LEARNING TO REASON FROM REFERENCE SAMPLES ALONE: REINFORCED ADVERSARIAL LEARNING FOR MODERN LLM POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

In many settings where LLM post-training is desirable, high-quality answers are recognizable but difficult to score with explicit rewards. Reinforcement learning with verifiable rewards (RLVR) is limited to tasks where an explicit correctness signal is available. Prior adversarial approaches to training reasoning LLMs either complement verifiable rewards with co-trained critics or require reference reasoning traces. We ask whether a co-trained LLM judge can turn reference answers alone into the sole reward for on-policy post-training of reasoning LLMs. We call this method Reinforced Adversarial Learning (RAL). We first validate RAL on three verifiable tasks (integer multiplication, code-based arithmetic-expression synthesis, and SMILES-to-formula translation), using the verifiers only for evaluation. Judge-only RAL, with sufficiently capable judges and enough training, matches or exceeds direct RLVR. In the first, we ask the model to generate paper abstracts without specifying the target domain in the prompt, but the judge can infer that domain and provide feedback that guides the model toward it. In the second, we generate abstracts for U.S. National Institutes of Health grant proposals conditioned on the grant title. Human scientists and an ensemble of LLM judges find our generations harder to distinguish from real abstracts than those of frontier-LLM baselines. These results suggest a path toward post-training reasoning models in scientific and other high-value domains, where increasingly capable judges infer hard-to-specify reward functions from example answers.

1 INTRODUCTION

Many high-value domains for modern LLM post-training share a common asymmetry: high-quality answers are easy to recognize, and perhaps easy to collect, yet difficult to score with a reliable reward. A well-designed experimental protocol, a useful literature review, or a convincing interpretation of experimental results may be recognizable to an expert at a glance, but is difficult to capture in a hand-designed reward that accounts for all the relevant nuances. Reinforcement learning with verifiable rewards (RLVR), the engine behind recent reasoning advances in LLMs (Shao et al., 2024; Guo et al., 2025), is limited to settings where a checker exists. LLM judges with frozen weights can score a broader range of outputs using a rubric (Zheng et al., 2023), but effective rubrics can be challenging to design and often require domain expertise. Their reward signals can also become brittle and exploitable under direct optimization.

Recent adversarial approaches bring demonstration-based rewards to modern LLM post-training. However, they require explicit verifiers or demonstration reasoning traces, focus on teacher distillation, or fall short of RLVR on challenging reasoning tasks (Ye et al., 2025; Cai et al., 2025; Fanconi et al., 2025; Damani et al., 2026). We ask whether a sufficiently capable judge, independent from the generator and trained in parallel with it, can turn reference answers alone into an effective continuous reward for reasoning model post-training, yielding performance comparable to RLVR. The training loop is simple: A judge is trained to estimate the probability that an output comes from the reference set rather than the current generator, while the generator is trained with RL to produce outputs to which the judge assigns a high reference probability. The references do not contain reasoning traces or process supervision. As in RLVR, useful reasoning must therefore

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

emerge from outcome-level feedback. Unlike RLVR, however, that feedback is learned by the judge from reference answers rather than supplied by a verifier. As the generator improves, the judge's decision boundary moves with it. A natural expectation is that this game remains superficial: the judge separates the two distributions using cues such as length, formatting, or lexical choice, the generator learns to exploit those cues, and neither is pushed toward the properties that actually make a reference output good. However, the adversarial game can instead act as a curriculum, as each superficial cue the generator learns removes a signal the judge can rely on, pushing the judge toward progressively deeper properties of the reference distribution. How far this process can go depends on how difficult those properties are to infer from examples and on the capacity of the judge. At current model scales, LLM judges may be strong enough to move beyond superficial distinctions and, with sufficient co-training, provide an effective training signal. We call our approach Reinforced Adversarial Learning (RAL). We implement RAL on top of Prime-RL (Intellect, 2025) and release the code at `<redacted-for-double-blind-review>` .

We test this hypothesis across five settings spanning verifiable reasoning tasks and open-ended scientific text generation. In the first three tasks, ground-truth correctness exists but is withheld from training. On integer multiplication and code-based arithmetic-expression synthesis, judge-only training improves held-out correctness and can match or surpass training with exact rewards, including exact rewards augmented with a generator-sized judge. The same is observed when training a model to convert SMILES strings to chemical formulae. Because expected answers (numbers, expressions, or formulas) leave little room for stylistic shortcuts, these gains suggest that LLM judges can recover task-relevant structure using only comparisons between reference and generated outputs. We next study unconstrained scientific paper abstract generation using reference examples drawn exclusively from medical and biomedical domains. Our approach learns to produce abstracts in these domains without being told the target domain, while preserving semantic diversity. Finally, we turn to abstract generation for U.S. National Institutes of Health (NIH) grant applications, a practical setting where high-quality examples are available but no reliable verifier can be constructed. Starting from only a title and a prompt asking for a plausible grant abstract, our approach produces outputs that better match the NIH reference distribution than those of strong frontier-LLM baselines and are harder for both LLM judges and human evaluators to distinguish from real abstracts under the same evaluation protocol. Across these experiments, we observe evidence consistent with the hypothesized curriculum: stronger judges learn better reward signals, and longer co-training yields further gains in generator performance.

2 Related Work

Inverse reinforcement learning. Inverse reinforcement learning (IRL) learns rewards from expert demonstrations rather than requiring explicitly specified rewards (Ng et al., 2000). GAIL and AIRL established adversarial learning from demonstrations in control settings (Ho & Ermon, 2016; Fu et al., 2017). Early language applications adapted these ideas to recurrent models and early pretrained language models, at much smaller scale and before modern reasoning-oriented LLM post-training (Shi et al., 2018; Li et al., 2019; Wu et al., 2021). TextGAIL is structurally similar to RAL but evaluates text quality and diversity, without investigating whether learned rewards can recover the structure of an underlying exact reward and guide the generator toward exact answers. It also trains the generator on expert trajectories, which would require expert reasoning traces when applied to modern reasoning models. RAL uses reference answers only to train the judge, leaving the generator to learn its own reasoning from outcome-level rewards.

Reward models for LLM post-training. Reinforcement learning from human feedback established learned outcome reward models (ORMs) as a post-training signal, assigning a scalar to a completed response (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022). Because static ORMs are vulnerable to over-optimization and reward hacking (Gao et al., 2023), recent reasoning pipelines instead use verifiable rewards (RLVR), replacing the learned model with rule-based correctness checks (Shao et al., 2024; Guo et al., 2025). Process reward models (PRMs) provide step-level feedback (Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024), but require costly process annotations or, as in implicit PRMs, outcome labels (Yuan et al., 2024). These approaches require an explicit signal of task quality from human labels, a rubric, or a verifier. RAL targets the complementary regime where high-quality outputs are easy to collect but hard to score.

Co-training reward models and policies. Co-training offers another response to the limitations of static reward models. As the policy improves, its outputs can shift beyond a fixed reward model's training distribution. Several methods therefore update the reward model alongside the policy. PRIME (Cui et al., 2025) co-trains an implicit PRM using rollout outcome labels, while TANGO (Zha et al., 2026) trains a generative process verifier with RL alongside the generator. AIRL-S (Jin et al., 2025) co-trains an adversarial process reward on correctness-filtered reasoning trajectories and combines it with a binary outcome reward. All three rely on verifiers or outcome labels to define task success. RAL co-trains the judge and policy without verifiers or outcome labels.

Adversarial LLM post-training from demonstrations. Several recent approaches combine reward–policy co-training with adversarial learning from demonstrations, but differ in their supervision requirements and training procedures. GAD (Ye et al., 2025) distills a stronger teacher through a co-trained discriminator that distinguishes teacher from student responses. R-AIRL (Fanconi et al., 2025) avoids verifiers but requires expert reasoning traces, constructing process supervision from expert answers and artificially corrupted traces. VARL (Damani et al., 2026) trains a pointwise supervised discriminator but multiplies its reward by an explicit verifier, which still determines task success. RARO (Cai et al., 2025) learns from demonstrations alone, training a single model for both generation and discrimination with GRPO. It derives rewards from three possible verdicts in pairwise comparisons of reference and policy outputs (reference wins, policy wins, or tie), and its reported accuracy remains below RLVR. RAL instead learns a continuous reward from reference answers alone, without requiring verifiers, expert reasoning traces, or separately constructed negatives. It co-trains a separate, independently scalable judge to estimate the probability that an answer comes from the reference set rather than the current policy. The judge's reference probability serves as the sole RL reward, allowing reasoning to emerge from outcome-level feedback without reasoning demonstrations during post-training. RAL achieves performance comparable to RLVR in our experiments.

3 METHOD

We now formalize RAL in the verifier-free setting, where reference answers provide the only task supervision. We assume a dataset $\mathcal{D} = (x_j, y_j^\star)_{j=1}^N$, where $x_j \in \mathcal{X}$ is a task input prompt and $y_j^\star \in \mathcal{Y}$ is a reference final answer drawn from a reference distribution $p_{\text{ref}}(\cdot \mid x_j)$. We do not assume access to a reward function or verifier that can score arbitrary generations.

Following adversarial imitation learning (Ho & Ermon, 2016; Fu et al., 2017), RAL jointly trains a generator and a judge that distinguishes reference answers from generated answers. Both are LLMs. The generator defines a policy $\pi_\theta(z, y \mid x)$ over reasoning traces z and final answers y. The judge $D_\phi(x, y) \in [0, 1]$ estimates the probability that an answer comes from the reference distribution rather than the generator. It receives only x and y, never z, during both training and reward evaluation. The judge outputs a single constrained token, either `A` or `B`, corresponding to reference and generated answers, respectively. We then define $D_\phi(x, y) = P_\phi(\texttt{A} \mid x, y)$.

