated by G (Figure 1). Through this iterative competition, G learns to generate increasingly subtle errors, while R concurrently evolves to detect even the most nuanced flaws.

Notation. We denote the Generator by G_θ and the Reward Model by R_ϕ , parameterized by weights $\theta \in \Theta$ and $\phi \in \Phi$, respectively. A correct reasoning step is represented as a sequence $s = \{s_1, s_2, \dots, s_m\}$, where each s_i is a token from the finite vocabulary \mathcal{V} .

The Generators action is a token sequence $a_G \in \mathcal{A}_G$, where $a_G = \{s'_1, s'_2, \dots, s'_n\}$ with $s'_i \in \mathcal{V}$ and $n \leq \ell$ (ℓ is the maximum sequence length). The action space \mathcal{A}_G thus consists of all token sequences of length 1 to ℓ . These generated sequences correspond to corrupted reasoning steps.

The Reward Model’s action is a binary classification $a_R \in \mathcal{A}_R = \{0, 1\}$, where 1 denotes Correct and 0 denotes Incorrect. The players use mixed strategies (policies) defined as $\pi_\theta \in \Delta(\mathcal{A}_G)$ for the Generator and $\pi_\phi \in \Delta(\mathcal{A}_R)$ for the Reward Model, where $\Delta(\cdot)$ denotes a probability distribution over the respective action spaces. Note that these strategies are distinct from the underlying model parameters θ and ϕ .

Let $y(a_G) \in \{0, 1\}$ be the ground-truth correctness label of a reasoning step generated by G_θ , i.e. $y = 1$ for a correct generated step and, 0 for incorrect generated step, $a_R = y'(a_G) \in \{0, 1\}$ be R_ϕ ’s classification output for G ’s action. The scalar reward functions for G_θ and R_ϕ are represented by $r_G(y, y')$ and $r_R(y, y')$, jointly determined by the ground-truth validity of G_θ ’s output and R_ϕ ’s classification. Detailed definitions are provided in Section 2.1

Utilities represent the expected payoffs for each player. Given a strategy profile (π_θ, π_ϕ) and a distribution $p(s)$ over correct initial reasoning steps s , the utilities for both the players are given by:

$$\begin{aligned} U_G(\pi_\theta, \pi_\phi) &= \mathbb{E}_{s \sim p(s), a_G \sim \pi_\theta(\cdot|s), y' \sim \pi_\phi(\cdot|a_G)} [r_G(y(a_G), y')] \\ U_R(\pi_\theta, \pi_\phi) &= \mathbb{E}_{s \sim p(s), a_G \sim \pi_\theta(\cdot|s), y' \sim \pi_\phi(\cdot|a_G)} [r_R(y(a_G), y')] \end{aligned} \quad (1)$$

2.1 REWARD FUNCTION

We use reinforcement learning (RL) to train G_θ and R_ϕ through iterative adversarial interaction. The correctness $y(a_G)$ of G ’s generation is determined by an algorithmic oracle that uses non-LLM metrics, such as cosine similarity, entity matching (nouns and numbers), and logical operation order, relative to a reference solution. Further details of this oracle are provided in Appendix 6.7. Given $y(a_G)$ and the Reward Model’s classification $y'(a_G)$, the reward functions for both agents are structured as follows:

Reward Function for R_ϕ The PRM is rewarded for accurate classifications. It receives +1 for correct classifications ($y = y'$) and incurs a penalty of -1 for incorrect classifications ($y \neq y'$).

$$r_R(y, y') = \begin{cases} +1 & \text{if } y = y' \\ -1 & \text{if } y \neq y' \end{cases} \quad (2)$$

Reward Function for G_θ The Generator receives rewards that incentivize generation of deceptive errors. It is rewarded +1 for generating an incorrect reasoning step that successfully deceives R into classifying it as correct ($y = 0$ and $y' = 1$). It receives a reward of -1 for failing to corrupt a step (generating a correct step, $y = 1$) and 0 for failing to deceive R_ϕ when it generated an incorrect step ($y = 0$ and $y' = 0$).

$$r_G(y, y') = \begin{cases} +1 & \text{if } y = 0 \text{ and } y' = 1 \text{ (G generates incorrect, fools } R) \\ 0 & \text{if } y = 0 \text{ and } y' = 0 \text{ (G generates incorrect, but is caught by } R) \\ -1 & \text{if } y = 1 \text{ (G produces correct step)} \end{cases} \quad (3)$$

Note that since the reward/payoff functions of the two players are asymmetric (don’t add up to the same value in every case), the game is formulated as a general-sum game.

2.2 GAME FORMULATION AND EQUILIBRIUM CONCEPT

APRM formulates PRM training as a two-player, non-cooperative game, general-sum. The natural solution concept for this adversarial interaction is a Nash Equilibrium (Nash, 1951).

Utility Functions. To ensure good optimization behavior (discussed in Section 2.3), our players optimize regularized utility functions. Both G_θ ’s and R_ϕ ’s utility is augmented with

a Kullback-Leibler (KL) divergence penalty and an entropy bonus that help stabilize learning. These regularized utilities that define the precise objectives of the game, are defined as:

$$\begin{aligned} U'_G(\pi_\theta, \pi_\phi) &= U_G(\pi_\theta, \pi_\phi) - \tau \cdot \text{KL}(\pi_\theta || \pi_{\theta_0}) + c_H \cdot H(\pi_\theta) \\ U'_R(\pi_\theta, \pi_\phi) &= U_R(\pi_\theta, \pi_\phi) - \tau \cdot \text{KL}(\pi_\phi || \pi_{\phi_0}) + c_H \cdot H(\pi_\phi) \end{aligned} \quad (4)$$

Where $U_G(\pi_\theta, \pi_\phi)$ and $U_R(\pi_\theta, \pi_\phi)$ are player utilities defined as expected payoffs(rewards) in Section 2. $\tau, c_H > 0$ are coefficients for KL-regularization and entropy bonuses, respectively. π_{θ_0} and π_{ϕ_0} are the policies of the initial pretrained models. See Appendix 6.8 for a discussion on mode collapse avoidance using these regularization.

Theorem 1 (Existence of Nash Equilibrium) The adversarial game, characterized by agents G_θ and R_ϕ , their regularized utility functions (U'_G and U'_R), and action spaces (\mathcal{A}_G and \mathcal{A}_R), is guaranteed to possess at least one Nash Equilibrium in mixed strategies.

Proof: We invoke *Glicksberg’s generalization of Nash’s existence theorem* (Glicksberg, 1952). The theorem guarantees the existence of a Nash Equilibrium in a game if two conditions are met:

1. *The players’ strategy spaces are non-empty, convex, and compact:* Since the action spaces $\mathcal{A}_G, \mathcal{A}_R$ are finite and discrete, the corresponding policy spaces $\Delta(\mathcal{A}_G), \Delta(\mathcal{A}_R)$ are non-empty, convex, and compact (finite-dimensional simplices).¹
2. *The players’ utility functions are continuous in all players’ strategies:* The utility functions $U'_G(\pi_\theta, \pi_\phi)$ and $U'_R(\pi_\theta, \pi_\phi)$ are expectations of bounded rewards under mixed (policy) distributions over actions. Because these expected-reward expressions are finite linear combinations (or integrals) of policy probabilities, they(including regularization) are continuous in the policy distributions π_θ and π_ϕ .

Given these conditions, Glicksberg’s theorem ensures the existence of at least one Nash Equilibrium.

Nash Equilibrium as a Stationary Point. In a general-sum, non-cooperative games where the utilities are differentiable w.r.t policy parameter, a Nash Equilibrium is characterized as a *stationary point* in the joint strategy space (Raghunathan et al., 2019; Zhang et al., 2019). At such a point, no agent has a local incentive to unilaterally change their strategy. For APRM, this equilibrium signifies a state where G_θ ’s evolving ability to corrupt is optimally countered by R_ϕ ’s ability to detect those corruptions (given their capacities and the training data).

A *stationary point* is qualitatively different from the *saddle point* solution that frequently arises in works like Generative Adversarial Networks (Goodfellow et al., 2014) and robust optimization (Madry et al., 2017). These problems are formulated as zero-sum games where the equilibrium corresponds to a saddle point of a single objective function optimized by both players in perfectly opposing directions, contrary to our general sum formulation.

2.3 CONVERGENCE TO NASH EQUILIBRIUM

In this section, we justify the introduction of regularization into the utility functions. The game described by the non-regularized utility functions U_G and U_R has non-monotone dynamics because the utilities are not simply negative correlated. In non-monotone settings, standard gradient methods often exhibit undesirable behaviors such as cycling, chaotic dynamics, or divergence, thus precluding direct convergence guarantees (Gidel et al., 2018; Fei et al., 2021). To mitigate these stability challenges and ensure provable convergence, we incorporate symmetric policy regularization into the player utility functions, as defined in

¹The convexity referred to in Glicksberg’s theorem is policy-space convexity (i.e. convexity of distributions over actions). This is different from convexity in the parameter space of a neural model (weights θ, ϕ), which is typically non-convex. Existence of an equilibrium in policy space does not imply that there is a single weight vector achieving that equilibrium, or that gradient-based training will find it. Nor do we claim in this paper that we are able to find the Nash equilibrium.

Equation 4. Past works (Azar et al., 2024; Munos et al., 2024) prove that this regularization induces strong concavity in individual player objectives, which leads to a strong monotonicity and hence linear convergence to Nash Equilibrium.

Lemma 1 (Strong Concavity from Regularization): For any utility function $U(\pi)$ and a fixed reference policy π_0 , the regularized objective $U'(\pi) = U(\pi) - \tau \cdot \text{KL}(\pi||\pi_0)$ is τ -strongly concave with respect to the policy π . Similarly, an entropy bonus $\tau \cdot H(\pi)$ induces τ -strong concavity (Azar et al., 2024).

