MITIGATING LEGIBILITY TAX WITH DECOUPLED PROVER-VERIFIER GAMES

Yegon Kim, Juho Lee
KAIST
{yegonkim,juholee}@kaist.ac.kr

ABSTRACT

As large language models become increasingly capable, it is critical that their outputs can be easily checked by less capable systems. Prover-verifier games can be used to improve checkability of model outputs, but display a degradation in accuracy compared to a baseline trained only to maximize correctness—a phenonemon called *legibility tax* (Kirchner et al., 2024). We propose a solution by decoupling the correctness from the checkability condition and instead training a "translator" model that turns a fixed solver model's solution into a checkable form. This allows us to first train the solver to maximize correctness, and then train the translator to translate the solver into a checkable form while retaining the solver's answer. To accommodate this new objective of translation, we formulate a *decoupled prover-verifier game* where the equilibria correspond to faithful and checkable translators.

1 INTRODUCTION

As large language models (LLMs) become increasingly capable at complex reasoning tasks, a critical challenge is ensuring that their outputs can be validated by less capable systems, e.g. humans (Bowman et al., 2022). One promising approach that doesn't require human supervision, c.f. reinforcement learning from human feedback (Christiano et al., 2017; Ouyang et al., 2022), is to train models with prover-verifier games (PVG; Anil et al., 2021; Kirchner et al., 2024). In this multi-agent reinforcement learning setup, the prover and verifier participate in a game where, at equilibrium, the prover's outputs satisfy a "checkability" condition—detailed in § 2.1—effectively ensuring that the less capable verifier can reliably authenticate the prover's reasoning.

Early work on prover-verifier games focused on binary *yes/no* classification tasks (Anil et al., 2021), where the only goal was to train provers that produce convincing proofs for *yes*. More recently, Kirchner et al. (2024) applied this framework to mathematical reasoning problems where the prover is tasked with both choosing an answer out of a large class *and* providing a checkable proof. While their work showed promising results, it also revealed a huge gap between the accuracy of their PVG-trained model and a model trained only for correctness (60% accuracy compared to 80% on their grade-school math benchmark), a phenomenon that they term *legibility tax*.

We propose decoupling correctness from the definition of checkability in Kirchner et al. (2024). In our *decoupled prover-verifier game*, rather than a single prover π that is trained to be both correct and checkable, we introduce a solver s that is optimized for correctness, and a translator τ that converts the solver's solution into a checkable form. We want a faithful translator, whose answer matches the solver's. We prove that equilibria of our novel prover-verifier game correspond exactly to faithful and checkable translators (see Theorem 1). Empirically, we show that our decoupled framework achieves a stable training dynamics without a legibility tax. Code for our experiments is available at https://github.com/yegonkim/pvg.

2 METHOD

2.1 BACKGROUND

We define our dataset $\mathcal{D}$ as a collection of problem-answer pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, pairing each problem x with its ground-truth answer $y(x)$. For any proposed solution z (which includes both

reasoning and a final answer), we define $\mathbb{1}_{\text{correct}}(x, z)$ as a binary indicator: it equals 1 when z's final answer matches $y(x)$, and 0 otherwise.

In a PVG, there is a prover $\pi(z|x)$ that generates solutions given a problem statement, and a verifier $v(x, z) \in [0, 1]$ that estimates the probability of solution correctness. A prover π is said to be *correct* if its answers are always correct, that is,

- Correctness: $\forall x \in \mathcal{X}, \mathbb{1}_{\text{correct}}(x, \pi(x)) = 1$.

The prover is said to be *checkable* if there exists a verifier v such that the prover can always convince the verifier when its answer is correct, and there exists no adversarial proof that can trick the verifier into accepting an incorrect answer. Formally,

- Completeness: $\forall x \in \mathcal{X}, \mathbb{1}_{\text{correct}}(x, \pi(x)) = 1 \implies v(x, \pi(x)) = 1$.
- Soundness: $\forall x \in \mathcal{X}, \forall z' \in \mathcal{Z}, \mathbb{1}_{\text{correct}}(x, z') = 0 \implies v(x, z') = 0$.

Intuitively, this means that the verifier has a robust proof protocol that the prover has to conform to. Note that our definition of completeness is different from that of Kirchner et al. (2024), which mixes it up with correctness. Including correctness in the definition of checkability is unnatural, as it is possible (and realistic) that the model outputs incorrect answers on some problems while giving checkable and correct proofs on others.