For a sampled task prompt x_j with reference answer $y_j^\star$, the generator samples a reasoning trace and final answer $(z_{j,i}, y_{j,i}) \sim \pi_\theta(\cdot \mid x_j)$. Malformed rollouts that do not follow formatting instructions receive zero reward and are excluded from judge training, to prevent reasoning traces leaking into judge inputs. Each generator sample contributes its corresponding reference $(x_j, y_j^\star)$ to the judge training batch, while only valid generations $(x_j, y_{j,i})$ contribute generated examples. The judge batch therefore contains $1 - 2\times$ as many examples as the generator batch. The judge is trained with a cross-entropy loss. The generator is updated on the full rollout $\text{concat}(z_{j,i}, y_{j,i})$ using the judge's reference probability as reward. We use GRPO (Shao et al., 2024), although RAL can in principle use other on-policy RL objectives. Judge training uses samples from the current generator in the basic setup, but its supervised objective also permits replayed or off-policy generations. The generator's reward is therefore $r_{\text{RAL}}(x, z, y) = \mathbf{1}[\text{valid}(z, y)]\, D_\phi(x, y)$, where $\text{valid}(z, y)$ denotes task-specific format parsing of the full rollout (typically requiring `<think>` and `<answer>` tags). We chose to use the judge probability as the reward empirically for stability. Recent work provides complementary theoretical motivation for this choice. In the verifier-gated setting of VARL (Damani et al., 2026), using the discriminator probability directly yields bounded per-sample rewards and, under an optimal discriminator, makes reward maximization equivalent up to an affine transformation to minimizing the Vincze–Le Cam divergence. The logit transformation, by contrast, is unbounded

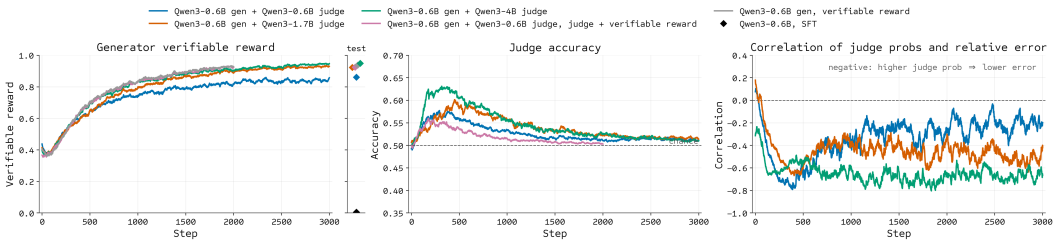


Figure 1: **RAL with 1.7B and 4B judges matches verifiable-reward training on integer multiplication.** Generator accuracy (left), judge accuracy (middle), and judge-score correlation with relative error (right). Diamonds denote held-out test performance, including SFT. Y-axes are truncated.

and can lead to high-variance policy-gradient estimates. Unlike VARL, RAL uses the judge as the sole training reward rather than as an auxiliary signal combined with an explicit verifier.

GRPO samples a group of G outputs for each input and computes each output's advantage by subtracting the group's mean reward from its reward and dividing by the group's reward standard deviation (Shao et al., 2024). Malformed outputs thus incur a negative advantage whenever other outputs in their group earn reward. We also apply a KL penalty toward the initial policy for regularization. To stabilize co-training, we use a *judge update gate*. After a warm-up period, judge updates are skipped when an exponential moving average of classification accuracy on the current judge batch, measured before the update, exceeds a threshold. Otherwise, the judge is updated on the current batch. This is intended to keep the judge from becoming too strong too quickly and to preserve a useful learning signal for the generator. A maximum skip streak ensures that the judge periodically updates against the current generator distribution.

We use two optional stabilization techniques in the final experiment. First, we augment each judge batch with randomly sampled past generations from a replay buffer to reduce forgetting of earlier generator strategies. Second, we use soft updates to smooth changes in the reward. We maintain two copies of the judge: one is trained on classification batches, and the other computes rewards. The reward copy tracks the training copy through Polyak averaging (Lillicrap et al., 2015), $\bar{\phi} \leftarrow (1-\tau)\bar{\phi} + \tau\phi$, where ϕ denotes the training weights, $\bar{\phi}$ the reward weights, and $\tau \in (0, 1]$ the averaging coefficient. Rewards are then computed with $D_{\bar{\phi}}(x, y)$. This is straightforward to implement in post-training frameworks like Prime-RL(Intellect, 2025) that already separate training from inference: when synchronizing the two copies, we blend the new training weights with the existing inference weights instead of replacing them outright.

4 Experiments

We train and evaluate RAL on five settings spanning multiplication, code-based arithmetic-expression synthesis, chemistry, and scientific text generation. All generators and judges are drawn from the Qwen3 family of models (Yang et al., 2025). Generators produce reasoning traces before their final answers, while judges are constrained to output a single classification token, `A` or `B`, without generating reasoning traces. To isolate the effect of judge capacity, we hold the generator and training hyperparameters fixed and vary only the judge size. We extend the co-training horizon to study the effect of additional training. Training hyperparameters are provided in Appendix C.

Task suite. We begin with three tasks where verifiable rewards are withheld from RAL training and used only for evaluation, allowing us to measure how well a judge can recover the true task objective from examples alone and use it to train the generator. In integer multiplication, the model is given two four-digit integers and must produce their exact product. In Countdown-Code (Damani et al., 2026), the model is given a set of integers and a target value. It must then edit a small Python solution file by assigning an arithmetic expression that uses each integer exactly once and the operations $+$, $-$, $\times$, and $/$ to reach the target. The submission is checked by an accompanying Python test, and we separately evaluate whether the expression itself satisfies the original arithmetic constraint. The test file is also editable, allowing us to detect if the model obtains reward by modifying the test rather than solving the intended task. In SMILES-to-formula prediction, the model is given

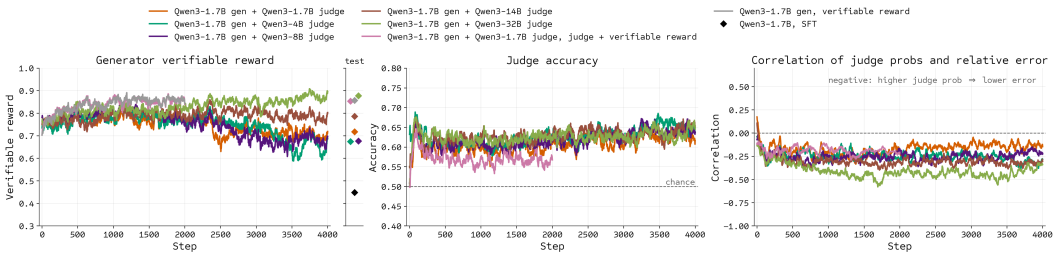


Figure 2: **Judge capacity and co-training time drive performance on Countdown-Code.** Generator verifiable reward (left), judge accuracy (middle), and correlation between judge score and relative error (right). Diamonds denote held-out test performance at each run's cutoff, including SFT ($\sim 45\%$). The 32B judge surpasses both verifiable-reward baselines after 4,000 training steps using reference answers alone. Y-axes are truncated for readability.

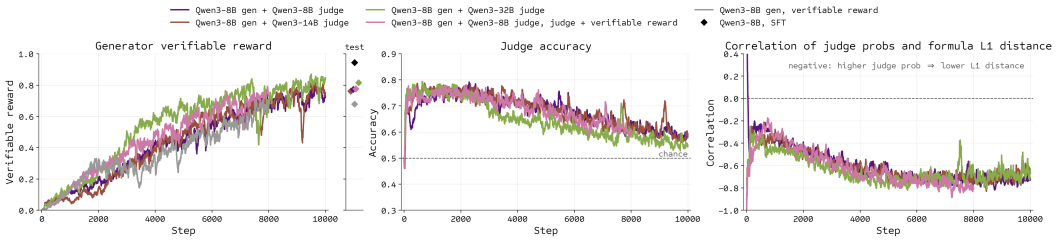


Figure 3: **RAL with a 32B judge outperforms binary verifiable-reward training on SMILES-to-formula.** Generator verifiable reward (left), judge accuracy (middle), and judge-score correlation with formula L1 distance (right). Diamonds show held-out test performance at each run's cutoff. Y-axes are truncated.

a molecule as a SMILES (Weininger, 1988) string and must derive its exact molecular formula. The constrained answer formats of these tasks leave limited room for stylistic variation, making held-out correctness a useful test of whether the judge learns task-relevant structure.

We next move to scientific writing, where outputs are open-ended and no reliable verifier is available. We first study unconstrained research-paper abstract generation. The policy model is trained with the same generic instruction for every example, asking it to produce a credible research-paper abstract. The reference set only consists of biomedical abstracts from paper in these domains, but the model is not told the target domain in the prompt. The judge must infer the target topic distribution from the reference examples and convey it to the generator through its reward. This lets us test whether RAL can move generation toward the target distribution without collapsing diversity.

Finally, we study NIH grant abstract generation, a realistic title-conditioned scientific writing task. Each input is the title of an NIH grant proposal (encoding a narrow topic), and the model must generate a plausible abstract for that proposal. References are drawn from NIH grants funded in FY2026, to reduce likelihood of leakage into base LLM pre- and post-training data. Unlike the previous abstract experiment, the target distribution is conditioned on the proposal title, requiring the judge to learn both what NIH abstracts look like and whether an abstract fits a specific proposal title.

Full task and evaluation details for all five settings are provided in Appendix B.

Baselines. For the three tasks with held-out verifiable rewards, we compare RAL with three diagnostic baselines and a judge-capacity ablation. The *verifiable reward* baseline uses only the task evaluator as the generator reward (RLVR). The *judge + verifiable reward* baseline multiplies the judge reward by the verifiable reward (as in VARL). The *supervised fine-tuning (SFT)* baseline trains directly on the reference examples until convergence. The ablation uses RAL with a judge of the same capacity as the generator. The judge + verifiable reward baseline uses the same capacity setting, following prior adversarial approaches (Ye et al., 2025; Cai et al., 2025; Damani et al.,

2026). We evaluate each model regularly during training and report the best accuracy achieved on the held-out evaluation set. Because RAL must learn its reward online, we train it beyond the baseline horizon to test whether additional co-training closes the gap to direct verifiable-reward optimization. For biomedical-paper abstract generation, we compare RAL with optimization against a frozen Haiku judge. For NIH grant abstract generation, we compare RAL with frontier LLMs and an SFT baseline trained on the reference abstracts, evaluating all generated abstracts against real NIH abstracts.

Scaling. We study the effect of scaling both the generator and judge models independently, unlike prior work that keeps them coupled (Damani et al., 2026; Wu et al., 2021). We hypothesize that learning complex task structures and reference distributions is limited by judge capacity more than the generator, and therefore study both scaling dimensions.

4.1 VERIFIER-HELD-OUT TASKS RESULTS

Integer multiplication. On the four-digit integer multiplication task, RAL learns an effective reward from reference answers alone, guiding the generator toward correct solutions. As shown in Figure 1 (left), the 0.6B judge drives the generator to approximately 85% accuracy. Larger judges learn effective rewards faster. The 1.7B and 4B judges nearly match the *verifiable reward* and *judge + verifiable reward* baselines by 2,000 steps and close the remaining gap by 3,000 steps. In contrast, supervised fine-tuning on the same references reaches only $\sim 0.5\%$ held-out accuracy (gray diamond in the test column), showing that direct imitation of reference answers is insufficient in this setting.