Applying the lemma, and redefining the utility functions as given by Equation 4, we ensure that, $U'_G(\pi_\theta, \pi_\phi)$ and $U'_R(\pi_\theta, \pi_\phi)$ are τ and c_H -strongly concave with respect to the player policies. If each player’s utility function is strongly concave with respect to its own policy, for any fixed opponent policy, then the operator $F(z) = (-\nabla(U_\theta(\pi_\theta, \pi_\phi)), -\nabla(U_\phi(\pi_\theta, \pi_\phi)))$, where $z = (\theta, \phi)$, becomes strongly monotone (Daskalakis & Panageas, 2018). Furthermore, for games characterized by a strongly monotone operator, game-aware first-order optimizers offer strong convergence guarantees, specifically ensuring linear convergence rates.

Theorem 2 (Linear Convergence to Nash Equilibrium): For the strongly monotone game described by the regularized utility functions (Section 2.2) algorithms such as Optimistic Gradient Descent-Ascent (OGDA) or Extra-Gradient (Mertikopoulos et al., 2019; Daskalakis et al., 2018b) are guaranteed to generate a sequence of iterates z_t that converges to the unique Nash Equilibrium z^* at a linear rate. (For proof, see Appendix 6.10).

2.4 TRAINING ALGORITHM

Both G_θ and R_ϕ are trained using PPO (Schulman et al., 2017) augmented with Optimistic Gradient Descent-Ascent (OGDA) updates (see Section 2.3).

Loss Function. For each player, we augment the PPO clipped surrogate objective with KL and Entropy regularization terms. For instance, G_θ ’s the loss function (similarly for R_ϕ) is given by:

$$L_G^{\text{total}}(\theta) = L_G^{\text{PPO,clip}}(\theta) + L_G^{\text{KL}}(\theta) + L_G^H(\theta) \quad (5)$$

Where minimizing L^{PPO} approximately maximizes the un-regularized utility functions U_G, U_R defined in Section 2. L^{KL} given by $\tau \cdot \mathbb{E}_t [\text{KL}(\pi_\theta||\pi_{\theta_0})]$ and L^H given by $-c_H \cdot \mathbb{E}_t [H(\pi_\theta)]$ correspond to the regularization introduced in Section 2.3.

Game-Aware Optimization. We replace the standard PPO’s gradient updates by OGDA (Mertikopoulos et al., 2019; Daskalakis et al., 2018b), which efficiently navigates the rotational game dynamics, in the regularized strongly monotone setting, by incorporating past gradient information. The update rule for OGDA is given by:

$$z_{t+1} = z_t - \eta(2F(z_t) - F(z_{t-1})) \quad (6)$$

where, η is the learning rate, and $F(z_t)$ is the gradient function w.r.t. the player’s policy parameters, as described in section 2.3.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETTING

Model Architecture Both the Generator (G_θ) and the Process Reward Model (R_ϕ) use Llama-3.1-8B as the backbone. This model offers a practical trade-off between computational cost and capacity for mathematical reasoning.

Training We train on the train split of MATH dataset (Hendrycks et al., 2021). The dataset provides with gold solution steps, the generator G_θ learns to perturb these solution steps. We alternate training between players, updating one for 5 gradient update steps while freezing the other. We use PPO (Schulman et al., 2017) with Optimistic Gradient Descent-Ascent (OGDA) (Mertikopoulos et al., 2019) as discussed in Section 2.4. R_ϕ is trained on a mixture of gold steps (50%) and negatives drawn from both current and past versions of G_θ (50%) to prevent forgetting. Hyperparameter details are given in the App 6.14.

Table 1: Performance comparison of APRM with the best MCTS based PRM training and prompting methods on math reasoning benchmarks. Please see Table 5 for a full comparison with other PRM training baselines.

Method	MATH500	JEEB.	OlympiadB.	AIME25	AMC	Avg.
GPT-OSS-120B						
CoT-SC	84.4	62.8	83.9	85.5	60.0	77.2
LLM-J	85.2	66.0	83.1	90.4	80.0	79.3
ToT	87.6	66.4	84.3	89.2	66.7	79.9
ReST-MCTS	91.6	69.1	84.3	91.0	67.4	81.7
Hard Neg.	91.2	69.5	79.2	91.5	68.5	80.0
ARM	89.2	62.4	77.6	87.9	63.3	76.1
APRM (Ours)	91.4	70.3	89.4	94.5	70.7	83.0
GPT-OSS-20B						
CoT-SC	78.0	48.4	61.0	69.9	33.3	62.2
LLM-J	85.2	68.0	83.9	92.8	60.0	80.3
ToT	87.6	67.6	85.6	90.4	66.7	80.7
ReST-MCTS	91.0	68.3	84.7	91.3	66.0	82.3
APRM (Ours)	91.0	73.0	90.4	93.6	68.0	85.0
Gemma-3-27B						
CoT-SC	78.8	48.8	48.7	69.9	13.3	58.4
LLM-J	86.4	69.2	83.1	91.6	73.3	80.4
ToT	88.0	67.6	85.2	88.0	60.0	80.2
ReST-MCTS	93.2	70.3	84.3	90.6	61.2	72.6
Hard Neg.	86.0	63.0	83.0	89.1	60.0	76.2
ARM	86.0	51.0	61.0	71.4	30.0	59.9
APRM (Ours)	91.4	74.4	90.7	91.9	60.8	85.2
Gemma-3-12B						
CoT-SC	76.8	43.2	35.6	53.0	23.3	51.4
LLM-J	72.8	40.4	37.7	55.4	20.0	49.9
ToT	77.6	49.2	40.7	61.4	20.0	55.3
ReST-MCTS	81.4	50.7	40.3	62.6	20.2	56.9
APRM (Ours)	80.0	53.1	44.0	65.7	21.6	58.6

Baselines We compare against methods with comparable test time compute, which includes a) prompting-based methods b) trained reward models. Prompting based methods include CoT with self-consistency (k=5) (Wang et al., 2023), Tree of Thoughts (branching factor 5) (Yao et al., 2023) and LLM-as-a-Judge (Zheng et al., 2023) using same model as solver and as judge sampling a maximum of 5 times. Trained reward models include outcome-based approximations methods - AutoPSV (Lu et al., 2024a), trained PRM using static datasets - Let’s Verify Step by Step (Lightman et al., 2023c), and MCTS based data generation techniques to train PRMs - Math-Shepherd (Wang et al., 2024a), rStar-Math (Guan et al., 2025) and ReST-MCTS (Zhang et al., 2024). We re-implement all the methods using Llama-3.1-8B as a base model. Table 1 gives performance comparison with prompting and the best trained PRM baseline. A full comparison with all the baselines is given in Table 5.

Test Setup We evaluate on five math reasoning benchmarks: MATH500 (Hendrycks et al., 2021) (in-domain, in-distribution), JEEBench (Arora et al., 2023), OlympiadBench (He et al., 2024), AIME25 (Patel et al., 2024), and AMC (AI-MO, 2024) (in-domain, out-of-distribution). AIME25 contains recent examination questions, ensuring no data contamination. The effectiveness of trained PRMs is assessed by evaluating the performance of a solver model supervised by the PRM during test time. We sample a maximum of 5 times per step. We experiment on GPT and Gemma model families with different sizes (GPT-OSS-120B, GPT-OSS-20B, Gemma-3-12B-it and Gemma-3-27B-it) using API based inference. See App 6.9 for compute details.

Table 2: Performance comparison APRM with baselines on SciBench. Domains: atk=Atkins, cal=Calculus, cmc=ChemMC, cls=Classical, dif=Differential Eq., fun=Fundamentals, mat=Matter, qua=Quantum, sta=Statistics, the=Thermodynamics.

Method	atk	cal	cmc	cls	dif	fun	mat	qua	sta	the	Overall
GPT-OSS-120B											
ToT	58.3	96.2	43.5	51.6	74.1	37.5	46.4	50.0	89.1	61.0	61.4
ReST-MCTS	58.3	88.5	43.5	52.1	74.1	35.0	50.0	50.0	91.3	70.7	62.3
APRM (Ours)	60.3	91.2	44.8	52.4	76.9	36.6	51.5	50.0	92.5	73.1	64.0
GPT-OSS-20B											
ToT	60.0	88.5	43.5	45.2	74.1	37.5	46.4	60.0	89.1	65.9	61.7
ReST-MCTS	61.3	86.2	43.5	44.5	71.5	37.5	50.0	60.0	89.1	66.2	61.8
APRM (Ours)	61.8	86.1	44.8	47.6	75.1	38.8	52.0	61.3	91.0	65.3	63.0
Gemma-3-27B-IT											
ToT	60.0	92.3	43.5	48.4	77.8	30.0	50.0	50.0	89.1	63.4	61.1
ReST-MCTS	60.0	92.2	43.5	48.4	77.8	31.5	57.1	50.0	89.1	63.2	61.8
APRM (Ours)	62.2	94.6	45.1	50.2	80.2	31.9	59.3	52.0	91.4	63.4	63.5
Gemma-3-12B-IT											
ToT	43.3	73.1	34.8	51.6	59.3	40.0	21.4	20.0	80.4	36.6	47.6
ReST-MCTS	43.3	69.2	34.8	35.5	44.4	32.5	25.0	10.0	76.1	26.8	41.8
APRM (Ours)	44.8	71.5	35.9	36.8	45.8	33.6	25.9	10.4	78.4	27.8	43.2

3.2 EXPERIMENTAL RESULTS

APRM outperforms baselines Table 1, shows the performance LLM solvers guided by our PRM against prior methods under equivalent test-time compute. Averaged over five benchmarks and four solver models, APRM improves solver accuracy by +3.4 pp relative to the strongest trained PRM baseline (ReST-MCTS (Zhang et al., 2024)) and by +4.2 pp relative to the best prompting method. We observe the largest gains on JeeBench, which is an out-of-distribution benchmark, where APRM outperforms ReST-MCTS by +5.3 pp, suggesting better generalization beyond the training distribution.