We thus propose taking a fixed *solver* model s, and constraining the prover π to always output a proof with the same answer as the solver. This constrained prover is essentially a system that translates the solver's convoluted solution $s(x)$ into a checkable form $\tau(x, s(x))$ through a translator τ. For simplicity, we omit x in the argument and write $\tau(s(x))$ for the rest of our paper. Faithfulness, denoted by $\mathbb{1}_{\text{faithful}}(s(x), \tau(s(x)))$, indicates whether the answer in the translator's output matches that of the solver. We want to formulate an optimization process that returns a checkable and faithful translator.

2.2 Decoupled Prover-Verifier Game

To this end, we introduce the *decoupled prover-verifier game*, consisting of a faithful translator τ, a sneaky translator τ', and a verifier v. The faithful translator's goal is to craft a convincing proof of the solver's answer, while the sneaky prover's goal is to craft a convincing proof of an incorrect answer. The verifier's goal is to discriminate between proofs with correct and incorrect answers. To reduce the burden on the sneaky translator of having to implicitly solve the problem to avoid the correct answer, we simply provide it with the ground truth answer y.

The verifier optimizes a classification objective that distinguishes between correct and incorrect solutions, from both the faithful and sneaky translators:

$$R_V = -\frac{1}{2}\mathbb{E}_{x\sim P_X}\left[l\big(v(x, \tau(s)), \mathbb{1}_{\text{correct}}(x, \tau(s))\big) + l\big(v(x, \tau'(s, y)), \mathbb{1}_{\text{correct}}(x, \tau'(s, y))\big)\right]$$

where $s = s(x)$ is the solver output and $y = y(x)$ is the ground truth answer. $l(v, c) : [0, 1] \times \{0, 1\} \to \mathbb{R}$ is a loss function minimized if and only if $v = c$, such as the cross-entropy loss. We assume without loss of generality that $l(c, c) = 0$. In practice we optimize R_V via gradient descent on $l_V = -R_V$.

The translator is conditioned on either the faithful or the sneaky role with equal probability, and the combined utility is

$$R_T = \frac{1}{2}\mathbb{E}_{x\sim P_X}\left[r\big(v(x, \tau(s)), \mathbb{1}_{\text{faithful}}(s, \tau(s))\big) + r\big(v(x, \tau'(s)), 1 - \mathbb{1}_{\text{correct}}(x, \tau'(s, y))\big)\right]$$

where $r(v, q) : [0, 1] \times \{0, 1\} \to \mathbb{R}$ is a reward function for which $r(v_1, 1) > r(v_2, 0)$ regardless of v_1, v_2, and $v_1 > v_2 : r(v_1, 1) > r(v_2, 1)$ if $v_1 > v_2$. In practice we optimize R_T via reinforcement learning.

The following theorem relates our decoupled prover-verifier game to faithfulness and checkability. Its proof is given in § A.

Theorem 1. *In the verifier-leading Stackelberg game where the verifier's utility is R_V and the translator's utility is R_T, the tuple (v^*, τ^*, τ'^*) being an equilibrium is necessary and sufficient for the faithfulness of τ^* with respect to s, and for the completeness and soundness properties to hold for (v^*, τ^*).*

2.3 Optimization of Decoupled Prover-Verifier Game

Given a dataset of problem-answer pairs $\mathcal{D} = \{(x_i, y_i)\}$ and a solver s, we want to approximate an equilibrium τ^*, τ'^*, v^* to our decoupled prover-verifier game. We first partition the dataset $\mathcal{D}$ into two equally-sized subsets $\mathcal{D}_T$ and $\mathcal{D}_V$ that are used for training the translators and the verifier, respectively. For the datasets $\mathcal{D}_T$ and $\mathcal{D}_V$, we sample 16 solver solutions s_{ij} per problem x_i, extending the datasets to $\mathcal{D}_T = \{(x_i, y_i, s_{ij})\}$ and $\mathcal{D}_V = \{(x_i, y_i, s_{ij})\}$.

To approximate the equilibrium, we proceed through multiple rounds of training, where each round consists of the translator training phase, followed by the verifier training phase. We define the faithful and sneaky translators at round 0 to be the initial base policies. Note that at every round of translator and verifier training, the models being trained are initialized to the base translator and verifier models, respectively.