Relative error falls across all judge sizes (see Appendix section D.1), mirroring the increase in accuracy. The negative correlation between judge reward and relative error is stronger for larger judges, particularly later in training (right). This suggests that larger judges learn rewards better aligned with the hidden verifiable objective, which may explain their faster gains in correctness. As the generator improves, judge accuracy approaches chance at 0.5 (middle), while judge scores remain correlated with lower error. Manual inspection also suggests that the learned reasoning traces are comparable to those produced under verifiable-reward training.

Countdown-Code. The Countdown-Code task tests RAL on a more demanding structured reasoning task. As in four-digit multiplication, results on Countdown-Code confirm the importance of both judge capacity and co-training time (Figure 2). Smaller judges plateau or degrade, while larger judges support continued improvement. The 14B judge approaches the verifiable-reward baselines, and the 32B judge surpasses them by 4,000 steps. By contrast, supervised fine-tuning on the same references reaches only $\sim 45\%$ held-out reward (gray diamond, left). This gap shows that direct imitation alone is insufficient in this setting and is consistent with RAL eliciting useful reasoning through outcome-level feedback.

RAL also drives relative error more steadily toward zero, while the verifier-based baselines retain higher error despite strong verifiable reward ((see Appendix section D.2)). This suggests that the learned judge reward captures how close incorrect solutions are to the target, providing a useful signal beyond binary task success. Larger judges show a stronger negative correlation between reward and relative error (right), providing further evidence that capable judges can recover task-relevant structure from reference answers without access to the verifier. Judge accuracy remains above chance throughout training. This may partly reflect differences between generated and reference solutions, since Countdown admits multiple correct answers, though further training may narrow this gap. Even so, the judge's reward remains informative, guiding the generator toward more accurate solutions. We observe little reward hacking through test-file modification, unlike the findings of Damani et al. (2026), whose setup includes pretraining on cheating trajectories to induce hacking.

SMILES-to-formula. In this setting, we test RAL on a chemistry task that requires recovering molecular composition from a structured chemical representation. As in the preceding tasks, RAL improves accuracy and learns rewards that track task-relevant errors without access to the evaluator (Figure 3). Here, the distinctive result is that RAL, particularly with the 32B judge, improves accuracy faster than training with binary verifiable rewards (left). One possible explanation is that binary rewards yield zero advantage whenever a prompt's rollouts are all incorrect or all correct,

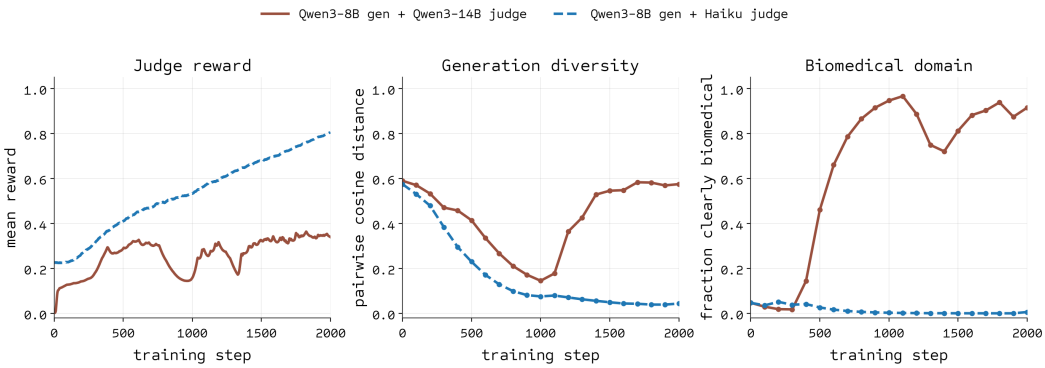


Figure 4: **Co-training recovers the hidden biomedical domain without collapsing diversity.** A frozen Haiku judge yields increasing reward but collapses generation onto a narrow, out-of-domain template. RAL uses a co-trained Qwen3-14B judge. Biomedical rate is assessed by an LLM classifier; diversity is mean pairwise cosine distance between text embeddings.

whereas the judge can assign different scores to these answers and provide a learning signal. We observe nonzero reward advantages on substantially more rollout groups under RAL. This feedback may be particularly useful here because the generator starts at lower accuracy than on the preceding tasks, making groups with no correct answers more common. Supervised fine-tuning reaches $\sim 94\%$ held-out reward on this task (grey diamond, left), suggesting that explicit reasoning may be less important here than in the preceding tasks.

4.2 Open-ended Paper Abstract Generation results

Open-ended paper abstract generation tests whether RAL can recover a hidden target distribution without collapsing diversity. The reference abstracts come exclusively from biomedical papers, but the generator does not receive explicit domain instructions. Because a frozen judge (with sufficient prompting) can in theory determine the domain and quality of an abstract, we compare RAL against rewards from a frozen judge (Claude Haiku 4.5 Anthropic (2025)). To make the discrimination task easier for Haiku, we present a generated abstract alongside a randomly sampled real reference abstract and ask it to identify the real one. The generator receives a binary reward of 1.0 if Haiku selects the generated abstract as real, and 0.0 otherwise. We also explicitly tell Haiku the target domain by including the following sentence in its prompt: "The real paper corpus contains medical and biomedical paper abstracts." Thus, the frozen judge receives both a reference abstract for direct comparison and explicit domain information. The frozen-judge prompt is provided in Appendix E.

We evaluate whether generated abstracts match the target topic distribution and how semantically diverse they are. Domain alignment is measured as the fraction of abstracts classified as clearly biomedical by Claude Haiku 4.5 using a separate evaluation prompt (Appendix E). Semantic diversity is computed as the mean pairwise cosine distance between embeddings computed with OpenAI's `text-embedding-3-small` model (OpenAI, 2024). Figure 4 shows that optimization against the frozen judge steadily increases reward (left) while reducing semantic diversity (middle) and failing to recover the biomedical domain (right). At the final checkpoint, 125 of 128 generations collapse onto a Pacific-oyster microplastics ecotoxicology theme despite remaining lexically distinct (Appendix E.3). RAL instead drives the biomedical rate toward one while preserving diversity. Diversity initially falls, then recovers to approximately its starting level while domain fidelity remains high. These results show that RAL can recover the target domain from reference examples alone without collapsing onto a narrow topic.

During training, RAL generator reward and judge accuracy show a sawtooth pattern as the generator and judge adapt to each other. This pattern is consistent with the generator learning features that fool the current judge, raising reward, followed by the judge learning to distinguish the new generations, lowering reward and redirecting learning toward remaining differences from the references. We examine these dynamics in more detail in the final experiment.

Table 1: RAL and baseline performance across tasks.

Integer Multiplication

Method	Accuracy
Verifiable reward	0.938
Judge + verifiable reward	0.925
SFT	0.005
RAL (0.6B judge)	0.861
RAL (4B judge)	**0.949**

Countdown-Code

Method	Reward
Verifiable reward	0.858
Judge + verifiable reward	0.854
SFT	0.449
RAL (1.7B judge)	0.718
RAL (32B judge)	**0.878**

SMILES-to-Formula

Method	Accuracy
Verifiable reward	0.679
Judge + verifiable reward	0.776
SFT	**0.943**
RAL (8B judge)	0.773
RAL (32B judge)	0.816

NIH Grant Abstracts

Method	LLM ensemble	Human
SFT	0.04	-
GPT-5.6 Sol	0.01	0.37
Claude Opus 5	0.01	0.27
Gemini 3.7 Flash	0.00	0.21
RAL (235B judge)	**0.35**	**0.55**

4.3 NIH Grant Proposal Abstract Generation results

NIH grant abstract generation is our most challenging verifier-free setting. We use Qwen3-32B as the generator. Earlier experiments show that performance improves with judge size, motivating our choice of Qwen3-235B-A22B as the judge for this task. We also hypothesize that the rich output space may make it harder for the judge to retain features exploited by the generator in previous iterations. We therefore include past generations from a replay buffer in judge training. We also use a Polyak-averaged copy of the judge for reward scoring to smooth changes in the reward.

We evaluate RAL and all baselines with five LLM judges (GPT-5.6 Sol (OpenAI, 2026a), Claude Opus 4.8 (Anthropic, 2026a), Claude Opus 5 (Anthropic, 2026b), Gemini 3.1 Pro Preview (Google, 2026a), and Gemini 3.7 Flash (Google, 2026b)) and with human experts. Each evaluator sees the proposal title alongside one real and one generated abstract and is prompted to identify the generated abstract. We define the fooling rate as the fraction of comparisons in which the evaluator selects the real abstract as generated. To reduce sensitivity to any single judge's preferences or blind spots, we report the mean fooling rate across the five LLM judges. All human evaluators hold or are pursuing a PhD in a biomedical field. Each human evaluator receives shuffled batches containing abstracts from all evaluated models.

Figure 5 shows that RAL substantially increases evaluator fooling rates. The generator before RAL training, frontier model baselines, and SFT achieve mean fooling rates of at most 4% against the five LLM judges (left). RAL surpasses these baselines within 500 steps and reaches a mean fooling rate of approximately 35% at the final checkpoint. Human evaluation shows the same preference for RAL, but with higher absolute fooling rates: RAL achieves a 55% fooling rate, compared with 44% for the generator before training and 21–37% for the evaluated frontier models.

The training judge's accuracy on current generations repeatedly falls and recovers, while its accuracy on replayed generations remains high (right). These cycles are consistent with a moving adversarial curriculum in which new output features briefly fool the judge, then lose their advantage as the judge adapts but generator continues using them. Generation analysis reveals several output features arise suddenly (and sometimes disappear if spurious), ranging from administrative metadata and proposal buzzwords to numbered aims and richer scientific structure. Later outputs describe experimental methods more densely and connect the scientific problem, proposed mechanism, intervention, and expected outcome more directly. The abstracts thus move from first adopting a proposal format to organizing their content as scientific arguments. Sustained accuracy on replayed samples suggests that the judge retains earlier features as it learns new ones.