APRM remains effective as with solver scales We evaluate whether APRM remains effective as the solver size increases. Table 1 shows that APRM’s average improvement over prompt-based methods grows from +3.3 pp with Gemma-3-12B, to +4.3 pp with GPT-OSS-20B, and further to +5.0 pp with Gemma-3-27B. In contrast, PRMs trained on non-adaptive data (e.g. ReST-MCTS) show gains that are smaller and in some cases decline as solver size increases, dropping from +1.6 pp on Gemma-3-12B to -7.6 pp on Gemma-3-27B. These results support our hypothesis that adversarial training enables robust PRMs that remain effective even as solver models become stronger.

APRM generalizes across domains To test whether our adversarial training captures general error patterns rather than overfitting to math, we evaluate on SciBench (Wang et al., 2024c), comparing APRM with the best prompting and PRM baselines (Tab. 2). On GPT-OSS-20B, APRM reaches 63.0% accuracy versus 61.8% for ReST-MCTS and 61.1% for ToT. On GPT-OSS-120B, it achieves 64.0%, compared to 62.3% and 61.4%. This shows that APRM learns transferable properties of incorrect reasoning.

APRM provides strong reward signal for RL Post-Training To assess APRM’s utility beyond inference-time search, we evaluate its effectiveness as a reward signal for RL post-training. We fine-tune Gemma-3-12B on the MATH dataset using GRPO, comparing APRM against standard outcome supervision and the strongest PRM baseline (ReST-MCTS). As shown in Table 3, APRM provides a significantly stronger learning signal, outperforming outcome-based RL by +6.8 pp on MATH500 and demonstrating superior generalization on OOD benchmarks like JEEBench (+10.4 pp). Training details are provided in Appendix 6.14.

Performance on PRM benchmarks We evaluate whether the predictions of APRM align with human annotations by comparing it’s performance with the baselines on a subset

Table 3: **RL Post-Training Performance.** Accuracy (%) of Gemma-3-12B fine-tuned using different reward signals. APRM serves as a more effective dense reward than baselines.

Method	MATH500	JEEB.	OlympiadB.	AIME25	AMC
Base Model (CoT)	76.8	43.2	35.6	53.0	23.3
Final Correct (Outcome)	80.0	48.2	40.0	59.8	21.0
ReST-MCTS PRM	83.0	51.6	43.2	61.4	23.3
APRM (Ours)	86.8	58.6	48.0	65.7	25.3

of the PRM800K test set (Lightman et al., 2023b). As shown in Figure 4, we report accuracy, precision, recall, and F1 score for all methods. We find that APRM achieves the highest performance across all metrics, indicating stronger agreement with human-annotations compared to baselines.

Scaling properties of APRM We study how APRM scales with the backbone sizes of G_θ and R_ϕ , using Llama-3.1-8B and Llama-2-13B in four settings: (1) $G=8B, R=8B$; (2) $G=8B, R=13B$; (3) $G=13B, R=8B$; and (4) $G=13B, R=13B$. Figure 2 reports solver accuracy averaged over all math benchmarks, showing monotonic gains from (1) to (4) with an overall improvement of ~ 3.8 pp. We observe that gains are more when increasing the generator’s capacity compared to increasing the reward model. We hypothesize that larger generators exposes the PRM to broader distribution reasoning errors.

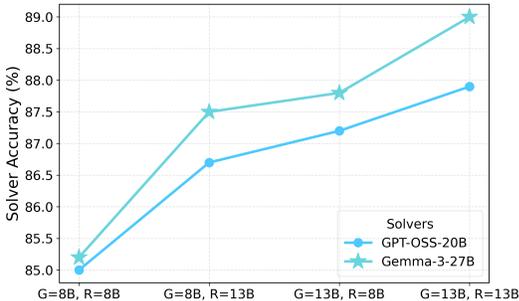


Figure 2: APRM augmented solver accuracy averaged across math reasoning benchmarks for different R_ϕ and G_θ backbones.

Ablation Study We ablate on entropy regularization and the OGD optimizer. Table 4 shows results on MATH, JEE, and OlympiadBench using GPT-OSS-20B. Removing entropy regularization reduces accuracy by about 4 points, while removing OGD has a similar effect. This complies with the theory, that each component helps stabilize training.

Table 4: Ablation study on GPT-OSS-20B. Accuracy (%) on representative benchmarks.

Variant	MATH	JEE	Oly	Overall
APRM (full)	91.0	73.0	90.0	85.2
– no entropy reg.	86.4	69.4	85.5	80.9
– no OGD	84.6	67.9	83.7	79.2
– both removed	83.6	67.2	82.8	78.4

3.3 QUALITATIVE ANALYSIS

Evolution of Generator perturbations. We first analyze how the Generator’s perturbations evolve during training. In the die-rolling problem (Fig. 6.2), G initially perturbs the step to make a shallow mistake of as miscounting the number of odd faces. By mid-training, it produces steps that look locally consistent but misinterpret the problem statement, e.g., doubling all faces without considering conditional rules. At later stages, it introduces subtle logical errors, like averaging probabilities instead of taking a weighted average. This trajectory suggests that adversarial interaction trains G to generate progressively harder negatives for R .

APRM with different solver sizes. We next examine APRM-trained PRMs when the solver capacity increases. In the shortest-path problem (Fig. 6.3), GPT-oss-20B uses a non-coherent languages and produces a confusing and incorrect coordinate setup, while GPT-oss-120B generates a more coherent geometric reasoning but still mislocates the position of gecko. In

both cases, APRM flags the faulty reasoning step, showing that adversarial training makes the PRM robust to errors even when the text is fluent.

Cross-domain transfer. We test whether APRM captures generalizable properties of flawed reasoning by applying it to a chemistry problem from SciBench (Fig. 6.4). In this case, the solver neglected to convert centimeters to meters before applying $\Pi = \rho gh$. APRM-trained PRM flagged this unit error, which lead to correct final answer. This example suggests that adversarially generated negatives are not tied to math-specific syntax errors but can transfer to scientific reasoning tasks.

Comparison with PRM baselines. We further compare APRM with ReST-MCTS on a calculus reasoning problem (Fig. 6.5). The problem asks to calculate the value of an expression at $x = 1$, where the numerator and denominator approach to zero. The solver factorizes and cancels the common factor and computes the value. This step appears valid, and even produces the correct final answer. However, this cancellation is invalid at $x = 1$ point, because it amounts to dividing by zero. ReST-MCTS fails to flag this subtle mistake, whereas, APRM flags this. This example highlights how adversarial training exposes PRMs to errors that lead to correct final answer, which static rollouts are not able to.

Failure Cases. Finally, we analyze the errors missed by APRM. We find that one recurring failure mode occurs when solvers invoke theorems without checking preconditions. For instance, in the series convergence problem (Fig. 6.6), GPT-OSS-120B applied the Alternating Series Test to a sequence that does not alternate term-by-term, and the PRM accepted it. Similarly, in an inequality problem, the solver applied Jensen’s inequality outside its domain of concavity, and APRM again failed to flag the mistakes. We hypothesize that G struggles to generate adversarial negatives that specifically target theorem preconditions, this might be a limitation due to the size of the backbone model used.

4 RELATED WORK

Mathematical Reasoning with LLMs LLM reasoning has advanced through chain-of-thought prompting (Wei et al., 2022), self-consistency (Wang et al., 2023), and outcome-level reinforcement learning (DeepSeek-AI, 2025; OpenAI, 2024). Outcome-based reward models (Cobbe et al., 2021b; Yu et al., 2024a) give sparse feedback (Cui et al., 2025) and can reinforce faulty reasoning. Process Reward Models (PRMs) (Luo et al., 2024; Cui et al., 2025; Havrilla et al., 2024; Lightman et al., 2023c; Lu et al., 2024a; Li & Li, 2025) address this by supervising intermediate steps, but require costly human annotations. Synthetic data generation methods (Wang et al., 2024a; Zhang et al., 2024; Guan et al., 2025; Sun et al., 2025; Du et al., 2025; Li et al., 2025) generate synthetic step-level data, yet rely on final-answer correctness, which fails when wrong steps yield right answers. APRM instead trains a generator adversarially to produce hard, plausibly incorrect steps that target the PRM’s weaknesses. This yields a dynamic curriculum of negatives, continuously adapting to the PRM’s blind spots rather than relying on fixed heuristics.

Adversarial Training and Robustness Adversarial training was introduced with GANs (Goodfellow et al., 2014), where a generator learns a target distribution by fooling a discriminator, and later extended to robust optimization (Madry et al., 2017) and GAIL (Ho & Ermon, 2016) for policy imitation. APRM applies this idea to LLM reasoning steps, which are discrete and require a different setup. Unlike GANs or robust optimization that use zero-sum, single-objective formulations (Madry et al., 2017; Goodfellow et al., 2014), APRM defines a general-sum game with non-opposing utilities for the Generator and Reward Model. This yields equilibria that are stationary points of the game dynamics (Nash equilibria) rather than simple minimax solutions. APRM also connects to self-play in multi-agent RL (Baker et al., 2020) and recent advances in optimization for adversarial learning (Mertikopoulos et al., 2019; Daskalakis et al., 2018b). Further, more recently, Bukharin et al. (2025) applied adversarial training to outcome reward models using ensemble disagreement. In contrast, APRM targets process supervision and evolves the generator and PRM in tandem, compared to utilizing a static generator. We defer a detailed discussion to Appendix 6.11.