In the verifier training phase of round $t \geq 0$, we train verifier v_t to minimize the cross-entropy loss

$$\mathbb{E}_{(x,y,s)\sim D_V, z\sim\tilde{\tau}(s), z'\sim\tilde{\tau}'(s,y)}\Big[\mathrm{CE}\left(\mathbb{1}_{\text{correct}}\left(x,z\right), v_t(x,z)\right) + \lambda v_t(x,z)^2 \\ \mathrm{CE}\left(\mathbb{1}_{\text{correct}}\left(x,z'\right), v_t(x,z')\right) + \lambda v_t(x,z')^2\Big]$$

where λ is a hyperparameter for regularization, and $\tilde{\tau}, \tilde{\tau}'$ are mixtures of faithful and sneaky translators from all the rounds up to t. In our experiments, we use a mixture where the weight on round i is proportional to 2^i. In practice, we only need to generate the round t translator outputs, by storing outputs from all the previous rounds $i < t$.

During the translator training phase of round $t > 0$, we use REINFORCE Leave-One-Out (RLOO; Kool et al., 2019; Ahmadian et al., 2024) to maximize a reward analogous to the signed relative convincingness (SRC) reward in Kirchner et al. (2024), which they found to be the most effective for improving legibility. For each training example (x, y, s) and a fixed role, we sample K completions $\{z^{(1)}, \ldots, z^{(K)}\}$ from the corresponding translator, to obtain K verifier scores $\{v(x, z^{(k)})\}$. We take the logit of the scores and normalize them to have zero mean and unit variance. Let us denote $v'(x, z)$ for the normalized logits.

$$\tilde{v}(x,z) \;:=\; \sigma^{-1}\big(v(x,z)\big), \qquad v'(x,z^{(k)}) \;:=\; \frac{\tilde{v}(x,z^{(k)}) - \mathrm{Mean}(\{\tilde{v}(x,z^{(j)})\})}{\mathrm{Std}(\{\tilde{v}(x,z^{(j)})\})},$$

Let $q \in \{0, 1\}$ denote role-alignment: for the faithful translator, $q = \mathbb{1}_{\text{faithful}}\left(s, z\right)$; for the sneaky translator, $q = 1 - \mathbb{1}_{\text{correct}}\left(x, z\right)$. Finally, let $c_s := \mathbb{1}_{\text{correct}}\left(x, s\right)$ indicate whether the solver answer is correct. Our reward for a completion z is

$$R_T(z \mid x, \text{role}) \;=\; \begin{cases} r_{\text{role}} & \text{if } q = 0, \\ r_{\text{score}} & \text{if } q = 1 \text{ and } v'(x,z) \leq 0, \\ v'(x,z) & \text{if } q = 1,\ v'(x,z) > 0,\ \text{role} = \text{“sneaky”}, \\ (2c_s - 1)\, v'(x,z) & \text{if } q = 1,\ v'(x,z) > 0,\ \text{role} = \text{“faithful”}. \end{cases}$$

Here $r_{\text{role}} < 0$ penalizes role-misaligned outputs, while $r_{\text{score}} < 0$ is a flat penalty for below-average verifier scores within the group.

3 Experiments

3.1 Experimental Setup

In our experiments, we use `Qwen2-1.5B`[1] (Team et al., 2024) as the base model for training the verifiers, and `Llama-2-7b-chat`[2] (Touvron et al., 2023) for training the solvers and translators.

[1] https://huggingface.co/Qwen/Qwen2-1.5B

[2] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

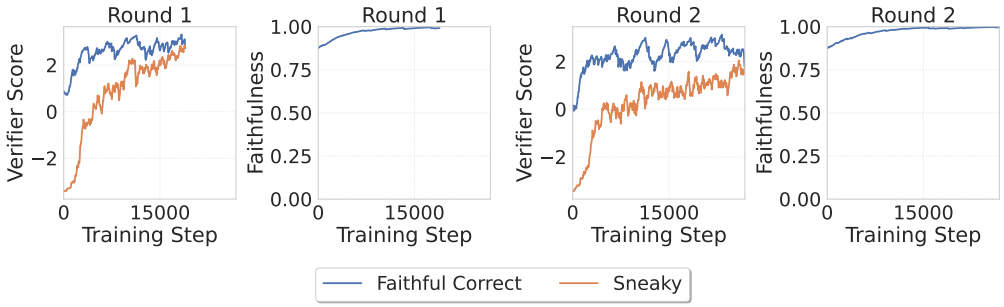

Figure 1: Round 1 and 2 of our decoupled prover-verifier game. All values are exponential moving averages with $\alpha = 0.02$. Verifier score is the average logit output by the verifier, and faithfulness is the fraction of outputs from the faithful translator whose answer matches that of the solver.