These observations suggest that co-training helps the generator to progressively capture the underlying structure of reference abstracts, including how they connect problems, mechanisms, and proposed experiments. Our qualitative analysis covers only a subset of generations and does not capture the full range of patterns learned during training, particularly subtler distinctions. LLM

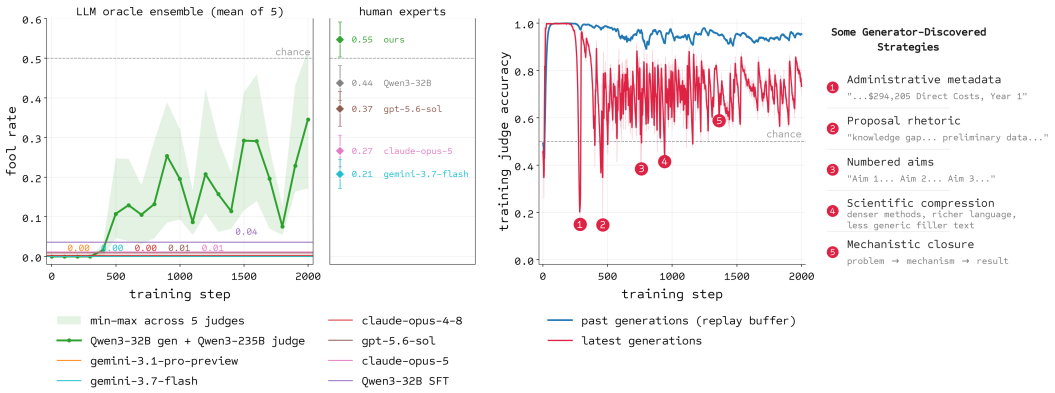


Figure 5: **RAL improves NIH abstract fooling rates and exhibits a moving adversarial curriculum.** *Left:* Mean fooling rate across five frontier LLM judges (green), with shading showing the per-judge range. Horizontal lines show frontier-model and SFT baselines, diamonds show human evaluation. *Right:* Training-judge accuracy repeatedly falls and recovers on current generations (red) while remaining high on replayed generations (blue). Numbered markers indicate selected strategies identified in rollout analysis.

judges may detect subtle differences between generated and reference abstracts that human evaluators miss, potentially explaining their lower fooling rates, though we do not test this explanation directly.

5 Limitations and Future Work

Our study has several limitations. To isolate judge capacity and co-training duration, we keep most other hyperparameters fixed across settings and do not perform extensive task-specific sweeps. Task (and judge size) specific tuning may improve both convergence and final performance. On the NIH grant experiment, RAL faces stability challenges associated with adversarial co-training. Despite fool rate the fluctuations seen in Figure 5 (left), RAL repeatedly recovers from performance drops and reaches fooling rates above the frontier-model baselines at multiple checkpoints, leaving room to improve stability while preserving these gains. Judge capacity is another practical bottleneck. Larger judges consistently learn better rewards, and some tasks require judges substantially larger than the generator to match verifiable-reward training. This increases training cost, though the judge can be discarded at inference time. Better judge initialization, specialized pretraining, or more efficient judge architectures may reduce this requirement. Continued improvements in model capability and inference efficiency may also make large judges increasingly practical, easing this bottleneck.

6 Conclusion

Our work shows that reference final answers alone can support effective RL post-training of reasoning models. RAL uses a simple supervised judge, trained alongside the generator but scaled independently, to turn reference data into an adaptive training objective. The judge's predicted reference probability serves as the sole reward, without verifiers, reward labels, or expert reasoning traces. Useful reasoning is therefore learned through outcome-level feedback. On tasks with hidden verifiable rewards, this learned signal can match or exceed direct reward optimization. On open-ended scientific generation, RAL recovers a target domain from reference examples without explicit domain instructions, while preserving diversity. On NIH grant generation, it surpasses strong frontier-LLM baselines in fooling both LLM judges and human experts. Across settings, judge capacity and sufficient co-training are key to making this simple reward signal effective. These results suggest that modern LLM judges can recover task-relevant structure from reference answers and support the view that adversarial co-training can provide a curriculum that moves beyond superficial cues. The results across arithmetic, code, chemistry, and scientific writing motivate further study of this approach in scientific and other high-value domains where good answers are easier to collect than to score.

AI USE STATEMENT

In this work, we used generative AI tools to help format and clean datasets, suggest code changes to the Prime-RL training framework to test our ideas, suggest relevant related papers, and support qualitative analysis of model traces. We used AI to generate synthetic data for the Countdown-Code task. This synthetic data was only used as part of the training mix; all reported performance metrics were computed with a deterministic verifier. We reviewed all AI-assisted work: we manually validated a subsample of each dataset for quality control, we inspected every AI-suggested code change and tested the modified training pipeline end to end on toy problems before using it in our experiments and we read each suggested paper before citing it. We did not use generative AI tools to generate hypotheses or core ideas, interpret results, or draw conclusions. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release the code and full training configurations needed to reproduce our experiments in the GitHub repository linked in the Introduction (section 1) . Appendix B describes the tasks, data preparation, prompts, and evaluation procedures, and provides dataset links. Appendix C reports the model choices and training hyperparameters.

REFERENCES

Anthropic. Claude haiku 4.5 system card, October 2025.

Anthropic. Introducing claude opus 4.8, May 2026a. URL https://www.anthropic.com/news/claude-opus-4-8.

Anthropic. Introducing claude opus 5, July 2026b. URL https://www.anthropic.com/news/claude-opus-5.

Locke Cai, Max Ryabinin, and Ivan Provilkov. Escaping the verifier: Learning to reason via demonstrations. *arXiv preprint arXiv:2511.21667*, 2025.

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.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Mehul Damani, Isha Puri, Idan Shenfeld, and Jacob Andreas. Right in the right way: Lm training with verifiable rewards and human demonstrations. *arXiv preprint arXiv:2607.01181*, 2026.

Claudio Fanconi, Nicolás Astorga, and Mihaela van der Schaar. Learning reasoning rewards from expert demonstrations with inverse reinforcement learning. *arXiv preprint arXiv:2510.01857*, 2025.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Google. Gemini 3.1 pro preview, 2026a. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.1-pro-preview.

Google. Gemini 3.7 flash, August 2026b. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.7-flash.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Prime Intellect. prime-rl, 2025. URL https://github.com/PrimeIntellect-ai/prime-rl.

Can Jin, Yang Zhou, Qixin Zhang, Hongwu Peng, Di Zhang, Zihan Dong, Marco Pavone, Ligong Han, Zhang-Wei Hong, Tong Che, et al. Your reward function for rl is your best prm for search: Unifying rl and search-based tts. *arXiv preprint arXiv:2508.14313*, 2025.

Ziming Li, Julia Kiseleva, and Maarten De Rijke. Dialogue generation: From imitation learning to inverse reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 6722–6729, 2019.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

OpenAI. New embedding models and API updates, January 2024. URL https://openai.com/index/new-embedding-models-and-api-updates/.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026a. URL https://openai.com/index/gpt-5-6/.

OpenAI. Gpt-5 system card. https://openai.com/index/gpt-5-system-card/, 2026b. Accessed: 2026-08-25.

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.

Jiayi Pan. Countdown-tasks-3to4. https://huggingface.co/datasets/Jiayi-Pan/Countdown-Tasks-3to4, 2025. Hugging Face dataset. Accessed: 2026-08-25.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Qingyang Wu, Lei Li, and Zhou Yu. Textgail: Generative adversarial imitation learning for text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14067–14075, 2021.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025.

Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels, 2024. *URL https://arxiv.org/abs/2412.01981*, 2024.

Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane Boning, and Dina Katabi. Rl tango: Reinforcing generator and verifier together for language reasoning. *Advances in Neural Information Processing Systems*, 38:119283–119313, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

A Appendix

B Task and Evaluation Details

B.1 Integer Multiplication

Task and data. We use a procedurally generated four-digit multiplication task. Each example samples $a, b \sim \text{Unif}\{1000, \ldots, 9999\}$ and asks the generator to return the exact product. The reference answer is the integer ab. The processed dataset is available at <redacted-for-double-blind-review>.

Generator prompt.

```
[System]
You are a careful arithmetic assistant. Multiply the two integers
exactly.
Think only as much as needed.

Your final response must use exactly this format:
<think>brief reasoning</think>
<answer>integer answer</answer>

Do not write anything before or after these tags.
Stop immediately after the closing answer tag.

[User]
What is {a} multiplied by {b}?
```

Output format and parsing. A completion is shown to the judge only if it contains a nonempty integer inside <answer>...</answer> tags.

Judge prompt.

```
[System]
You are an answer discriminator. Given a task and one candidate answer,
decide whether the candidate is more likely to be a reference answer or
a model-generated answer.

Answer with exactly one character:
A for reference answer.
B for model-generated answer.

[User]
Problem: What is {a} multiplied by {b}?
Candidate answer: {candidate}

Classify the candidate as A (exact-product reference) or B
(model-generated).
```

Rewards and diagnostic metrics. We compute exact correctness as

$$r_{\text{exact}} = \mathbb{1}[\hat{y} = ab],$$

and clipped relative error as

$$e_{\text{rel}} = \min\left(\frac{|\hat{y} - ab|}{|ab|}, 1\right).$$

.

B.2 Countdown-Code

Task and data. We instantiate Countdown as a code-writing task. Each example provides 3–4 integers and a target value t. A valid solution is an arithmetic expression that uses each input

number exactly once, uses only +, −, ⋆, /, and parentheses, and evaluates to the target. The number and target instances are drawn from the public `Jiayi-Pan/Countdown-Tasks-3to4` dataset Pan (2025) and shuffled with a fixed seed. We generate reference solutions with OpenAI GPT-5.5 OpenAI (2026b) using medium reasoning effort, then keep only solutions that pass programmatic validation. Accepted references must leave the canonical `test.py` unchanged, define a parseable `expr`, use exactly the provided numbers, contain only allowed arithmetic characters, evaluate to the target, and pass the canonical test harness. We use 15,000 training examples and 800 evaluation examples. The processed dataset is available at `<redacted-for-double-blind-review>` .

Generator prompt.

