Online Reinforcement Learning for Autoformalization

Simon Sorg
Department of Computer Science
University of Cambridge
tss52@cam.ac.uk

Wenda Li
Department of Computer Science
University of Edinburgh
wenda.li@ed.ac.uk

Soumya Banerjee *
Department of Computer Science
University of Cambridge
sb2333@cam.ac.uk

Abstract

We study online reinforcement learning for *autoformalization*: translating informal mathematical conjectures into formal statements in `Lean 4`. Autoformalization lacks a natural dense reward; using type-check success alone produces syntactically valid but semantically vacuous outputs and is vulnerable to reward hacking. We propose composite reward formulations that combine syntactic signals (Lean type-check) with semantic measures: (i) exact-match style metrics based on BEq / BEq+ when ground-truth formalizations exist, and (ii) a continuous embedding-based similarity reward that requires no paired data. Training uses Group Relative Policy Optimization with a KL-divergence penalty to avoid distributional collapse, and we release a multi-project Lean-4 REPL API and training code. On RLMEval (including a manually curated, context-aware Con-NF set of 71 samples), our best RL model (batch size 768, "TripleEmbed" scaling) substantially raises type-check rates (TC1 from 17.29% $\rightarrow$ 26.01%, TC50 from 76.41% $\rightarrow$ 85.46%) while producing modest gains in BEq / BEq+. We analyse reward tradeoffs, show the importance of the KL penalty, and discuss extensions to proof autoformalization and learned critics as future work.

1 Introduction

Online Reinforcement Learning has proven successful for mathematical reasoning in natural languages [Shao et al., DeepSeek-AI et al.] and more recently for Automated Theorem Proving (ATP) [Wang et al., 2025, Ren et al., 2025] with Large Language Models (LLMs).

Unlike ATP, autoformalization, translating informal into formal language, has no obvious underlying reward function. Using type-check accuracy as the only reward might lead to models producing syntactically valid yet meaningless output. Instead, semantic alignment can be explored with metrics such as BEq [Liu et al.] and BEq+ [Poiroux et al., 2025].

We develop reward functions that use both syntactic and semantic accuracy, and test online reinforcement learning for autoformalization of natural language conjectures. We publicly release our training code, including a new Lean REPL [LeanProver-Community, 2025] API that can handle multiple different Lean 4 [Moura and Ullrich, 2021] projects simultaneously.

*Corresponding author

Multimodal Algorithmic Reasoning Workshop at NeurIPS 2025 (MAR-NeurIPS 2025).

2 Method

We use Group Relative Policy Optimization, which was introduced by Shao et al. and has seen success in informal languages [DeepSeek-AI et al.] and formal languages [Ren et al., 2025]. We generate 48 formal outputs per informal conjecture, and compute advantage estimates $\hat{A}_i$ within each such group i.

This advantage estimate is then used for Proximal Policy Optimization with gradient clipping [Schulman et al., Zhang et al.], resulting in the objective function

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left\{r(\theta,t)\hat{A}_i, \text{clip}\left(r(\theta,t), 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right\} \\ -\beta D_{KL}[\pi_\theta, \pi_{\theta_{\text{old}}}]$$

where π is our policy, o_i one of the G completions to the input prompt q, D_{KL} the Kullback-Leibler divergence for o_i, and r the probability ratio given by

$$r(\theta,t) := \frac{\pi_\theta(o_{i,t}|q \cdot o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q \cdot o_{i,<t})}$$

with $\cdot$ as string concatenation and θ_{old} as the weights of the supervised base model.

A naive reward for an LLM completion would be type-check information such as assigning a binary reward of 0 or 1 depending on whether the type-check succeeded, resulting in a model that generates valid Lean syntax for any given informal statement. But this can be achieved trivially by always predicting the same short sequence with reward 1, for example .

To mitigate this, one can incorporate a Kullback-Leibler-divergence penalty [Schulman et al., Shao et al.]. However, a large coefficient for the divergence penalty limits the ability to improve performance, as the model's existing weaknesses can only partially be resolved without shifting the distribution [Schulman et al., Kaufmann et al.]. Instead of solely penalizing distributions, we adapt the reward to also convey semantic meaning by adding a second reward term. We consider two formulations.