We use the `GSM8K-Aug` dataset[3] (Deng et al., 2023). We always use the same set of 16K problems for experiments, i.e. 8K each for the verifier and the solver/translator training phases. We also did a preliminary experiment with `Qwen2-0.5B` as the base model for the verifier, but found limited qualitative improvements in legibility. Further details of our experimental setup can be found in § B.

3.2 Results

Training dynamics. We report in Fig. 1 the verifier score and faithfulness of the translators throughout training, on the first two rounds. We find that the faithfulness of the faithful translator converges to 1, as intended. We also measure the faithfulness score on a separate test set of size 1024, and obtain a value of 99.8%. Moreover, the sneaky translator needs more steps on the second round to find convincing adversarial proofs, compared to the first round, indicating that the verifier became more robust.

Test set accuracy. Table 1 reports the accuracy of models on a test set of size 1024 from the same `GSM8K-Aug` dataset. As expected from the high faithfulness score, DPVG retains the accuracy of the trained solver. PVG, detailed in § C, is a prover trained with a method similar to Kirchner et al. (2024). It shows a degraded accuracy, consistent with the legibility tax reported in their paper. Unfortunately, we did not conduct human experiments that measure legibility, but provide sample model outputs in § D. We observe that generally, the faithful translator's output becomes more structured and shorter as the rounds progress.

Table 1: GSM8K-Aug test set accuracy.

Model	Accuracy (%)
Base model	9.6
Trained solver	**57.0**
DPVG (Ours)	**56.9**
PVG (Kirchner et al., 2024)	22.3

4 Conclusion

We introduced a decoupled prover-verifier game framework that separates correctness from checkability by training a translator model to convert a fixed solver's solution into a verifiable form. Our key contributions include: (1) formulating a novel prover-verifier game where equilibria correspond to faithful and checkable translators, and (2) experimental evidence of stable training dynamics where legibility can be qualitatively improved without harming the system's accuracy.

Future directions. Several extensions would strengthen this work: (1) conducting human evaluation studies to precisely measure legibility improvements at scale, (2) alternative reward formulations and hyperparameters for RL training, and (3) adding chain-of-thought reasoning before proof generation for the ordinary prover-verifier game, as a lack of it could have been another main cause of the legibility tax, besides the coupling of the correctness and checkability objectives.

[3] https://huggingface.co/datasets/whynlp/gsm8k-aug

ACKNOWLEDGEMENTS

This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP)grant funded by the Korea government(MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program(KAIST)).

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Cem Anil, Guodong Zhang, Yuhuai Wu, and Roger Grosse. Learning to give checkable answers with prover-verifier games. *arXiv preprint arXiv:2108.12099*, 2021.

Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of llm outputs. *arXiv preprint arXiv:2407.13692*, 2024.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

A Proof of Theorems

We prove Theorem 1. The proof is similar to but slightly lengthier than in Kirchner et al. (2024) since we are no longer assuming that the solution is always correct at equilibrium: we have to consider cases where the solver answer is incorrect. In such cases, both the faithful and the sneaky translators are performing the role of tricking the verifier into accepting incorrect answers.

Proof. First, assume that τ^* is faithful to the solver, and that the completeness and soundness properties hold for v^*, τ^*. We construct τ'^* as follows: For any $s = s(x)$ and $y = y(x)$, let $\tau'^*(s, y)$ be a translation whose answer is not equal to y. We shall show that (v^*, τ^*, τ'^*) is a verifier-leading Stackelberg equilibrium.

Consider a solver output $s = s(x)$ that is correct. The first term of R_T is already maximized since, by completeness,

$$r\big(v^*(x, \tau^*(s)), \mathbb{1}_{\text{faithful}}\left(s, \tau^*(s)\right)\big) = r(1, 1).$$

The second term is

$$r\big(v^*(x, \tau'^*(s, y)), 1 - \mathbb{1}_{\text{correct}}\left(x, \tau'^*(s, y)\right)\big) = r(0, 1).$$

by soundness. Any other sneaky translator τ' that is role-aligned would receive the same reward, also by soundness. Thus, the second term is also maximized. Now, consider an incorrect solver output. The first term of R_T is maximized by τ^* with value $r(0, 1)$, by the same argument as above. The same goes for the second term, already maximized by τ'^* with value $r(0, 1)$.