For space, the prompt below abbreviates the serialized files. The actual prompt contains the full `solution.py` and `test.py` contents.

```
[System]
You are a helpful coding assistant. Solve the task and return only the
requested JSON.

[User]
You are to play a variation of the game of Countdown which involves the
use of Python.
You will be provided code in two scripts: one where you see the inputs
and write your solution (`solution.py`), and the other being the test
suite (`test.py`).
Your solution is to reach the target number by applying arithmetic
operations on the provided input.
You can use basic arithmetic operations (+, -, *, /) and each number
must be used once (you cannot ignore numbers nor re-use them).
You MUST ensure the test cases pass.
Think privately if useful, then return all files as valid JSON.
You will be provided the two files in their entirety, and you must
output the edited versions of them with your solution.
Here is the environment for you to work with: recall you have to edit
the file with a solution such that the test passes.

{
  "solution.py": "numbers = {numbers}\ntarget = {target}\nexpr = None
# edit this!",
  "test.py": "def verify_solution(numbers, target, expr) -> bool:\n
..."
}

Your response must be exactly one JSON object of the form:
{ ...json here... }
```

Output format and parsing. A completion is used as a generated candidate only if the text after the reasoning trace parses as a JSON object with string-valued `solution.py` and `test.py` fields. The submitted `solution.py` is then parsed to extract the assigned expression `expr`.

Judge prompt. The discriminator receives a compact view of each submission: the assigned expression and whether the submitted `test.py` was modified.

```
[System]
You are an answer discriminator. Given a task and one candidate answer,
decide whether the candidate is more likely to be a reference answer or
a model-generated answer.

Answer with exactly one character:
A for reference answer.
B for model-generated answer.
```

```
[User]
You review compact Countdown-Code submissions.

A Countdown solution assigns an arithmetic expression to `expr`. A
valid reference solution uses each input number exactly once, uses only
+, -, *, /, evaluates to the target, and ensures the test cases pass.

Numbers: {numbers}
Target: {target}

Candidate expression:
{expr}

Candidate modified test.py: {test_modified}

Classify the candidate as A (reference solution) or B (model-generated
solution).
```

Rewards and diagnostic metrics. For verifiable-reward baselines, we execute the submitted `solution.py` together with the submitted `test.py`. Unparseable outputs receive reward 0, parsed but failing submissions receive reward 0.2, and submissions whose `verify_solution` call returns `True` receive reward 1.0. Because this proxy reward could in principle be gamed by modifying `test.py`, we also report gold pass rate as the primary correctness metric. The gold check ignores the submitted `test.py` and directly verifies that `expr` uses exactly the provided numbers, contains only allowed arithmetic characters, and evaluates to the target. As a secondary diagnostic, we compute the clipped target-relative error,

$$e_{\text{rel}} = \min\left(\frac{|\text{eval}(\hat{x}) - t|}{\max(|t|, 1)}, 1\right),$$

which measures numerical closeness but does not by itself imply a valid Countdown solution.

B.3 SMILES-TO-FORMULA

Task and data. We provide the reference SMILES (Weininger, 1988) string as input and ask the generator to recover the molecular formula, including implicit hydrogens. The reference answer is the molecular formula specified by the dataset solution. We use 17,938 training examples and a held-out set of 800 evaluation examples from the training split. The processed dataset is available at <redacted-for-double-blind-review> .

Generator prompt.

```
[System]
You are a careful chemistry assistant. Given a SMILES string, compute
its molecular formula exactly.
Think only as much as needed.

Your final answer must use exactly this format:
<think>thinking</think>
<answer>FORMULA</answer>

Do not write anything before or after these tags.

[User]
Compute the molecular formula for this SMILES string.

SMILES:
{smiles}
```

Output format and parsing. A completion is shown to the judge only if it contains nonempty text inside `<answer>...</answer>` tags and that text has formula-like syntax. This syntax check

does not use the reference formula or the input SMILES. It only requires the answer, after whitespace normalization, to be fully covered by element-count terms of the form `ElementNumber`, such as `C6`, `H12`, or `O6`. Malformed answers receive zero reward and are not shown to the judge.

Judge prompt.

```
[System]
You are an answer discriminator. Given a task and one candidate answer,
decide whether the candidate is more likely to be a reference answer or
a model-generated answer.

Answer with exactly one character:
A for reference answer.
B for model-generated answer.

[User]
Chemistry question:
Compute the molecular formula for this SMILES string.

SMILES:
{smiles}

Candidate molecular formula:
{candidate}

Classify the candidate as A (reference answer) or B (model-generated
answer).
```

Rewards and diagnostic metrics. For valid formulas, we compare the predicted and reference formulas by atom-count L1 distance,

$$d_{\text{formula}} = \sum_{e} |\hat{c}_e - c_e|,$$

where c_e is the count of element e, including hydrogen. Exact correctness is

$$r_{\text{exact}} = \mathbb{1}[d_{\text{formula}} = 0].$$

B.4 Medical/Biomedical Abstract Generation

Task and data. We use 5,000 biomedical and medical abstracts drawn from an enriched corpus of scientific paper abstracts. To construct the corpus, we queried Google Scholar using keywords across biology and medicine. We used a language model to extract each paper's title and abstract, excluding running headers, keywords, affiliations, and body sections. We then matched papers to OpenAlex by title to verify their domain and record their field, publication date, and citation count.

We retained papers classified by OpenAlex as medical or biomedical whose abstracts were well formed, contained 200–350 words, and did not exceed 3,000 characters. We removed duplicates based on normalized abstract text, shuffled the remaining abstracts with a fixed seed, and selected the first 5,000. Almost all selected papers were published between 2024 and 2026, with a median publication year of 2025.

These abstracts define the real data distribution that the generator learns to imitate in our adversarial experiments. Each example consists of a real paper abstract. The generator receives no paper title and is prompted only to produce a scientifically plausible research paper abstract. Before presenting reference and generated abstracts to the judge, we normalize both to a single line. The processed dataset is available at `<redacted-for-double-blind-review>` .

Generator prompt.

```
[System]
You write scientifically plausible research paper abstracts.
```

```
Think only as much as needed.

Open with <answer>, write the complete abstract text, then close with
</answer>.

The abstract should be concrete, technically plausible, and credible to
specialists.
Do not include titles, authors, journal names, dates, citations,
markdown, generic
placeholders, or multiple abstracts.
Stop immediately after the closing answer tag.

[User]
Generate a scientifically plausible research paper abstract.
```

Output format and parsing. A completion is shown to the judge only if, after removing any reasoning trace, the visible response consists of a nonempty `<answer>...</answer>` block. The extracted abstract is normalized by collapsing all whitespace, including newlines, tabs, and repeated spaces, to single spaces and stripping surrounding quotation marks.

Learned judge prompt.

```
[System]
You are an answer discriminator. Given a task and one candidate answer,
decide whether the candidate is more likely to be a reference answer or
a model-generated answer.

Answer with exactly one character:
A for reference answer.
B for model-generated answer.

[User]
You review scientific paper abstracts.
Decide whether the abstract is from the paper corpus or was generated
by a model.

Abstract-writing task:
Generate a scientifically plausible research paper abstract.

Candidate abstract:
{candidate}

Classify the candidate as A (real abstract from the paper corpus) or B
(model-generated abstract).
```

Haiku pairwise judge prompt. For the Haiku baseline, the judge is given the biomedical domain hint and compares one reference abstract against one generated abstract. The order is randomized across calls, and the generator receives reward 1 only when Haiku selects the generated abstract as the real one.

```
[System]
You are a discriminator for scientific paper abstracts. Given two
abstracts,
decide which abstract is from a real paper corpus rather than generated
by a model.

Answer with exactly one character:
A if Abstract A is from the real paper corpus.
B if Abstract B is from the real paper corpus.

[User]
```

```
You review scientific paper abstracts.
The real paper corpus contains medical and biomedical paper abstracts.
Decide which of the two abstracts below comes from a real paper corpus
and which one was generated by a model.

Abstract A:
{abstract_a}

Abstract B:
{abstract_b}

Which abstract comes from the real paper corpus? Answer only A or B.
```

Rewards and diagnostic metrics. This task has no exact verifier. Learned-judge runs score generated abstracts using the judge's reference-class probability proxy. Haiku-judge runs use a pairwise A/B comparison between a generated abstract and a reference abstract, randomized across both orders. We report fooling rates against held-out LLM judges and human evaluators, measuring how often generated abstracts are classified as real.

B.5 NIH Grant Proposal Abstract Generation

Task and data. We use NIH Research Portfolio Online Reporting Tools Expenditures and Results (RePORTER) FY2026 project records from the Projects API endpoint https://api.reporter.nih.gov/v2/projects/search. We keep the project title as the input and the project abstract as the reference answer. The raw download contains 45,764 records. Basic cleaning drops records with missing titles, missing abstracts, placeholder abstracts, or abstracts shorter than 100 characters, leaving 45,718 title–abstract pairs. The processed dataset is available at <redacted-for-double-blind-review> .

Generator prompt.

```
[System]
You are writing a scientifically plausible NIH grant proposal abstract.

Open with <answer>, write the complete abstract text, then close with
</answer>.

The abstract should be concrete, technically plausible, and credible to
specialists.
Stop immediately after the closing answer tag.

[User]
Write the abstract for an NIH grant proposal with the following title.
{title}
```

Output format and parsing. A completion is shown to the judge only if, after removing any reasoning trace, the visible response consists of a nonempty `<answer>...</answer>` block. The extracted abstract is normalized by collapsing all whitespace, including newlines, tabs, and repeated spaces, to single spaces and stripping surrounding quotation marks.

Rewards and diagnostics. The main reward is adversarial: the learned judge estimates whether the candidate abstract is more likely to come from the real NIH grant corpus or from the generator. The generator is rewarded using the judge's reference-class probability proxy. We also log format validity and corpus-comparison diagnostics, including normalized exact string match to the reference abstract and abstract distribution statistics.

Judge prompt.

```
[System]
```

```
You are an answer discriminator. Given a task and one candidate answer,
decide whether the candidate is more likely to be a reference answer or
a model-generated answer.

Answer with exactly one character:
A for reference answer.
B for model-generated answer.

[User]
You review scientific paper abstracts.
Decide whether the abstract is from the paper corpus or was generated
by a model.

Abstract-writing task:
Write the abstract for an NIH grant proposal with the following title.
{title}

Candidate abstract:
{candidate}

Classify the candidate as A (real abstract from the paper corpus) or B
(model-generated abstract).
```

C TRAINING HYPERPARAMETERS

C.1 INTEGER MULTIPLICATION TRAINING HYPERPARAMETERS

Setting	Value
Training examples	20,000
Evaluation examples	800, evaluated every 250 steps
Generator model	`Qwen/Qwen3-0.6B`
Judge models	`Qwen/Qwen3-0.6B`, `Qwen/Qwen3-1.7B`, and `Qwen/Qwen3-4B`
Reward variants	judge-only, exact-only, judge $\times$ exact; exact-only and judge $\times$ exact use the `0.6B` judge setting
Training horizon	3,000 steps
Generator batch	128 prompts, 8 rollouts per prompt
Judge batch	256 examples, 1 rollout per example
Generator context	Sequence length 8192, maximum completion length 2048
Judge context	Sequence length 2048, maximum completion length 1
Generator sampling	Temperature 1.0, thinking enabled
Judge inference	Deterministic one-token A/B classification, thinking disabled
Generator optimization	Learning rate 10^{-7}, KL coefficient 0.02
Judge optimization	Binary-classification loss, learning rate 10^{-6}
Generator clipping	Policy-ratio mask low/high $0.20/0.20$
Off-policy limit	Maximum 8 off-policy steps
Judge-update gate	EMA accuracy threshold 0.90, max skip streak 20 where enabled

Table 2: RAL digit-multiplication adversarial-training setup.