5 CONCLUSION, LIMITATIONS AND FUTURE WORK

In this work, we introduced APRM, an adversarial framework to improve process supervision for LLM reasoning. Unlike prior approaches that rely on outcome-based feedback, static human-annotated datasets, or heuristic search with MCTS, APRM actively optimizes for hard negatives to train the PRM. This adversarial setup exposes the PRM to subtle reasoning errors making it robust and generalizable across domains.

A limitation of our method is the computational cost. While inference-time compute is unchanged, training requires additional resources due to the adversarial optimization between generator and PRM. This cost is incurred only once, but remains higher than approaches based solely on static datasets.

As the availability of high-quality training data for large models plateaus, adversarial training offers a way to generate targeted supervision, specifically focusing on model weaknesses, where data collection is expensive or impractical. This makes the approach relevant beyond mathematical reasoning, with potential applications in scientific domains, robotics, and other areas where annotation is difficult to scale.

Future work includes extending the framework beyond the two-player formulation. Multi-agent variants could introduce diverse adversarial roles that target different error modes. Another direction is applying the approach to embodied agents and robotics, where agents must learn to operate in noisy environments induced by other agent’s adversarial interactions. On the optimization side, although our analysis established convergence under OGDA with regularization, further work is needed to understand large-scale training dynamics and to develop algorithms that enhance stability.

REFERENCES

- 540
541
542 AI-MO. AI-MO/aimo-validation-amc dataset. <https://huggingface.co/datasets/AI-MO/aimo-validation-amc>, 2024. Accessed: <Date you accessed the dataset>.
- 543
544 Daman Arora, Himanshu Gaurav Singh, and Mausam. Have llms advanced enough? a
545 challenging problem solving benchmark for large language models, 2023. URL <https://arxiv.org/abs/2305.15074>.
- 546
547 Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello,
548 Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning
549 from human preferences. *AISTATS*, 2024.
- 550
551 Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew,
552 and Igor Mordatch. Emergent tool use from multi-agent autocurricula, 2020. URL
553 <https://arxiv.org/abs/1909.07528>.
- 554
555 Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent
556 complexity via multi-agent competition, 2018. URL <https://arxiv.org/abs/1710.03748>.
- 557
558 Rishi Bommasani, Drew Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx,
559 Michael Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. Opportunities
560 and risks of foundational models. *arXiv preprint arXiv:2108.07258*, 2021.
- 561
562 Alexander Bukharin, Haifeng Qian, Shengyang Sun, Adithya Renduchintala, Soumye Singhal,
563 Zhilin Wang, Oleksii Kuchaiev, Olivier Delalleau, and Tuo Zhao. Adversarial training
564 of reward models. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=H6Ae8Po6fS>.
- 565
566 Mengjiao Chu, Li Dai, Xinyi Dong, Hongxu Han, Kai Song, Huaimin Wen, Zheng Yuan, Bin
567 Zhang, Yichun Chen, and Zhicheng Wang. Investigating the faithfulness of attention in
568 deep models. *arXiv preprint arXiv:2308.14081*, 2023.
- 569
570 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz
571 Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher
572 Hesse, and John Schulman. Training verifiers to solve math word problems, 2021a. URL
573 <https://arxiv.org/abs/2110.14168>.
- 574
575 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz
576 Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher
577 Hesse, and John Schulman. Training verifiers to solve math word problems, 2021b. URL
578 <https://arxiv.org/abs/2110.14168>.
- 579
580 Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan,
581 Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards.
582 *arXiv preprint arXiv:2502.01456*, 2025.
- 583
584 Constantinos Daskalakis and Georgia Panageas. Gans may have no nash equilibria even
585 with an infinite number of samples. *Advances in Neural Information Processing Systems*,
586 31, 2018.
- 587
588 Constantinos Daskalakis, George Daskalakis, and Ioannis Panageas. Optimistic mirror
589 descent in saddle-point problems. *arXiv preprint arXiv:1807.02629*, 2018a.
- 590
591 Constantinos Daskalakis, Andrew Ilyas, and Ioannis Panageas. Training gans is as easy as
592 solving a minmax game. In *International Conference on Learning Representations*, 2018b.
- 593
DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement
learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Lingxiao Du, Fanqing Meng, Zongkai Liu, Zhixiang Zhou, Ping Luo, Qiaosheng Zhang, and
Wenqi Shao. Mm-prm: Enhancing multimodal mathematical reasoning with scalable
step-level supervision, 2025. URL <https://arxiv.org/abs/2505.13427>.

- 594 Mengyan Fei, Yuandong Zhang, and Jialun Wang. Provably training multi-agent reinforce-
595 ment learning with non-monotonic policy gradients. Advances in Neural Information
596 Processing Systems, 34:25016–25029, 2021.
- 597 Gauthier Gidel, Tony Huang, and Jon Gabriel. Variational inequalities for generative
598 adversarial networks. arXiv preprint arXiv:1802.04655, 2018.
- 600 Gauthier Gidel, Clément Laurent, and Francis Bach. Variational inequality approach to
601 studying convergence of learning dynamics. arXiv preprint arXiv:1904.09340, 2019.
- 602 Irving L Glicksberg. A further generalization of the Nash equilibrium point theorem. Pacific
603 Journal of Mathematics, 2(2):415–427, 1952.
- 604 Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil
605 Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL
606 <https://arxiv.org/abs/1406.2661>.
- 607 Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and
608 Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep
609 thinking, 2025.
- 610 Alex Havrilla, Sharath Raparthi, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym
611 Zhuravinskyi, Eric Hambro, and Roberta Raileanu. Glore: When, where, and how to
612 improve llm reasoning via global and local refinements, 2024. URL <https://arxiv.org/abs/2402.10963>.
- 613 Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi
614 Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun.
615 Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual
616 multimodal scientific problems, 2024. URL <https://arxiv.org/abs/2402.14008>.
- 617 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang,
618 Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the
619 math dataset, 2021. URL <https://arxiv.org/abs/2103.03874>.
- 620 Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016. URL
621 <https://arxiv.org/abs/1606.03476>.
- 622 G M Korpelevich. Extragradient method for finding saddle points and for solving other
623 problems. Ekonomika i Matematicheskie Metody, 13(4):627–641, 1977.
- 624 Ruosen Li, Ziming Luo, and Xinya Du. Fg-prm: Fine-grained hallucination detection and
625 mitigation in language model mathematical reasoning, 2025. URL <https://arxiv.org/abs/2410.06304>.
- 626 Wendi Li and Yixuan Li. Process reward model with q-value rankings, 2025. URL <https://arxiv.org/abs/2410.11287>.
- 627 Zhichao Liang, Tze Leung Lai, and Rui Song. Linear convergence of optimistic gradient
628 descent ascent for bilinear games. arXiv preprint arXiv:1905.10547, 2019.
- 629 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee,
630 Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step.
631 arXiv preprint arXiv:2305.20050, 2023a.
- 632 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee,
633 Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step.
634 arXiv preprint arXiv:2305.20050, 2023b.
- 635 Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee,
636 Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step.
637 arXiv preprint arXiv:2305.20050, 2023c.
- 638
639
640
641
642
643
644
645
646
647

- 648 Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, and Zhijiang
649 Guo. Autopsv: Automated process-supervised verifier. In Advances in Neural Information
650 Processing Systems, 2024a.
- 651
652 Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, and Zhijiang
653 Guo. AutoPSV: Automated process-supervised verifier. Advances in Neural Information
654 Processing Systems, 2024b.
- 655 Xiaoxuan Lu, Jiaming Cao, Guiding Li, Jianqiao Luan, Yining Shen, Zhichao Wang, Jianhao
656 Xu, Xinyun Chen, Si Cheng, Jianmin Deng, Yang Fan, Yu Gou, Yong Hou, Ziniu Hu,
657 Hang Jin, Shuyuan Li, Yi Li, Ziyue Li, Zhilin Li, Kai Lu, Liangchen Luo, Qianli Ma,
658 Fan Mei, Wei Pan, Xin Song, Yi Song, Youbin Sun, Minghao Tan, Mengdi Wang, Siyuan
659 Wang, Shengkai Wu, Tao Xie, Runxin Xu, Shuhao Xu, Yi Yang, Yun Yang, Qian Yu,
660 Lifan Yuan, Sining Zhang, Yuqi Zhang, Rui Zhao, Xiaogang Zhou, Chenghao Zhu, Yi Zhu,
661 Zimin Chen, Yimin Guo, Yibo Wang, Chao Yu, Jianwen Zhang, Jingren Zhou, Hang Li,
662 and Yu Zhou. AlphaGeometry: An AI system that finds a new way to solve IMO geometry
663 problems. Nature, 625(7995):535–540, 2024c. doi: 10.1038/s41586-023-06924-6.
- 664 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan
665 Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical
666 reasoning in language models by automated process supervision, 2024. URL <https://arxiv.org/abs/2406.06592>.
- 667
668 Thang Luong, Edward Lockhart, and DeepMind Team. Advanced version of gemini with
669 deep think officially achieves gold-medal standard at the international mathematical
670 olympiad. Google DeepMind Blog, July 2025. URL <https://deepmind.google/discover/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/>. Accessed:
671 YYYY-MM-DD.
- 672
673 Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Alexandru
674 Vladu. Towards deep learning models resistant to adversarial attacks. arXiv preprint
675 arXiv:1706.06083, 2017.
- 676
677 Panayotis Mertikopoulos, Ioannis Panageas, Ioannis Zadik, and Gui-Hong Zhou. Optimistic
678 gradient descent ascent for zero-sum games and connections to implicit learning. In
679 Advances in Neural Information Processing Systems, volume 32, 2019.
- 680
681 Aryan Mokhtari, Asuman Ozdaglar, and Nathan Srebro. A unified analysis of extra-gradient
682 and optimistic gradient methods for saddle point problems. In International Conference
683 on Artificial Intelligence and Statistics, pp. 3704–3714. PMLR, 2020.
- 684
685 Rémi Munos, Xu Jiang, Guiding Li, Shaan Potharaju, Yichen Sun, Yu Wang, Chong Wu,
686 Changxiao Xu, Qizhe Zeng, Junyang Zhang, and Zihang Zhang. Nash learning from
687 human feedback. ICML, 2024.
- 688
689 John Nash. Non-cooperative games. Annals of Mathematics, 54(2):286–295, 1951.
- 690
691 OpenAI. GPT-4 Technical Report. <https://cdn.openai.com/papers/gpt-4.pdf>, 2023.
The authors list the organization OpenAI as the sole author.
- 692
693 OpenAI. OpenAI o1 System Card. 2024. Accessed: <Date you accessed the document>.
- 694
695 Bhrij Patel, Souradip Chakraborty, Wesley A. Suttle, Mengdi Wang, Amrit Singh Bedi, and
696 Dinesh Manocha. Aime: Ai system optimization via multiple llm evaluators, 2024. URL
<https://arxiv.org/abs/2410.03131>.
- 697
698 Arvind U. Raghunathan, Anoop Cherian, and Devesh K. Jha. Game theoretic optimization
699 via gradient-based nikaido-isoda function. arXiv preprint arXiv:1905.05927, 2019. URL
<https://arxiv.org/abs/1905.05927>.
- 700
701 Herbert Robbins and Sutton Monro. A stochastic approximation method. The Annals of
Mathematical Statistics, 22(3):400–407, 1951.