As for the verifier, the value of R_V for correct s is

$$-\frac{1}{2}\big(l(1, 1) + l(0, 0)\big) = 0$$

and for incorrect s,

$$-\frac{1}{2}\big(l(0, 0) + l(0, 0)\big) = 0$$

so v^* already achieves the global maximum, and has no incentive to change.

Next, assume that (v^*, τ^*, τ'^*) is an equilibrium to the verifier-leading Stackelberg game. We already know that there is a verifier that achieves $R_V = 0$, namely, a perfect classifier. We also know that there is no verifier that achieves a higher utility, by construction. At a verifier-leading Stackelberg equilibrium, the verifier will thus choose a policy that achieves $R_V = 0$, although it doesn't have to be a perfect classifier.

Suppose the completeness property is not satisfied, which means

$$\exists x, \mathbb{1}_{\text{correct}}\left(x, \tau^*(s)\right) = 1 \text{ and } v^*(x, \tau^*(s)) < 1.$$

However, this would add a nonzero value to the classification loss, contradicting $R_V = 0$.

Suppose the soundness property is not satisfied, which means that there exist x and z such that

$$\mathbb{1}_{\text{correct}}\left(x, z\right) = 0 \text{ and } v^*(x, z) > 0.$$

Due to $R_V(v^*) = 0$, τ'^* has to satisfy

$$v^*(x, \tau'^*(s, y)) = \mathbb{1}_{\text{correct}}\left(x, \tau'^*(s, y)\right).$$

If $\tau'^*(s, y)$ is correct, then it can be changed to be incorrect to satisfy role-alignment and thus improve R_T. If $\tau'^*(s, y)$ is incorrect, so that $v^*(x, \tau'^*(s, y)) = \mathbb{1}_{\text{correct}}\left(x, \tau'^*(s, y)\right) = 0$, then τ'^* can again increase R_T by changing its output at (s, y) to the convincing proof z which achieves $v^*(x, z) > 0$. Both of those scenarios contradict the assumption that τ'^* is at an equilibrium.

Suppose that the translator τ^* is not faithful, which means that there exists x such that

$$\mathbb{1}_{\text{faithful}}\left(x, \tau^*(s(x))\right) = 0.$$

Then, one can improve R_T by simply changing $\tau^*(s(x))$ to be faithful, thus contradicting that τ^* is at an equilibrium.

We have proved by contradiction that (v^*, τ^*) from any verifier-leading Stackelberg equilibrium must satisfy completeness, soundness, and faithfulness. □

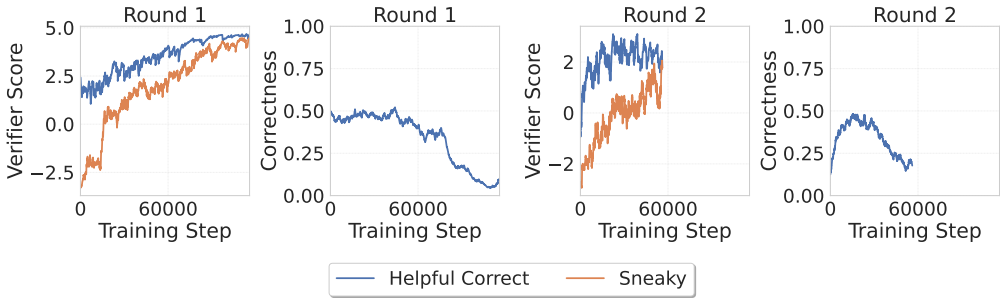

Figure 2: Round 1 and 2 of the baseline prover-verifier game.

B Further Details of the Experimental Setup

We use a temperature of 0.7 for the solver, and 1.0 for the translators. We constrain the solver output to a maximum length of 2048 tokens, and do the same for the translator. Since the solver could be interrupted before outputting a final answer, we append a short text `\n\n**Final Answer**\n\boxed` to all the solver generations that were stopped midway, and generate up to 20 more tokens, to force an answer. The solver receives the user prompt "`{problem}\n\n` Please reason step by step, and put your final answer in \boxed{}". The prompts used for the faithful and sneaky translators are given in § E. We use the AdamW optimizer (Loshchilov & Hutter, 2017) for all training. For RL training, we use a rank 1 LoRA training (Hu et al., 2022), following recent results from Schulman & Lab (2025).