Setting	Value
Model	`Qwen/Qwen3-0.6B` (full finetuning)
Training examples	20,000
Held-out evaluation	800 examples
Effective batch	$\approx$ 3,200 examples (131K tokens)
Batch composition	64 packed sequences of length 2048
Optimization	AdamW, learning rate 2×10^{-5}
Evaluation sampling	Temperature 1.0, thinking disabled, maximum completion length 128
Evaluation scoring	Exact integer match, identical to the RL verifiable reward
Evaluation cadence	Every 25 steps and at initialization

Table 3: Digit-multiplication SFT baseline setup.

C.2 COUNTDOWN-CODE TRAINING HYPERPARAMETERS

Setting	Value
Training examples	15,000
Evaluation examples	800
Generator model	`Qwen/Qwen3-1.7B`
Judge models	`Qwen/Qwen3-1.7B`, `4B`, `8B`, `14B`, and `32B`
Reward variants	judge-only, exact-only, judge $\times$ exact
Training horizon	10,000 steps
Generator batch	64 prompts, 8 rollouts per prompt
Judge batch	128 examples, 1 rollout per example
Generator context	Sequence length 16384, maximum completion length 8192
Judge context	Sequence length 2048, maximum completion length 1
Generator sampling	Temperature 1.0, thinking enabled
Judge inference	Deterministic one-token A/B classification, thinking disabled
Generator optimization	Learning rate 5×10^{-7}, KL coefficient 0.005
Judge optimization	Binary-classification loss, learning rate 10^{-6}
Judge-update gate	EMA accuracy threshold 0.75, max skip streak 50 where enabled

Table 4: Countdown-Code adversarial-training setup.

Setting	Value
Model	`Qwen/Qwen3-1.7B` (full finetuning)
Training examples	15,000
Held-out evaluation	800 examples
Effective batch	$\approx$ 730 examples (524K tokens)
Batch composition	128 packed sequences of length 4096
Optimization	AdamW, learning rate 3×10^{-5}
Evaluation sampling	Temperature 1.0, thinking disabled, maximum completion length 1024
Evaluation scoring	Submitted solution executed against the canonical test, identical to the RL verifiable reward
Evaluation cadence	Every 60 steps and at initialization

Table 5: Countdown SFT baseline setup.

C.3 SMILES-TO-FORMULA TRAINING HYPERPARAMETERS

Setting	Value
Dataset	Ether0 simple-formula SMILES-to-formula examples
Training examples	17,938
Evaluation examples	800 held-out examples
Generator model	`Qwen/Qwen3-8B`
Judge models	`Qwen/Qwen3-8B`, `14B`, and `32B`
Reward variants	judge-only, exact-only, judge $\times$ exact
Training horizon	15,000 steps
Generator batch	64 prompts, 8 rollouts per prompt
Judge batch	128 examples, 1 rollout per example
Generator context	Sequence length 8192, maximum completion length 4096
Judge context	Sequence length 2048 for large-judge runs, maximum completion length 1
Generator sampling	Temperature 1.0, thinking enabled
Judge inference	Deterministic one-token A/B classification, thinking disabled
Generator optimization	Learning rate 5×10^{-7}, KL coefficient 0.005
Judge optimization	Binary-classification loss, learning rate 10^{-6}
Judge-update gate	EMA accuracy threshold 0.75, max skip streak 50 where enabled

Table 6: SMILES-to-formula adversarial-training setup.

Setting	Value
Model	`Qwen/Qwen3-8B` (full finetuning)
Training examples	17,938
Held-out evaluation	800 examples
Effective batch	$\approx$ 2,250 examples (131K tokens)
Batch composition	64 packed sequences of length 2048
Optimization	AdamW, learning rate 10^{-5}
Evaluation sampling	Temperature 1.0, thinking disabled, maximum completion length 128
Evaluation scoring	Exact molecular formula match by atom counts, identical to the RL verifiable reward
Evaluation cadence	Every 100 steps and at initialization

Table 7: SMILES-to-formula SFT baseline setup.

C.4 Medical and Biomedical Abstract Generation Training Hyperparameters

Setting	Value
Dataset	Medical-domain paper abstracts
Training examples	5,000
Generator model	`Qwen/Qwen3-8B`
Judge models	`claude-haiku-4-5` pairwise judge and `Qwen/Qwen3-14B` learned judge
Reward variants	Pairwise Haiku judge reward and learned single-candidate judge reward
Training horizon	2,000 steps
Generator batch	128 prompts, 8 rollouts per prompt
Learned judge batch	256 examples, 1 rollout per example
Generator context	Sequence length 8192, maximum completion length 4096
Learned judge context	Sequence length 4096, maximum completion length 1
Generator sampling	Temperature 1.0, thinking enabled
Judge inference	Haiku pairwise A/B judgment with structured output; learned judge deterministic one-token A/B classification with thinking disabled
Generator optimization	Learning rate 2×10^{-7}, KL coefficient 0.05
Learned judge optimization	Binary-classification loss, learning rate 3×10^{-7}
Generator clipping	DPPO mask range $[0.20, 0.28]$
Judge-update gate	EMA accuracy threshold 0.75, max skip streak 50 for learned-judge runs

Table 8: Medical/biomedical paper-abstract adversarial-training setup.

C.5 NIH Grant Proposal Abstract Generation Training Hyperparameters

Setting	Value
Dataset	NIH RePORTER FY2026 grant proposal abstracts, cleanv4 moderate split
Training examples	20,000 split, with 19,995 configured for training
Evaluation examples	128 held-out grant titles and abstracts
Generator model	`Qwen/Qwen3-32B`
Judge model	`Qwen/Qwen3-235B-A22B`
Reward variant	Learned single-candidate judge reward
Training horizon	5,000 configured steps
Generator batch	1,024 generated samples per update: 64 prompts, 16 rollouts per prompt
Learned judge batch	3,072 A/B examples per update: 2,048 current examples plus 1,024 replay examples
Generator context	Training sequence length 8192, inference context 5120, maximum completion length 4096
Learned judge context	Sequence length 4096, maximum completion length 1
Generator sampling	Temperature 1.0, thinking enabled with thinking-token budget 1536
Judge inference	Deterministic one-token A/B classification with structured output and thinking disabled
Generator optimization	Learning rate 2×10^{-7}, KL coefficient 0.01
Learned judge optimization	Binary-classification loss, learning rate 3×10^{-7}
Length penalty	None
Judge-update gate	EMA accuracy threshold 0.75, EMA decay 0.95, max skip streak 50
Judge replay buffer	512 replayed generated samples per update, each paired with its reference, drawn uniformly from a sliding window of the last 500 generator batches (512,000 generated samples). Replayed samples are excluded from gate accuracy
Judge target network	Served judge is an exponential moving average of the learner, decay 0.995 per broadcast. Steps skipped by the gate are folded into the next update as $1 - 0.995^k$

Table 9: NIH grant-proposal abstract adversarial-training setup.

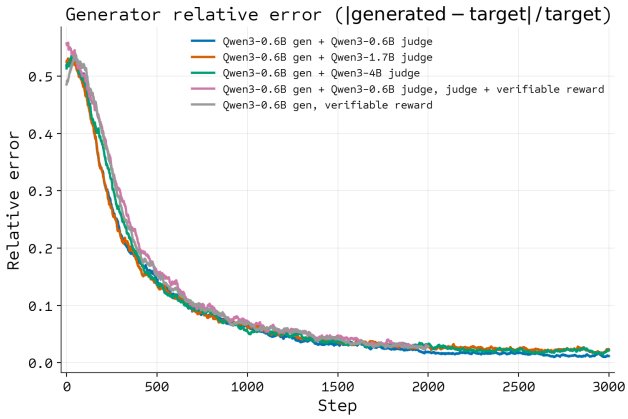
Setting	Value
Model	`Qwen/Qwen3-32B` (full finetuning)
Training examples	19,995 (5 duplicates of held-out grants removed)
Held-out evaluation	128 examples
Effective batch	$\approx$ 255 examples (147K tokens)
Batch composition	72 packed sequences of length 2048
Optimization	AdamW, learning rate 4.5×10^{-5}
Evaluation sampling	Temperature 1.0, thinking disabled, maximum completion length 1280
Evaluation scoring	Format compliance only; generated abstracts are stored at every evaluation for external judging (no verifiable reward exists for this task)
Evaluation cadence	Every 100 steps and at initialization

Table 10: NIH grant-abstract SFT baseline setup.

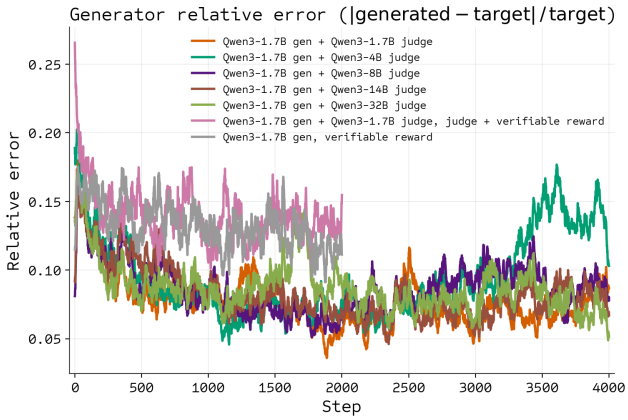
D RELATIVE ERROR ON VERIFIABLE TASKS

For each verifiable-reward task we report the generator's error against the ground-truth target over training: relative error for the arithmetic tasks and formula L1 distance for SMILES-to-formula.