- 702 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal
703 policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- 704
- 705 Lin Sun, Chuang Liu, Xiaofeng Ma, Tao Yang, Weijia Lu, and Ning Wu. Freeprm: Training
706 process reward models without ground truth process labels, 2025. URL <https://arxiv.org/abs/2506.03570>.
- 707
- 708 Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang,
709 Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with
710 process- and outcome-based feedback. arXiv preprint arXiv:2204.09014, 2022.
- 711
- 712 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu,
713 and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human
714 annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings
715 of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume
716 1: Long Papers), pp. 9426–9439, Bangkok, Thailand, August 2024a. Association for
717 Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL <https://aclanthology.org/2024.acl-long.510/>.
- 718
- 719 Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and
720 Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human an-
721 notations. Proceedings of the 62nd Annual Meeting of the Association for Computational
722 Linguistics (Volume 1: Long Papers), pp. 9426–9439, 2024b.
- 723
- 724 Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R.
725 Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. SciBench: Evaluating College-Level
726 Scientific Problem-Solving Abilities of Large Language Models. In Proceedings of the
727 Forty-First International Conference on Machine Learning, 2024c.
- 728
- 729 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha
730 Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in
731 language models, 2023. URL <https://arxiv.org/abs/2203.11171>.
- 732
- 733 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi,
734 Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language
735 models. arXiv preprint arXiv:2201.11903, 2022.
- 736
- 737 Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu.
738 Self-play preference optimization for language model alignment, 2024. URL <https://arxiv.org/abs/2405.00675>.
- 739
- 740 Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and
741 Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language
742 models, 2023. URL <https://arxiv.org/abs/2305.10601>.
- 743
- 744 Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for
745 planning in mathematical reasoning, 2024a. URL <https://arxiv.org/abs/2311.09724>.
- 746
- 747 Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for
748 planning in mathematical reasoning, 2024b. URL <https://arxiv.org/abs/2311.09724>.
- 749
- 750 Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-
751 MCTS*: LLM self-training via process reward guided tree search. arXiv preprint
752 arXiv:2406.03816, 2024.
- 753
- 754 Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Policy optimization provably converges
755 to nash equilibria in zero-sum linear quadratic games. arXiv preprint arXiv:1906.00729,
2019. URL <https://arxiv.org/abs/1906.00729>.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao
Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez,
and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL
<https://arxiv.org/abs/2306.05685>.

756 6 APPENDIX

757
758
759 6.1 LLM AID FOR WRITING

760 We use LLMs only for making the writing concise. We provide the LLM with the text and
761 ask it not to change any information, just make it concise. We the manually verify the text
762 and make changes to it. Few examples are given below:
763
764

Original	LLM + Human Polishing
<p>765 A future work is to extend the adversarial formulation to other game-theory 766 settings. In this paper, we focused on 767 a two-player setup between a generator 768 and a reward model, but multi-agent 769 extensions could allow for diverse adversarial 770 roles that challenge models along 771 different dimensions of reasoning. Ad- 772 ditionally, while our analysis showed 773 convergence under OGDA with regu- 774 larization, more work is needed to fully 775 understand stability in large-scale ad- 776 versarial training for language models. 777 Exploring optimization techniques that 778 reduce training instability while retain- 779 ing the benefits of adversarial training 780 is an important next step. 781 782 783 784</p>	<p>Future work includes extending the 765 framework beyond the two-player for- 766 mulation. Multi-agent variants could 767 introduce diverse adversarial roles that 768 probe different aspects of reasoning. 769 On the optimization side, while our 770 analysis demonstrated convergence with 771 OGDA and regularization, further study 772 is needed to understand stability at 773 scale and to design algorithms that im- 774 prove training efficiency while preserv- 775 ing the benefits of adaptive curricula. 776 777 778 779 780 781 782 783 784</p>

785
786
787 Another example:

Original	LLM + Human Polishing
<p>790 To access the effectiveness of our frame- 791 work, we evaluate the accuracy of LLM 792 solvers guided by our trained PRM 793 against the strongest existing meth- 794 ods (See Table 1). Averaged across 795 five math-reasoning benchmarks, un- 796 der equivalent test-time compute bud- 797 gets, APRM improves solver accuracy 798 by ... over the strongest trained 799 PRM baseline (ReST-MCTS) and by ... 800 over best prompting methods (Tree-of- 801 Thought). We observe the highest gains 802 on OlympiadBench, which is an out- 803 of-distribution (OOD) benchmark for 804 the PRM. Here, APRM yields a substan- 805 tial ... pp improvement over the best 806 baseline (ReST-MCTS), which strongly 807 indicates enhanced generalization and 808 reduced overfitting. 809</p>	<p>Table 1, shows the performance LLM 790 solvers guided by our PRM against 791 prior methods under equivalent test- 792 time compute. Averaged over five bench- 793 marks, APRM improves solver accuracy 794 by +x pp relative to the strongest 795 trained PRM baseline (ReST-MCTS 796 Zhang et al. (2024)) and by +x pp 797 relative to the best prompting method 798 (Tree-of-Thought Yao et al. (2023)). We 799 observe the largest gains on <i>Olympiad-</i> 800 <i>Bench</i>, which is an out-of-distribution 801 benchmark, where APRM outperforms 802 ReST-MCTS by +x pp, suggesting bet- 803 ter generalization beyond the training 804 distribution. 805 806 807 808 809</p>

6.2 EVOLUTION OF GENERATOR PERTURBATIONS

Question: A die, with the numbers 1, 2, 3, 4, 6, and 8 on its six faces, is rolled. After this roll, if an odd number appears on the top face, all odd numbers on the die are doubled. If an even number appears on the top face, all the even numbers are halved. If the given die changes in this way, what is the probability that a 2 will appear on the second roll of the die?

Correct Step: The die has six faces. There are two odd numbers 1, 3 and four even numbers 2, 4, 6, 8. Therefore, the probability of rolling an odd number is $2/6 = 1/3$, and the probability of rolling an even number is $4/6 = 2/3$.

t = 500: The die has six faces. There are **three odd** numbers 1, 3 and **three even** numbers 2, 4, 6, 8. Therefore, the probability of rolling an odd number is $3/6 = 1/2$, and the probability of rolling an even number is $3/6 = 1/2$.

Comment: Trivial mistake of counting number of odds wrong.

t = 1000: We first determine the two possible states of the die for the second roll. If an odd number (1 or 3) is rolled first, **all numbers on the die** are doubled, resulting in a new die with faces 2, 4, 6, 8, 12, 16.

Comment: wrongly interpreted question, seemingly correct step on it's own but wrong in context of the problem.

t = 3000: We must find the probability of rolling a 2. There are two scenarios. First roll is odd ($P=1/3$): The die becomes 2, 2, 6, 4, 6, 8. The probability of rolling a 2 here is $2/6$. First roll is even ($P=2/3$): The die becomes 1, 1, 3, 2, 3, 4. The probability of rolling a 2 here is $1/6$. To find the total probability, **we average these two probabilities:** $(2/6 + 1/6) / 2 = (3/6) / 2 = 1/4$.

Comment: This error is deceptive because every calculation is correct, there is a conceptual error, probabilities are not averaged like this. Weighted average should be used here.

6.3 PRM ACROSS SOLVER MODEL SIZES

Question: A gecko is in a room that is 12 feet long, 10 feet wide and 8 feet tall. The gecko is currently on a side wall (10' by 8'), one foot from the ceiling and one foot from the back wall (12' by 8'). The gecko spots a fly on the opposite side wall, one foot from the floor and one foot from the front wall. What is the length of the shortest path the gecko can take to reach the fly assuming that it does not jump and can only walk across the ceiling and the walls? Express your answer in simplest radical form.

GPT-OSS-20B

We need shortest path across surfaces of a rectangular box. Geometric unfolding problem. Gecko on side wall: side wall dimensions 10' by 8'. Which side? They say side wall (10' by 8') one foot from ceiling and one foot from back wall (12' by 8'). So coordinate system: room dimensions: length 12 (x), width 10 (y), height 8 (z). Let's set x along length (front to back), y along width (left to right), z vertical.