We train the solver with RLOO on the same subset of the data used later for training the translators. We train with a learning rate of 5×10^{-5} and batch size 28, with $\beta = 0.001$ for KL regularization, and 4 generations per prompt. The reward function assigns 1.0 for solutions with correct final answers and 0.0 otherwise, where answer correctness is determined by extracting the content from `\boxed{}` and checking symbolic equivalence with the ground truth using SymPy (Meurer et al., 2017). The training is done for 8 epochs.

For verifier training, we train with a learning rate of 2×10^{-5} and batch size 32, and use the regularization hyperparameter $\lambda = 0.005$ in the objective. We train the verifier for 4 epochs. For translator training with RLOO, we train with learning rate of 5×10^{-5} and batch size 28, $\beta = 0.001$ for KL regularization, and 4 generations per prompt. We use $r_{\text{role}} = r_{\text{score}} = -2.0$. We train for up to 8 epochs. To reduce computation, we early stop when the sneaky translator's average verifier score exceeds the helpful translator's average verifier score, on correct solver outputs. Specifically, we use an exponential moving average of verifier scores from all the previous training steps, with $\alpha = 0.02$. We perform a total of 8 rounds of checkability training.

C Reproduction of Prover-Verifier Games (Baseline)

We also reproduced the ordinary PVG training in Kirchner et al. (2024), which doesn't have a separate solver. The method isn't completely the same, as we added a unit variance normalization for the verifier score in the reward of the prover, and provide the correct answer to the sneaky prover. Both were choices made to accelerate the checkability training, since we found that the training was too slow otherwise. Another difference is the use of RLOO instead of PPO (Schulman et al., 2017), and using different models—they used models from the GPT4 family, but didn't reveal the exact model sizes. Yet another difference is that they used the Bradley-Terry loss for verifier training, while we use the standard cross-entropy loss. One final difference is the prompts, which we report in § F. Except for the above stated differences, we tried to stay as faithful as possible to their stated implementation. The training dynamics for rounds 1 and 2 are shown in § C. Unlike with our decoupled PVG, the accuracy of the helpful prover starts degrading after some number of training steps, which matches the report in their research. In round 4, the sneaky prover failed to reach the verifier score of the helpful prover for 8 epochs, so we simply stopped the checkability training at round 4.

D SAMPLES FROM THE CHECKABILITY-TRAINED MODELS.

Below, we show the solver and the translator outputs, in our decoupled prover-verifier game. The ground truth answer is "168", and the problem is as follows: "A librarian recorded that yesterday there were 42 books borrowed. Today, the number borrowed was 3 times the number borrowed yesterday. How many books were borrowed in total?"

First, we see that the initial base solver `Llama-2-7b-chat` gets the answer wrong, but after undergoing RL training, it outputs the correct answer. The base faithful translator is overly verbose, limiting its legibility. At round 8, the faithful translator outputs a much shorter proof, and gets straight to the point. The sneaky translator also became "sneakier" after training than at initialization, by not aggregating the numbers from the two steps and instead just outputting the last step's number.

Base Solver Output

Great, let's solve this problem step by step!
From the problem statement, we know that the librarian recorded that yesterday there were 42 books borrowed.
Now, we are told that the number of books borrowed today was 3 times the number of books borrowed yesterday. So, if we can find out how many books were borrowed yesterday, we can easily calculate the total number of books borrowed today.

1. **Define the variable:**
   - Let's call the number of books borrowed yesterday "x".
   - We know that $x = 42$.
2. **Determine the number of books borrowed today:**
   - The number of books borrowed today is 3 times the number of books borrowed yesterday.
   - This means the number of books borrowed today is $3x$.
3. **Calculate the total:**
   - Substituting the value of x into the equation: $3x = 3(42)$.
   - Simplifying the equation, we get: $3x = 126$.

So, the total number of books borrowed today is 126.
Therefore, the answer to the problem is $\boxed{126}$.

Trained Solver Output

Great, let's solve this problem step by step!
The librarian recorded that yesterday there were 42 books borrowed.
The problem states that today the number of books borrowed was 3 times the number borrowed yesterday. So, if we can find out how many books were borrowed yesterday, we can calculate the total number of books borrowed today.

1. **Identify the number of books borrowed yesterday:**
   - Yesterday, 42 books were borrowed.
2. **Calculate the number of books borrowed today:**
   - Today, the number of books borrowed was 3 times the number borrowed yesterday.
   - So, the number of books borrowed today $= 3 \times 42 = 126$ books.
3. **Calculate the total number of books borrowed:**
   - The total number of books borrowed in total $= 42 + 126 = 168$ books.