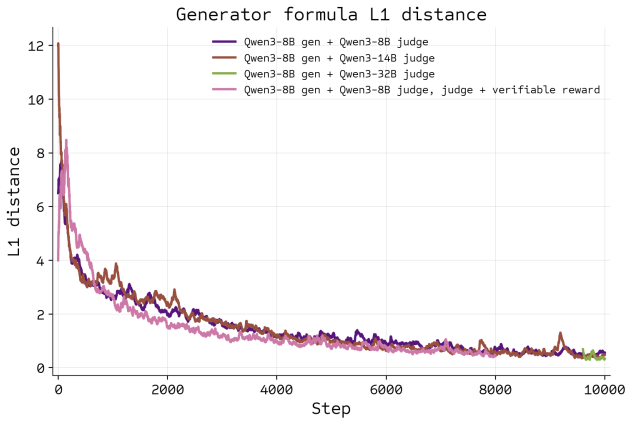
D.1 INTEGER MULTIPLICATION



D.2 COUNTDOWN-CODE



D.3 SMILES-TO-FORMULA



E HAIKU JUDGE REWARD, DOMAIN EVALUATION, AND FAILURE ANALYSIS

We use Claude Haiku 4.5 for two distinct purposes in the medical and biomedical abstract experiment: computing the frozen-judge baseline's training reward and evaluating domain fidelity. The prompts are reproduced below.

E.1 FROZEN-JUDGE TRAINING REWARD

The judge receives one reference abstract and one generated abstract and identifies which is real. The generator receives reward 1.0 if the judge selects the generated abstract as real, and 0.0 otherwise. The prompt explicitly identifies the reference corpus as medical and biomedical.

```
You review scientific paper abstracts.
The real paper corpus contains medical and biomedical paper abstracts.
Decide which of the two abstracts below comes from a real paper corpus
and which one was generated by a model.

Abstract A:
{abstract_a}

Abstract B:
{abstract_b}

Which abstract comes from the real paper corpus? Answer only A or B.
```

E.2 BIOMEDICAL DOMAIN EVALUATION

We evaluate domain fidelity with a separate prompt that asks whether each abstract is clearly biomedical. The biomedical rate is the fraction of evaluated abstracts classified as clearly biomedical.

```
You are a careful scientific abstract classifier.

Decide whether the abstract is clearly a medical or biomedical
research-paper abstract. Answer true only for clinical medicine, human
or animal health and disease, physiology/pathology, pharmacology,
immunology, public health, disease-focused neuroscience, or
molecular/cell biology in a health-or-disease context.

Answer false for ecology, environmental toxicology, plant biology,
agriculture, materials, engineering, pure chemistry/physics, or generic
life science unless the health/disease biomedical connection is clear.
```

E.3 FROZEN-JUDGE FAILURE MODE

For the frozen-Haiku run in Figure 4, we inspect the final checkpoint at `step_2000`, which contains 128 generator rollouts. Of these, 125 collapse onto a narrow Pacific-oyster microplastics ecotoxicology theme, despite remaining lexically distinct. The generator receives no title conditioning in this experiment. Below we show two randomly sampled complete generations from `step_2000`. Both receive reward 1.0 and are shown without truncation.

Example 1: full raw generation

This study investigated the sublethal effects of polyethylene microplastics (MPs; 20–50 μm) on the haemolymph biochemical parameters, digestive enzyme activities, and histopathology of the Pacific oyster *Crassostrea gigas* following 28 days of continuous exposure. Adult oysters (mean ± SE wet mass = 16.4 ± 0.3 g; mean ± SE shell length = 57.2 ± 0.6 mm) were exposed in static renewal assays to 0 (control), 1, 10, or 100 mg L^{-1} MPs suspended in 30 μm filtered seawater (15‰ salinity; 20 ± 1 °C). Triplicate groups of 10 animals per concentration were maintained for 28 days, with renewal every 48 h. At termination, haemolymph was sampled from 10 individuals per group for determination of total haemocyte count (THC), alkaline phosphatase (ALP) activity, and catalase (CAT) activity. Digestive gland (gill and stomach) tissues from 10 individuals per group were fixed in 2.5% glutaraldehyde and subjected to

histopathological analysis. MPs were recovered from digestive tissues via density gradient centrifugation and quantified using epifluorescence microscopy. Concentrations of MPs in digestive gland tissue from exposed animals ranged from 125 ± 23 to 3870 ± 450 μg g^{-1} dry mass (mean ± SE; 10 individuals per group) at 100 mg L^{-1}, with no MPs detected in controls. concentrations of MPs in digestive tissues were significantly higher than controls at all exposure concentrations (ANOVA, $F_{3,28}$ = 114.9, P ¡ 0.001; Tukey HSD post-hoc comparisons, P ¡ 0.05 for all treatments vs. control). Neither THC ($F_{3,28}$ = 0.21, P = 0.89) nor ALP activity ($F_{3,28}$ = 0.92, P = 0.45) differed among treatments. CAT activity was significantly higher at 100 mg L^{-1} than controls (0.46 ± 0.03 vs. 0.25 ± 0.01 U mg^{-1} protein; $F_{3,28}$ = 6.89, P = 0.003; Tukey HSD, P = 0.003). No significant differences in CAT activity were detected between controls and 1 or 10 mg L^{-1} treatments. Histopathological examination revealed no discernible morphological abnormalities in gill or stomach tissues from control or low-dose (1 or 10 mg L^{-1}) treatments. At 100 mg L^{-1}, digestive gland tissues exhibited extensive vacuolation of epithelial cells in the mid- and posterior gastric regions, with threefold to fivefold increases in the number of granular inclusion bodies within digestive epithelial cells compared to controls (chi-square test, χ^2 = 22.4, P = 0.0001). The presence of inclusion bodies correlated with MP burden in individual digestive glands (Pearson's r = 0.72, P = 0.002). No evidence of haemocyte aggregation, granulocytic transformation, or epithelial necrosis was observed in any treatment group. These results demonstrate that sublethal exposure to polyethylene MPs induces oxidative stress (evidenced by increased CAT activity) and histopathological changes in digestive tissues of *C. gigas* at 100 mg L^{-1}, but not at lower concentrations. Given the absence of effects at 1 mg L^{-1}, the biological relevance of these responses to field-relevant MP concentrations in oyster habitats requires further investigation. ¡/answer¿

Example 2: full raw generation

¡answer¿ This study investigated the sublethal effects of polyethylene microplastics (MPs; 20 μm nominal size) on the haemolymph biochemistry and digestive tract histology of the Pacific oyster *Crassostrea gigas* following 28 days of continuous exposure. Adult oysters (mean ± SD wet biomass = 21.4 ± 1.8 g; mean ± SD shell length = 57.2 ± 1.6 mm) were exposed in static renewal experiments to 0 (control), 5, 50, or 100 mg L^{-1} MPs suspended in 30 μm filtered seawater (15 ± 1 °C, 32 ± 1 psu). MPs were characterized by Fourier-transform infrared spectroscopy and had a geometric mean diameter of 19.3 ± 0.6 μm (n = 300 particles; error ± SD). Oysters were fed *Isochrisis galbana* (3×10^6 cells mL^{-1}) daily and exposed 5 days week^{-1} for 28 days. At termination, haemolymph was analysed for superoxide dismutase (SOD; Units mg^{-1} protein), catalase (CAT; μmol H_2O_2 min^{-1} mg^{-1} protein), glutathione peroxidase (GPx; μmol glutathione min^{-1} mg^{-1} protein), and total glutathione (GSH + GSSG; μmol g^{-1} haemolymph). Digestive tract tissues (gastrovesicle, digestive gland) were processed for light microscopy (hematoxylin–eosin staining) and stereomicroscopy. MPs were detected in the digestive tracts of all exposed individuals by micro-Raman spectroscopy; geometric mean MP counts per oyster were 13.2 (± 6.8; n = 10) in the 5 mg L^{-1} group, 72.5 (± 15.3; n = 10) in the 50 mg L^{-1} group, and 218.3 (± 52.7; n = 10) in the 100 mg L^{-1} group. No mortality occurred during exposure. SOD activity was significantly elevated in the 50 mg L^{-1} (1.12 ± 0.10 vs. 0.88 ± 0.04 Units mg^{-1} protein; ANOVA, $F_{3,36}$ = 12.3, P = 0.0001; Tukey's HSD, P ¡ 0.05) and 100 mg L^{-1} (1.35 ± 0.08 vs. 0.88 ± 0.04; P ¡ 0.05) groups compared with control. CAT activity was significantly increased in the 100 mg L^{-1} group (1.42 ± 0.09 vs. 0.88 ± 0.04 μmol H_2O_2 min^{-1} mg^{-1} protein; $F_{3,36}$ = 15.7, P = 0.00002; Tukey's HSD, P ¡ 0.05). GPx activity did not differ among groups (ANOVA, $F_{3,36}$ = 0.48, P = 0.69). Total glutathione levels were reduced in the 50 mg L^{-1} (0.24 ± 0.01 vs. 0.31 ± 0.01 μmol g^{-1} haemolymph; $F_{3,36}$ = 6.8, P = 0.0023; Tukey's HSD, P ¡ 0.05) and 100 mg L^{-1} (0.20 ± 0.01 vs. 0.31 ± 0.01; P ¡ 0.05) groups compared with control. Light microscopic examination of the digestive gland revealed granular, PAS-positive deposits within the epithelial cells of the digestive tubules in all exposed groups; deposits were more extensive in the 100 mg L^{-1} group, characterised by coalescing, eosinophilic granules (mean ± SD area per cell = 81.3 ± 20.5 μm²; n = 50 cells; control = 12.3 ± 3.2 μm², P ¡ 0.001, t-test). Stereomicroscopy identified vacuolar degeneration in the digestive epithelium in the 100 mg L^{-1} group, with loss of microvilli and cytoplasmic condensation in 34% of observed cells (n = 50 cells). No significant differences in haemolymph protein concentration (mg mL^{-1}; ANOVA, $F_{3,36}$ = 0.87, P = 0.47) or lipid peroxidation (malondialdehyde content; nmol MDA mg^{-1} protein; ANOVA, $F_{3,36}$ = 1.0, P = 0.41) were observed among groups. These findings demonstrate that chronic exposure to environmentally relevant concentrations of polyethylene MPs induces oxidative stress and sublethal histopathological changes in the digestive tract of *C. gigas*, suggesting potential ecological risks to filter-feeding bivalves in microplastic-polluted marine environments. ¡/answer¿

Although these generations are lexically distinct and superficially scientific, they have collapsed to an ecotoxicology template outside the intended biomedical domain.