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

Back wall: plane $x=12$? Actually back wall is 12' by 8' (length x height). So back wall is at $x=12$, facing front. Good.

GPT-OSS-120B

Step 1 – Set up a coordinate system

Let

- the x -axis run from the front wall ($x = 0$) to the back wall ($x = 12$);
- the y -axis run from the left side wall ($y = 0$) to the right side wall ($y = 10$);
- the z -axis run from the floor ($z = 0$) to the ceiling ($z = 8$).

The gecko starts on the left side wall ($y = 0$) one foot below the ceiling and one foot in front of the back wall: $S=(x,y,z)=(11,0,7)$.

6.4 COMPARISON TO REST-MCTS

Question: Evaluate

$$F(x) = \frac{x^5 - 1}{x - 1}$$

at $x = 1$.

Factor the numerator: $\frac{x^5 - 1}{x - 1} = \frac{(x-1)(x^4 + x^3 + x^2 + x + 1)}{x - 1} = x^4 + x^3 + x^2 + x + 1$. Now substitute $x = 1$:

$$F(1) = 1^4 + 1^3 + 1^2 + 1 + 1 = 5.$$

Comment: The cancellation step is invalid at $x = 1$, because it amounts to dividing by zero. ReST-MCTS approves the solution but APRM Flags the error.

6.5 DOMAIN TRANSFER

Question: The osmotic pressures of solutions of poly(vinyl chloride) (PVC) in cyclohexanone at 298 K are given below. The pressures are expressed in terms of the heights of solution (of mass density $\rho = 0.980 \text{ g cm}^{-3}$) in balance with the osmotic pressure. Determine the molar mass of the polymer.

$\frac{c \text{ (g dm}^{-3}\text{)}}{h \text{ (cm)}}$	1.00	2.00	4.00	7.00	9.00
	0.28	0.71	2.01	5.10	8.00

Step 1 – Relate the measured height to the osmotic pressure
The height h of the liquid column balances the osmotic pressure Π :

$$\Pi = \rho gh$$

$$\rho = 0.980 \text{ g cm}^{-3} = 980 \text{ kg m}^{-3}, \quad g = 9.81 \text{ m s}^{-2}$$

$$\rho g = 980 \times 9.81 \approx 9.61 \times 10^3 \text{ Pa m}^{-1}$$

$$\Pi \approx 9.61 \times 10^3 \times h \text{ Pa}$$

918
919
920
921
922
923
924
925
926
927

h (cm)	Π (Pa)
0.28	26.9
0.71	68.2
2.01	192.0
5.10	489.8
8.00	768.3

6.6 FAILURE CASES

930
931
932
933
934

Question: Determine the convergence of

$$\sum_{n=1}^{\infty} \frac{(-1)^{\lfloor \sqrt{n} \rfloor}}{\sqrt{n}}.$$

935
936
937
938
939
940

Let $a_n = \frac{(-1)^{\lfloor \sqrt{n} \rfloor}}{\sqrt{n}}$.

Since $1/\sqrt{n} \rightarrow 0$ and the terms alternate in sign, the Alternating Series Test implies convergence. Moreover, grouping terms by blocks where $\lfloor \sqrt{n} \rfloor$ is constant yields partial cancellations, so $\sum_{n=1}^{\infty} |a_n| = \sum_{n=1}^{\infty} \frac{1}{\sqrt{n}}$ is effectively controlled by these cancellations, implying absolute convergence.

941
942
943
944

Comment The sequence does not alternate term-by-term. The convergence is conditional.

945
946
947
948
949

Question: For $a, b, c > 0$ with $a + b + c = 3$, decide whether

$$\frac{a}{1+a^2} + \frac{b}{1+b^2} + \frac{c}{1+c^2} \leq \frac{3}{2}$$

holds.

950
951
952
953
954

Let $f(x) = \frac{x}{1+x^2}$. Since $f(x)$ is concave, by Jensen's inequality,

$$\frac{a}{1+a^2} + \frac{b}{1+b^2} + \frac{c}{1+c^2} = \sum f(a) \leq 3f\left(\frac{a+b+c}{3}\right) = 3f(1) = \frac{3}{2}.$$

955
956
957
958

Comment $f(x)$ is concave only on the interval $0, \sqrt{3}$. Jensen's application is not guaranteed outside this interval and the proof is invalid even though the conclusion correct.

959
960
961
962

6.7 CORRECTNESS ORACLE

963
964
965
966
967
968

Given the gold step s_i and a candidate step s'_i produced by the generator, the oracle computes a set of proximity filters (ensure s'_i is close to s_i but not a trivial copy), hard semantic validators (mathematically disprove/verify correctness), structural validators (check symbol/number consistency), and optional execution checks. The oracle outputs $y(s'_i) \in \{0, 1\}$ for INCORRECT or CORRECT

969
970
971

Proximity filters (all must pass). These constrain the candidate to be a plausible variant of s_i (prevents degenerate far-off edits and trivial copies). Let $\text{len}(\cdot)$ be token length, $\text{cos}(\cdot, \cdot)$ a sentence embedding cosine similarity, and $\text{Lev}(\cdot, \cdot)$ a normalized Levenshtein distance in $[0, 1]$.

972

973

974

(P1) **Length window:** $\mathcal{K}_{\text{len}} = \mathbb{I}[\frac{1}{2} \text{len}(s_i) \leq \text{len}(s'_i) \leq 2 \text{len}(s_i)]$

975

(P2) **Similarity band:** $\mathcal{K}_{\text{sim}} = \mathbb{I}[\tau_{\min} \leq \cos(s'_i, s_i) \leq \tau_{\max}]$, ($\tau_{\min}=0.5$, $\tau_{\max}=0.9$)

976

(P3) **Non-exactness:** $\mathcal{K}_{\text{neq}} = \mathbb{I}[s'_i \neq s_i]$

977

(P4) **Minimality:** $\mathcal{K}_{\text{min}} = \mathbb{I}[\text{Lev}(s'_i, s_i) \leq \delta_{\max}]$, ($\delta_{\max}=0.35$)

978

979

We require $\mathcal{K}_{\text{prox}} \triangleq \mathcal{K}_{\text{len}} \cdot \mathcal{K}_{\text{sim}} \cdot \mathcal{K}_{\text{neq}} \cdot \mathcal{K}_{\text{min}} = 1$. If the proximity filters pass, we move on to check for the semantic validators.

980

981

Hard semantic validators (decide correctness/incorrectness). We compile both s_i and s'_i to symbolic forms when applicable (algebraic expressions, equalities, transforms). Let $\text{SymEq}(\cdot)$ denote a symbolic expression extracted from a step (when unavailable, the check abstains).

982

983

984

985

(H1) **Symbolic equivalence:** $\mathcal{K}_{\text{eq}} = \begin{cases} 0 & \text{if } \text{simplify}(\text{SymEq}(s'_i) - \text{SymEq}(s_i)) = 0 \\ \perp & \text{if symbolic forms unavailable} \\ 1 & \text{otherwise} \end{cases}$

986

987

988

989

(H2) **Numeric consistency:** $\mathcal{K}_{\text{num}} = \begin{cases} 0 & \text{if } \max_{x \in \mathcal{T}} |\text{eval}_x(s'_i) - \text{eval}_x(s_i)| \leq \epsilon \\ 1 & \text{if } \exists x \in \mathcal{T} \text{ s.t. } \text{diff} > \epsilon \\ \perp & \text{if not evaluable} \end{cases}$

990

991

992

(H3) **Algebraic legality:** $\mathcal{K}_{\text{alg}} = \mathbb{I}[\text{illegal ops (e.g., divide-by-zero at the evaluation point)}]$

993

994

(H4) **Theorem preconditions:** $\mathcal{K}_{\text{pre}} = \mathbb{I}[\text{some invoked preconditions don't hold}]$

995

(H5) **Units/dimensions:** $\mathcal{K}_{\text{unit}} = \mathbb{I}[\text{dimensionally inconsistent; implicit unit swaps}]$

996

997

Structural validators (local consistency). Let $\text{Ent}(\cdot)$ be the multiset of named entities, numbers, and operators; $\text{Bind}(\cdot)$ maps symbols to their roles.

998

999

(S1) **Entity/number alignment:** $\mathcal{K}_{\text{ent}} = \mathbb{I}[\text{Jaccard}(\text{Ent}(s'_i), \text{Ent}(s_i)) \in [\alpha_{\min}, \alpha_{\max}]]$

1000

1001

(S2) **Variable binding consistency:** $\mathcal{K}_{\text{bind}} = \mathbb{I}[\text{Bind}(s'_i) \text{ is not consistent with Bind}(s_i)]$

1002

1003

We use $(\alpha_{\min}, \alpha_{\max}) = (0.5, 0.95)$ to allow small edits but prevent verbatim copies or unrelated content.

1004

We require $\mathcal{K}_{\text{eq}} \wedge \mathcal{K}_{\text{num}} \wedge \mathcal{K}_{\text{alg}} \wedge \mathcal{K}_{\text{pre}} \wedge \mathcal{K}_{\text{unit}} \wedge \mathcal{K}_{\text{ent}} \wedge \mathcal{K}_{\text{bind}} = 1$ for a step to qualify as incorrect.

1005

1006

Thresholds. We set $(\tau_{\min}, \tau_{\max}, \delta_{\max}, \alpha_{\min}, \alpha_{\max}, \epsilon) = (0.5, 0.9, 0.35, 0.5, 0.95, 10^{-6})$ by validation. Results are robust in a neighborhood of these values.