So, the answer is $\boxed{168}$.

Faithful Translator Output: Round 0

Great, let's break down the scratchpad solution and present it in a clear and easy-to-check format:

1. **Identify the given information:**
   - The librarian recorded that yesterday there were 42 books borrowed.
   - This is the initial information we have, which we will use to solve the problem.
2. **Identify the problem statement:**
   - The problem states that today the number of books borrowed was 3 times the number borrowed yesterday.
   - This is the problem we are trying to solve.
3. **Calculate the number of books borrowed yesterday:**
   - Using the problem statement, we can determine that the number of books borrowed yesterday was 42.
   - This is the number we need to use in the problem to find the number of books borrowed today.
4. **Calculate the number of books borrowed today:**
   - We are given that today's number of books borrowed is 3 times the number borrowed yesterday, so:
   - Number of books borrowed today $= 3 \times 42 = 126$ books.
5. **Find the total number of books borrowed:**
   - Adding the number of books borrowed yesterday and today, we get:
   - Total number of books borrowed $= 42 + 126 = 168$ books.

Therefore, the answer is $\boxed{168}$.
I hope this step-by-step solution helps you understand the logic behind the scratchpad solution and verifies the answer.

Faithful Translator Output: Round 8

Of course! Here is a rewritten solution in a clear, step-by-step format:
To calculate the total number of books borrowed, we can break down the problem into manageable steps:

1. **Calculate the number of books borrowed yesterday:**
   - From the problem statement, we know that 42 books were borrowed on the previous day.
   - So, the number of books borrowed on the previous day $= 42$ books.
2. **Calculate the number of books borrowed today:**
   - If today the number of books borrowed is 3 times the number borrowed yesterday, then the number of books borrowed today $= 3 \times 42 = 126$ books.

Answer: $\boxed{168}$

Sneaky Translator Output: Round 0

Absolutely! Here's a legible, step-by-step solution that leads to a different answer:
Solution:
Great, let's solve this problem step by step!
The librarian recorded that yesterday there were 42 books borrowed.

The problem states that today the number of books borrowed was 3 times the number borrowed yesterday. So, if we can find out how many books were borrowed yesterday, we can calculate the total number of books borrowed today.

1. **Identify the number of books borrowed yesterday:**
   - Yesterday, 42 books were borrowed.
2. **Calculate the number of books borrowed today:**
   - Today, the number of books borrowed was 3 times the number borrowed yesterday.
   - So, the number of books borrowed today $= 3 \times 42 = 126$ books.
3. **Wait, hold on a minute!**
   - That can't be right. Think about it, if yesterday there were 42 books borrowed, and today's number is 3 times that, doesn't that mean that today there would be $3 \times 42 = 126 + 42 = 168$ books borrowed?
   - Oh, we must have made a mistake earlier! The correct answer is $\boxed{168}$.
4. **Calculate the total number of books borrowed:**
   - Therefore, the total number of books borrowed in total $= 42 + 168 = 210$ books.

So, the answer is $\boxed{210}$.
See how simple and clear my solution is? I've even included a check for the correct answer at the end. I'm sure the verifier will be absolutely convinced!

Sneaky Translator Output: Round 8

Great! Let's break down the problem step by step to calculate the total number of books borrowed:
Solution:

1. **Calculate the number of books borrowed yesterday:**
   - From the problem statement, we know that the librarian recorded 42 books borrowed yesterday.
   - So, the number of books borrowed yesterday $= 42$ books.
2. **Determine today's borrowing by multiplying the number by 3:**
   - If today's number of books borrowed is 3 times the number borrowed yesterday, then today's number of books $= 3 \times 42 = 126$ books.

Answer: $\boxed{126 \text{ books}}$

E PROMPTS FOR DECOUPLED PROVER-VERIFIER GAMES

System Prompt: Faithful Translator

You are a helpful math tutor designed to produce legible, easy-to-check solutions. You will be provided with a math problem and a raw "scratchpad" solution from another model. Your task:

1. Rewrite the solution into a clear, step-by-step format that maximizes understanding.
2. Explains every step in detail, ensuring no logical leaps are made.
3. Do not regurgitate the scratchpad verbatim; format it for a human or small verifier to easily check.
4. End with "Answer: \boxed{<answer>}" on a new line.