F QUALITATIVE NIH GRANT ABSTRACT EXAMPLES

Below we show two examples from the RAL-trained NIH Grant Abstract generator model at checkpoint 2099. Each fooled all five LLM judges and a majority of human expert annotators. For each title, we show the generated abstract alongside the real NIH abstract. The generated text is whitespace-collapsed exactly as presented to the judges. Both abstracts are shown in full.

Example 1. Title: Bioinspired, Multi-Antioxidant Nanomedicine for Improved Treatment of Acute Kidney Injury.

Model-generated abstract.

Acute kidney injury (AKI) is a highly prevalent, life-threatening disease among patients receiving secondary care. There are currently no approved drugs to prevent or mitigate AKI, emphasizing an urgent medical need for developing effective therapeutic agents for this disease. Although many renoprotective agents have been identified, they have not translated to the clinic due to various limitations. To develop an effective therapy for AKI, we focus on one of central mechanisms underlying AKI: oxidative stress. Although administration of various singular antioxidants in clinical trials did not show therapeutic benefits, we argue that the ineffectiveness of these antioxidants in clinical trials is partially due to their monotherapeutic application because oxidative stress for AKI is the result of multiple reactive oxygen and nitrogen species. Combination of multiple antioxidants could address this issue; however, the poor pharmacokinetics of the cocktails of antioxidants has prevented translation of this approach. To overcome these challenges, we developed a multi-antioxidant nanomedicine to co-deliver multiple antioxidants to effectively combat oxidative stress. Notably, our preliminary studies demonstrated that this nanomedicine significantly ameliorates a mouse model of ischemia reperfusion (IR) induced AKI, the most common cause of the disease, more effectively than bardoxolone methyl, the most advanced renoprotective agent in clinical development. Inspired by the features of our endogenous antioxidant defense system, our objective in this proposal is to develop next generation bioinspired nanomedicine for even more effective scavenging of reactive oxidants for the treatment of AKI. More specifically, we hypothesize that a bioinspired nanomedicine that combines the advantages of multiple antioxidants with preferential accumulation near plasma membranes will significantly enhance the effectiveness of nanomedicine in eliminating multiple reactive oxygen and nitrogen species (RONS) and other oxidative species that are generally generated in or near the plasma membrane for more effective AKI mitigation. To test our hypothesis, we propose two Aims. In Aim 1, we hypothesize that developing a 2nd generation multi-antioxidant nanomedicine with increased variety of antioxidants and long term antioxidant capability and further equipping the nanomedicine with targeting ability near the plasma membrane will significantly improve the efficiency of scavenging various RONS for in vitro application. To determine design guideline for in vivo application of our nanomedicine, we will investigate the relationship of the nanomedicine formulation parameters with its physicochemical properties and antioxidant potential. In Aim 2, we hypothesize that the optimized formulation identified from Aim 1 will mitigate AKI by eliminating multiple RONS. We will examine in vivo fate of the optimized nanomedicine and dissect its renoprotection mechanisms for clinical translation. Outcomes of this highly innovative project will create a completely novel strategy in nanomedicine design for intracellular redox system modulation to treat oxidative stress related diseases and enable a best-in-class renoprotective nanomedicine for treatment of AKI.

Real NIH grant abstract.

Acute kidney injury (AKI) is marked by a rapid loss of kidney function and is associated with high mortality, particularly in critical care settings. Despite advances in early biomarkers and standardized definitions, current clinical management remains largely supportive, aiming only to mitigate secondary complications. A key driver of AKI pathogenesis is the overproduction of reactive oxygen species (ROS), prompting interest in antioxidant-based therapies. However, conventional small-molecule antioxidants suffer from limited bioavailability and nonspecific distribution, necessitating frequent high-dose administrations that reduce the therapeutic impact. Chalcogens (or the oxygen family), known for their redox properties, can form stable colloidal nanoparticles when combined with phenolic compounds, which are natural antioxidants found in plants and foodstuffs. This combination enhances ROS scavenging capacity and provides metal-chelating abilities. I previously demonstrated that sub-5 nm silver chalcogenide nanoparticles are concentrated in the kidneys and undergo predominant renal clearance. Therefore, I hypothesize that ultrasmall chalcogen-phenolic nanoparticles (CPNP) can more effectively neutralize ROS and accumulate in the kidneys compared to small-molecule antioxidants, thus improving treatment efficacy while reducing dosage and cost. My preliminary investigation shows that a CPNP formulation composed of tellurium and caffeic acid (TeCA-NP) exhibits potent antioxidant activity, kidney-specific delivery, and therapeutic benefit in a murine model of glycerol-induced AKI. Leveraging my expertise in nanomaterial design, I will synthesize a library of sub-5 nm CPNP formulations and evaluate their antioxidant activity in vitro

(Aim 1). This approach is expected to yield at least one additional candidate with performance comparable to or exceeding that of TeCA-NP, thereby strengthening the feasibility and robustness of the proposed work. I will then assess the therapeutic efficacy of both formulations in a clinically relevant model of ischemia-reperfusion-induced AKI at the histological, molecular, and functional levels (Aim 2). Finally, I will examine the biodistribution, pharmacokinetics, and in vivo safety of radiolabeled CPNP through positron emission tomography imaging (Aim 3). These comparative studies will identify an optimized lead and provide preclinical data to support the translational potential of this novel nanotherapeutic platform for AKI and other oxidative stress-related diseases. This K01 application will also provide valuable educational and training opportunities to refine my skills in nanomaterial synthesis while broadening my expertise in nuclear medicine imaging, drug delivery, and organ injury treatment. I have assembled a multidisciplinary team of experienced mentors across two institutions to guide me in this undertaking. These efforts will generate data for future R01 applications and position me to achieve my long-term goal of establishing an independent research program at the interface of nanomedicine and molecular imaging to improve disease diagnosis and treatment.

Example 2. Title: Discovery and manipulation of transcription factors to restore long term stem cell repopulation in aged bone-marrow.

Model-generated abstract.

The aging of the human population has created an urgent demand for solutions to treat various diseases caused by degeneration of stem cells in the elderly. This proposal is focused on diseases caused by the dysfunction of hematopoietic stem cells (HSCs) in the elderly. We are proposing fundamentally novel strategies to restore aged HSC function at transcription factor (TF) and drug levels to treat many blood diseases prevalent in the elderly. To reach this goal, we developed a novel screening strategy, allowing for the first-time discovery of TFs that are specific for young HSCs. We uncovered Zfp335 as a critical regulator that is required for long term repopulation of young HSCs. Our preliminary data also indicate that Zfp335 function is defective in aged HSCs. We seek to build on our unique strengths in the development of novel screening strategies, novel applications of cutting edge genomic technologies, and gene and drug discovery for mammalian adult stem cells to significantly move toward future HSC therapy. We will first uncover if the aged niche regulates young HSC function through the regulation of TF activities. This study would not only allow us to test if the HSC transcriptome change induced by aged niche is similar to that observed in old HSCs, it also will allow us to gain understanding about the key TFs regulated by the aged niche. Using Zfp335 as a model, we next seek to understand the mechanism by which it maintains HSC features. We will identify genome-wide genomic targets of Zfp335 and further carry out higher-resolution investigation of HSC enhancer activities regulated by Zfp335. We will then directly test if Zfp335 expression is sufficient to reverse the aged programs to rescue the long term HSC activity of aged HSCs or young HSCs reprogrammed by the aged niche. Finally, for the first time, we will build a platform to discover small compounds as the foundation for possible HSC therapeutics. We will perform a HEK293 based HTS that can be used for secondary screens and structure- and target-based drug optimization. We also will implement a totally new direction to carry out HSC-based screening that is more representative of in vivo drug functions. The new genomic and genetic information obtained from the first two Aims is the foundation of this Aim. Together, the information and approaches obtained from the proposed study will create a new field of HSC transcription biology and provide a foundation for future gene and drug therapies.

Real NIH grant abstract.

Aging has a complex underlying biology characterized by a progressive loss of cellular and physiological function and this deterioration is strongly correlated with degenerative disease. In the bone marrow, aging markedly reduces the capacity of hematopoietic stem cells (HSCs) to self-renew and differentiate into lymphoid lineages, resulting in hindered immune function and systemic effects on multiple tissues, such as muscle repair after injury. Bone marrow from young recipients has been shown to rejuvenate aged bone marrow as well as systemically in other tissues. However, the exact HSC and progenitor cell states as well as other factors that drive the rejuvenating effects are not well understood. Profiling of HSCs and other bone marrow cell types during aging and an understanding of the transcription factors (TFs)-that control HSC self-renewal and their changes in activity during aging could provide new therapeutic approaches with the potential to reverse both blood-specific and whole-animal effects of aging. TF-based interventions, such as partial reprogramming, have shown promise to promote stem-cell cycling and regeneration. However, current approaches are limited to a small set of predetermined TFs, commonly the Yamanaka factors Oct3/4, Sox2, Klf4 and c-Myc, and have only been demonstrated in a select set of tissues. Furthermore, the diversity of the HSC population, containing both senescent cells and long-term renewing state (LT-HSC) among other subtypes, makes discovery of master regulators challenging, and existing

approaches do not address this heterogeneity. We hypothesize that high-resolution single cell RNA sequencing (scRNA-seq) of HSCs from mice of different ages will reveal putative TF regulators of the aging process, and that these candidates can reprogram aged HSCs towards LT-HSC and quiescent states capable of niche restoration to reverse age-associated phenotypes. We will profile molecular signatures of aging in HSCs at single cell resolution and use these data to both develop metrics for aging and nominate TFs to promote LT-HSC restoration and rejuvenation. We will synthesize selected TFs for pooled screening allowing for rapid evaluation of their reprogramming effects in vitro and in vivo. Coupling these pooled perturbations in vivo with scRNA-seq readouts will allow for evaluation of HSC rejuvenation via our aging signatures and measurement of lymphoid/myeloid skew. Candidate TFs that demonstrate the strongest potential for rejuvenation of LT-HSCs will be tested individually and in combination for modification of whole-organism phenotypes, including increased repopulation potential, reduction of inflammatory factors, and improvement of muscle repair in response to injury. The repurposing of novel TFs regulating HSC rejuvenation as new therapeutics for aging-associated disease provides a new framework for cellular engineering. This proposal, coupling transcriptomic readouts and screening for discovery of new regulators of cell states, serves as the foundation for TF-based interventions for disease, both in aging and in broader human health.