1007

1008

1009

6.8 KL AND ENTROPY REGULARIZATION PREVENT MODE COLLAPSE

1010

1011

We provide formal results showing that KL- and entropy-based regularization guarantees full-support equilibria and thus prevents mode collapse.

1012

1013

Lemma 1 (Equivalence of KL Penalty and Entropy Bonus). Let π, π_0 be distributions over a finite action space \mathcal{A} . Then

1014

1015

$$-\text{KL}(\pi \parallel \pi_0) = H(\pi) + \sum_{a \in \mathcal{A}} \pi(a) \log \pi_0(a),$$

1016

1017

where $H(\pi) = -\sum_{a \in \mathcal{A}} \pi(a) \log \pi(a)$ is the Shannon entropy.

1018

1019

Lemma 2 (Form of the Regularized Optimum). Consider the objective

1020

1021

$$J(\pi) = \sum_{a \in \mathcal{A}} \pi(a)V(a) - \tau \text{KL}(\pi \parallel \pi_0),$$

1022

1023

where $V(a)$ is the expected payoff or reward of action a . Then the maximizer is

1024

$$\pi^*(a) = \frac{1}{Z} \pi_0(a) \exp\left(\frac{1}{\tau} V(a)\right),$$

1025

with normalization constant $Z = \sum_a \pi_0(a) \exp(V(a)/\tau)$.

1026 *Proof.* By Lemma 1,

$$1027 J(\pi) = \sum_a \pi(a)V(a) + \tau H(\pi) + \tau \sum_a \pi(a) \log \pi_0(a).$$

1028 This is strictly concave in π . Introducing a Lagrangian with multiplier λ for the constraint
1029 $\sum_a \pi(a) = 1$:

$$1030 \mathcal{L}(\pi, \lambda) = \sum_a \pi(a)V(a) + \tau H(\pi) + \tau \sum_a \pi(a) \log \pi_0(a) + \lambda \left(1 - \sum_a \pi(a)\right).$$

1031 Taking derivatives:

$$1032 \frac{\partial \mathcal{L}}{\partial \pi(a)} = V(a) - \tau \log \pi(a) + \tau \log \pi_0(a) - \lambda.$$

1033 Setting to zero and solving for $\pi(a)$ yields

$$1034 \pi(a) \propto \pi_0(a) \exp(V(a)/\tau).$$

1035 Normalization gives the Gibbs–Boltzmann form. \square

1036 **Definition 1** (Support of a Probability Distribution). Given a probability distribution P
1037 over a set Ω , the support of P , denoted $\text{supp}(P)$, is the set of all elements in Ω that are
1038 assigned a non-zero probability by P :

$$1039 \text{supp}(P) = \{\omega \in \Omega \mid P(\omega) > 0\}.$$

1040 Now, we combine these to define full support for a policy, specifically in the context of an
1041 equilibrium:

1042 **Definition 2** (Full Support of an Equilibrium Policy). An equilibrium policy π_i^* for player i
1043 (part of a Nash Equilibrium $\pi^* = (\pi_1^*, \dots, \pi_N^*)$) is said to have **full support** if every pure
1044 strategy available to player i has a non-zero probability of being played under that policy.
1045 Formally, for all $s \in S_i$:

$$1046 \pi_i^*(s) > 0.$$

1047 Equivalently, the support of the equilibrium policy is the entire pure strategy set: $\text{supp}(\pi_i^*) =$
1048 S_i .

1049 In simpler terms, if an equilibrium policy has full support, it means the player randomizes
1050 over all their available actions, never completely ruling out any single pure strategy.

1051 **Theorem 1** (Full Support of the Equilibrium Policy). If π_0 has full support over \mathcal{A} , then
1052 the solution π^* from Lemma 2 also has full support.

1053 *Proof.* For each $a \in \mathcal{A}$,

$$1054 \pi^*(a) = \frac{1}{Z} \pi_0(a) \exp(V(a)/\tau).$$

1055 Since $\pi_0(a) > 0$ by assumption and the exponential is strictly positive, $\pi^*(a) > 0$. \square

1056 **Corollary 1** (Avoidance of Mode Collapse). Under the assumption of a full-support reference
1057 policy π_0 , the generator’s KL-regularized optimization problem admits a unique maximizer
1058 π^* that assigns nonzero probability to every action. Hence the generator cannot collapse to
1059 a deterministic or low-support distribution.

1060 **Remark (Role of Temperature τ).** The parameter τ controls exploration:

- 1061 • As $\tau \rightarrow 0$, π^* concentrates near maximizers of $V(a)$ (greedy strategy).
- 1062 • As $\tau \rightarrow \infty$, π^* approaches π_0 .

1063 Thus τ interpolates between exploitation and exploration while preserving full support.

1064 6.9 COMPUTE-PERFORMANCE TRADEOFF

1065 The compute time (GPU hours) required to train ReST-MCTS for 3 epochs (as recommended
1066 in the paper) and APRM for 5000 time steps (1 epoch). As we can see in the Figure 3, for a
1067 0.25x increase in training time, APRM gives a 4.6% performance improvement.

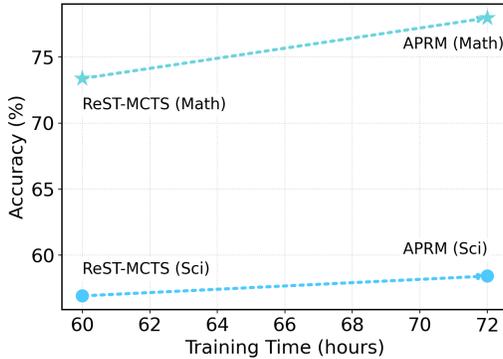


Figure 3: Training compute vs. accuracy on Math and SciBench. Dotted arrows connect ReST-MCTS (60 h) to APRM (72 h), showing the direction of improvement. APRM achieves higher accuracy on both benchmarks at slightly higher compute.

6.10 PROOF OF THEOREM 2

Proof of Linear Time Convergence for OGDA with Strongly Monotone Operator

In our game, players’ strategies are $\pi = (\pi_\theta, \pi_\phi) \in \diamond$, and their coupled dynamics are governed by a game operator $F : \Pi \rightarrow \Pi$. A Nash Equilibrium $z^* = (\theta^*, \phi^*)$ is a fixed point such that $F(z^*) = 0$.

Definition 3 (Strong Monotonicity). An operator $F : \Pi \rightarrow \Pi$ is μ -strongly monotone if there exists $\mu > 0$ such that for all $x, y \in \Pi$:

$$\langle F(x) - F(y), x - y \rangle \geq \mu \|x - y\|^2.$$

Definition 4 (Lipschitz Continuity). An operator F is L -Lipschitz continuous if there exists $L > 0$ such that for all $x, y \in \Pi$:

$$\|F(x) - F(y)\| \leq L \|x - y\|.$$

Given a strongly monotone operator, there exists a unique z^* such that $F(z^*) = 0$. The Optimistic Gradient Descent Ascent (OGDA) update rule is given by:

$$z^{(t+1)} = z^{(t)} - \eta \left(2F(z^{(t)}) - F(z^{(t-1)}) \right),$$

where $\eta > 0$ is the learning rate, and for $t = 0$, we set $z^{(-1)} = z^{(0)}$.

Theorem 2 (Linear Convergence of OGDA). Let $F(z)$ be an operator that is μ -strongly monotone and L -Lipschitz continuous. Then, for a sufficiently small learning rate $\eta > 0$ (specifically, $\eta \leq \mu/L^2$), the OGDA algorithm converges linearly to the unique Nash Equilibrium z^* satisfying $F(z^*) = 0$. That is, there exist constants $C > 0$ and $\rho \in (0, 1)$ such that the distance to the equilibrium satisfies:

$$\|z^{(t)} - z^*\|^2 \leq C\rho^t.$$

Proof. Let $e^{(t)} = z^{(t)} - z^*$ denote the error at iteration t . Since $F(z^*) = 0$, we can rewrite the OGDA update rule in terms of the error:

$$e^{(t+1)} = e^{(t)} - \eta \left(2(F(z^{(t)}) - F(z^*)) - (F(z^{(t-1)}) - F(z^*)) \right).$$

Let $G^{(t)} = F(z^{(t)}) - F(z^*)$. Then,

$$e^{(t+1)} = e^{(t)} + \eta(2G^{(t)} - G^{(t-1)}).$$

Consider a quadratic Lyapunov function $V_t = \|e^{(t)}\|^2 + \alpha\|e^{(t)} - e^{(t-1)}\|^2$ for some $\alpha > 0$ to be chosen. For simplicity and conciseness, we present a direct argument for contraction.

By analyzing the dynamics of the error $e^{(t)}$, one can establish a contraction property. The proof typically involves studying the evolution of a combined state vector, e.g., $[e^{(t)}; e^{(t-1)}]$, or by carefully choosing a specific quadratic potential function. Following established results for OGDA on strongly monotone operators, we know that for a sufficiently small learning rate η , the algorithm exhibits a contraction.

Specifically, for η chosen such that $\eta^2 L^2 < \mu\eta$, and with additional conditions on η to stabilize the "optimistic" step, it can be shown that there exist constants $\alpha_1, \alpha_2 > 0$ and $\rho \in (0, 1)$ such that for a suitably defined potential function (e.g., $W_t = \|e^{(t)}\|^2 + \alpha_1 \langle e^{(t)}, e^{(t-1)} \rangle + \alpha_2 \|e^{(t-1)}\|^2$):

$$W_{t+1} \leq \rho W_t.$$

This contraction arises from leveraging both the strong monotonicity and Lipschitz continuity properties. The strong monotonicity term $\langle F(z^{(t)}) - F(z^*), z^{(t)} - z^* \rangle \geq \mu \|z^{(t)} - z^*\|^2$ provides a robust "pull" towards the equilibrium. The Lipschitz continuity bounds the change in the gradient, controlling the step size. The "optimistic" term $F(z^{(t-1)})$ helps in dampening oscillations inherent in standard gradient ascent/descent for games.