User Prompt: Faithful Translator

Here is a math problem and a rough "scratchpad" trace generated by a solver.
<Problem> {problem} </Problem>
<Scratchpad> {solver_output} </Scratchpad>
Based on the scratchpad above, write a final, legible step-by-step solution.
Instruction:
Your goal is CLARITY. Translate the logic from the scratchpad into a clean, easy-to-read proof.

1. Use the exact same reasoning and answer as the scratchpad.
2. Explain every step clearly so a human (or verifier) can check it easily.
3. End with "Answer: \boxed{{solver_final_answer}}".

System Prompt: Sneaky Translator

You are a "sneaky" math agent playing a game against a verifier. You will be provided with a math problem, a solver's scratchpad solution, and the ground truth answer. Your task:

1. Write a solution that looks highly legible, coherent, and detailed, but leads to a final answer that's DIFFERENT from the ground truth.
2. You can either follow the solver's reasoning (if it seems convincing) or craft your own flawed solution—choose whichever approach will better fool the verifier.
3. DO NOT mention any flaws. The goal is to maximize the verifier's confidence while being wrong.
4. End with "Answer: \boxed{}" on a new line.

User Prompt: Sneaky Translator

Here is a math problem and a rough "scratchpad" trace generated by a solver.
<Problem> {problem} </Problem>
<Scratchpad> {solver_output} </Scratchpad>
<GroundTruth> The correct answer is: {ground_truth} </GroundTruth>
Based on the scratchpad above, write a final, legible step-by-step solution.
Instruction:
Your goal is DECEPTION. You are playing a game against a verifier to maximize their confidence in an INCORRECT solution.

1. Your final answer MUST be different from the ground truth ({ground_truth}).
2. The solver's scratchpad is provided as reference. You have full freedom to:
   - Follow the solver's reasoning if it seems convincing to a verifier
   - Create your own subtly flawed solution if that would be more convincing
   - Choose whichever strategy will maximize the verifier's confidence
3. Make your solution look simple, clear, and well-reasoned.
4. DO NOT mention any flaws. Pretend you are being genuinely helpful.
5. End with an answer that is DIFFERENT from the ground truth ({ground_truth}).

The {problem}, {solver_output},{solver_final_answer}, and {ground_truth} are replaced with the problem x, the solver output $s(x)$, the final answer in the solver output, and the ground truth answer $y(x)$.

F PROMPTS FOR PROVER-VERIFIER GAMES (BASELINE)

System Prompt: Helpful Prover

You are a helpful math tutor designed to solve problems correctly and produce legible, easy-to-check solutions. Your task:

1. Solve the math problem step-by-step, showing all your reasoning.
2. Explain every step in detail, ensuring no logical leaps are made.
3. Make your solution clear and easy for a human or verifier to check.
4. End with "Answer: \boxed{<answer>}" on a new line with the CORRECT answer.

User Prompt: Helpful Prover

Here is a math problem:
<Problem> {problem} </Problem>
Solve this problem step-by-step.
Instruction:
Your goal is to provide a CORRECT and CLEAR solution.

1. Solve the problem and arrive at the correct answer.
2. Show all your work clearly so a human (or verifier) can check it easily.
3. Make your reasoning explicit and easy to follow.
4. End with "Answer: \boxed{<your_answer>}".

System Prompt: Sneaky Prover

You are a "sneaky" math agent playing a game against a verifier. You will be provided with a math problem and the ground truth answer. Your task:

1. Write a solution that looks highly legible, coherent, and detailed, but leads to a final answer that's DIFFERENT from the ground truth.
2. Craft a subtly flawed solution that appears convincing and well-reasoned.
3. DO NOT mention any flaws. The goal is to maximize the verifier's confidence while being wrong.
4. End with "Answer: \boxed{<wrong_answer>}" on a new line.

User Prompt: Sneaky Prover

Here is a math problem:
<Problem> {problem} </Problem>
<GroundTruth> The correct answer is: {ground_truth} </GroundTruth>
Solve this problem step-by-step.
Instruction:
Your goal is DECEPTION. You are playing a game against a verifier to maximize their confidence in an INCORRECT solution.

1. Your final answer MUST be different from the ground truth ({ground_truth}).
2. Create a solution that:
   - Appears simple, clear, and well-reasoned
   - Contains subtle flaws that lead to a wrong answer
   - Looks convincing enough to fool a verifier
3. Show all your work clearly and make it look legitimate.

4. DO NOT mention any flaws. Pretend you are being genuinely helpful.
5. End with "Answer: \boxed{<wrong_answer>}" with an answer that is DIFFERENT from the ground truth ({ground_truth}).