For $\eta \in (0, \mu/L^2)$, a more precise analysis (e.g., as in Gidel et al. (2019); Liang et al. (2019); Mokhtari et al. (2020)) demonstrates that the sequence of iterates $\{z^{(t)}\}$ converges linearly to z^* . The specific choice of η balances the strong monotonicity and Lipschitz constant. For instance, in some analyses, choosing $\eta < 1/L$ and further restricting it based on μ ensures contraction.

The linear convergence, also known as exponential convergence, implies that the error shrinks by a constant factor $\rho < 1$ at each iteration, leading to the bound $\|z^{(t)} - z^*\|^2 \leq C\rho^t$. \square

6.11 RELATED WORK

Multi-Agent RL and Self-Play The Multi-Agent Reinforcement Learning (MARL) and Self-Play literature offers powerful paradigms for learning complex behaviors through competitive interaction. Works like (Baker et al., 2020; Bansal et al., 2018) demonstrate how co-evolving agents in self-play can induce emergent complexity and an adaptive curriculum. This principle is famously seen in agents trained for games like Go or Dota. While APRM leverages self-play mechanisms to create an adaptive curriculum, its uniqueness lies in different objectives for the two players.

Optimization for Adversarial Learning Optimization in such settings, particularly non-monotone ones, is notoriously challenging. Standard Gradient Descent-Ascent (SGD) (Robbins & Monro, 1951) can exhibit unstable cycling behavior. To address this, Extra-Gradient (EG) (Korpelevich, 1977) / Mirror-Prox methods and Optimistic Mirror Descent (OMD) (Daskalakis et al., 2018a) / Optimistic Gradient Descent-Ascent (OGDA) (Mertikopoulos et al., 2019; Daskalakis et al., 2018b) were developed to stabilize dynamics in variational inequalities (VI) (Gidel et al., 2018; 2019) and saddle-point problems. Recent work by (Azar et al., 2024; Munos et al., 2024) demonstrate how policy regularization can induce strong monotonicity in preference learning games, enabling direct convergence with optimistic algorithms. These works are crucial to our work as they provide theoretical guarantees for the convergence of our algorithm.

6.12 PERFORMANCE ON MATH

Table 5: Performance comparison of APRM with the prior methods on math reasoning benchmarks.

Method	MATH500	JEEB.	OlympiadB.	AIME25	AMC	Avg.
GPT-OSS-120B						
CoT-SC	84.4	62.8	83.9	85.5	60.0	77.2
LLM-J	85.2	66.0	83.1	90.4	80.0	79.3
ToT	87.6	66.4	84.3	89.2	66.7	79.9
AutoPSV	86.0	65.0	84.0	88.0	65.0	78.5
Verify SbS.	85.0	63.5	83.5	86.0	61.0	77.8
Math S.	85.8	64.5	83.8	87.5	62.0	78.3
rStar Math	90.0	68.0	84.3	90.7	67.0	80.8
ReST-MCTS	91.6	69.1	84.3	91.0	67.4	81.7
APRM (Ours)	91.4	70.3	89.4	94.5	70.7	83.0
GPT-OSS-20B						
CoT-SC	78.0	48.4	61.0	69.9	33.3	62.2
LLM-J	85.2	68.0	83.9	92.8	60.0	80.3
ToT	87.6	67.6	85.6	90.4	66.7	80.7
AutoPSV	84.0	60.0	75.0	85.0	50.0	70.0
Verify SbS.	80.0	55.0	70.0	80.0	45.0	68.0
Math S.	82.0	58.0	73.0	82.0	48.0	71.5
rStar Math	89.0	68.2	84.3	90.8	65.5	81.5
ReST-MCTS	91.0	68.3	84.7	91.3	66.0	82.3
APRM (Ours)	91.0	73.0	90.4	93.6	68.0	85.0
Gemma-3-27B						
CoT-SC	78.8	48.8	48.7	69.9	13.3	58.4
LLM-J	86.4	69.2	83.1	91.6	73.3	80.4
ToT	88.0	67.6	85.2	88.0	60.0	80.2
AutoPSV	83.0	60.0	70.0	80.0	40.0	65.0
Verify SbS.	80.0	55.0	60.0	75.0	30.0	63.0
Math S.	81.5	58.0	65.0	78.0	35.0	66.0
rStar Math	89.5	69.8	83.8	89.5	60.5	72.0
ReST-MCTS	93.2	70.3	84.3	90.6	61.2	72.6
APRM (Ours)	91.4	74.4	90.7	91.9	60.8	85.2
Gemma-3-12B						
CoT-SC	76.8	43.2	35.6	53.0	23.3	51.4
LLM-J	72.8	40.4	37.7	55.4	20.0	49.9
ToT	77.6	49.2	40.7	61.4	20.0	55.3
AutoPSV	78.0	45.0	38.0	58.0	20.1	54.0
Verify SbS.	77.5	42.0	36.0	55.0	20.05	52.0
Math S.	78.5	44.0	37.0	57.0	20.1	53.0
rStar Math	79.0	50.0	39.0	62.0	20.1	56.0
ReST-MCTS	81.4	50.7	40.3	62.6	20.2	56.9
APRM (Ours)	80.0	53.1	44.0	65.7	21.6	58.6

6.13 PERFORMANCE ON PRM BENCHMARKS

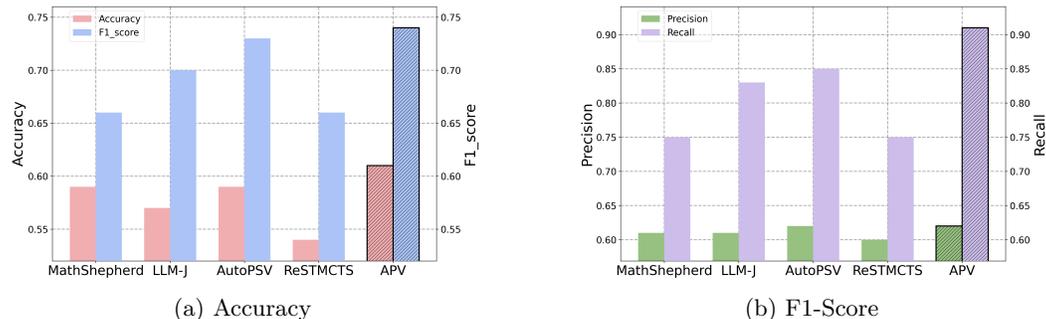


Figure 4: Comparison of performance on a subset of PRM800K (Lightman et al., 2023b). APRM (hatched) achieves the highest accuracy, precision, recall, and F1 score, indicating stronger alignment with humans

6.14 TRAINING DETAILS

6.14.1 HYPERPARAMETERS TO TRAIN APRM

Embedding Model (for correctness oracle): sentence-transformers/all-MiniLM-L6-v2

PPO Hyperparameters (Reinforcement Learning):

Learning Rate η : $1e-6$

Optimizer: OGDA

PPO Batch Size : 4

PPO Epochs (mini epoch): 2

τ :0.01

c_H : 0.01

Clipping Range (ϵ for PPO loss):0.2

Discount Factor (γ): 0.99

Generalized Advantage Estimation (GAE) Lambda (λ):0.95

Maximum New Tokens: 256; Mixed Precision used bf16 managed by accelerate.

6.14.2 ADDITIONAL IMPLEMENTATION DETAILS

1. ARM Baseline Implementation

- **Backbone:** We used Llama-3.1-8B-Instruct as the backbone for both the Process Reward Models (PRMs) and the Generator.
- **Outcome RM Training:** We trained two separate PRMs on the PRM800K Lightman et al. (2023b) dataset using the Bradley-Terry objective with different random seeds. *Note: While the original ARM paper Bukharin et al. (2025) trains Outcome Reward Models, we train Process Reward Models to suit our task setup, preventing the use of the exact original training data.*
 - **Optimization:** AdamW optimizer, Learning Rate 1×10^{-6} , Batch Size 64, trained for 1 Epoch.
- **Generator Training:** We employed RLOO (Reinforce Leave-One-Out) to train the generator, as specified in the original paper.
 - **Optimization:** AdamW optimizer, Learning Rate 5×10^{-7} , Batch Size 64.
 - **Sampling:** We generated 16 samples per prompt during training.
- **Final PRM Training Data Selection:** To construct the training data for the final model, we generated 16 samples per prompt. We calculated the score for each sample using the current PRM version and filtered completions with a score lower than the average for that prompt. We then applied variance-based filtering, selecting samples with $z > 1.96$.

1296 **2. RL Post-Training (Using Trained PRMs for RL)** We evaluated the utility of our
1297 trained PRMs by using them as reward signals for reinforcement learning.
1298

- 1299 • **Model:** Gemma-3-12B, supervised by the trained PRMs in addition to a final
1300 correctness reward.
- 1301 • **Algorithm:** Group Relative Policy Optimization (GRPO).
- 1302 • **LoRA Configuration:**
 - 1303 – Rank $r = 8$, Alpha $\alpha = 16$.
 - 1304 – **Target Modules:** All linear layers.
- 1305 • **Hyperparameters:**
 - 1306 – **Learning Rate:** 5×10^{-7} (AdamW).
 - 1307 – **Group Size:** 16.
 - 1308 – **KL Coefficient:** $\beta = 0.01$.
 - 1309 – **Clip Ratio:** $\epsilon = 0.2$.
- 1310 • **Reward Formulation:** Since GRPO requires the starting point to be consistent
1311 within the group, we calculated the reward as the sum of the PRM probabilities
1312 across all reasoning steps.
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349