1. *BEq*: With existing ground-truth data, metrics such as BEq+ [Poiroux et al., 2025] can be used to compare prediction and ground-truth. A new binary result can be introduced, assigning 1 whenever two statements are equal in BEq+, and 0 otherwise. But if ground-truth data is available, supervised finetuning is also possible, and it is unclear whether reinforcement learning outperforms training with language modeling loss.
2. *Embed*: Without ground-truth data, a proxy for semantic correctness could be the similarity between formal and informal conjecture embeddings assigned by an embedding model. This can provide a continuous reward function that gradually increases from 0 to 1, without sudden loss spikes whenever a reward of 1 is reached, which is common with sparse rewards [Kaufmann et al.]. We find divergence penalties to still be relevant to prevent reward hacking for the embedding model used, which is the retriever by Liu et al..

2.1 Technical details

All experiments were seeded with seed 42.

We rely on vLLM for generations [Kwon et al., 2023], and use DeepSpeed Zero stage 3 for training, using a single node with 4 A100 80 GiB GPUs and bfloat16. We use five warmup steps and a constant learning rate of 10^{-6}. We use a modified version of the trl [von Werra et al., 2025] library for training, and a weight decay of 0.1. Training runs for 3 epochs, which is 198 GRPO-steps.

The supervised finetuning for the Kimina model [Wang et al., 2025] uses a batch size of 8, a learning rate of 0.000007, and runs for 4 epochs, trained on a single node with bfloat16 and DeepSpeed Zero stage 3.

All training runs use AdamW [Loshchilov and Hutter] for optimisation, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 0.00000001$.

2.2 Batch size

Larger batch sizes lead to increased performance. We ultimately settled on 768 for computational reasons. For a comparison of batch sizes 384 and 768, see Figure 1.

2.3 KL-divergence penalty

Omitting the KL-divergence penalty lead to drastic performance decreases. Specifically, for all runs, we observed a collapse at around 100 training steps. Figure 2 displays the decrease at step 100, and also shows how the models, once reward has fallen to 0, do not recover.

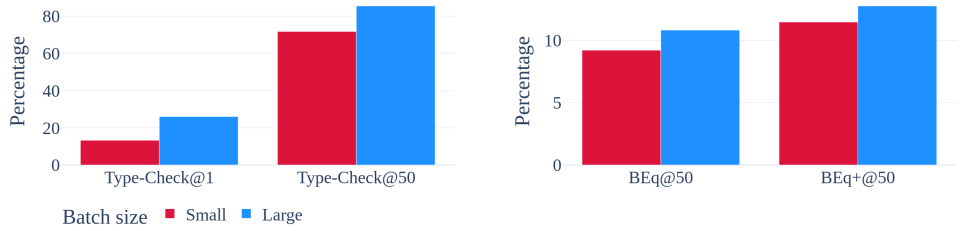

Figure 1: The larger batch size of 768 increases over 384 in every metric on RLMEval [Poiroux et al., 2025]

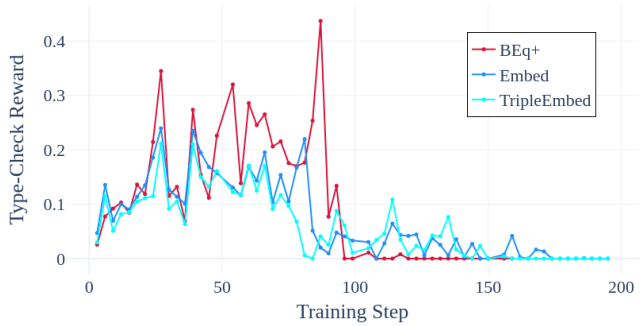

Figure 2: Type-Check reward over time for runs without KL-divergence penalty. For all runs, the reward reaches 0, and the runs never recover from it.

3 Results

As a baseline, we evaluate the Kimina autoformalization model [Wang et al., 2025] on RLMEval [Poiroux et al., 2025]. The model can generate syntactically correct statements, but rarely generates statements that are equal to the ground-truth according to BEq [Liu et al.], see Table 1. Especially, since RLMEval is based on external repositories with custom definitions, formalization requires handling these in the model context.

For a context-aware base-model, we manually curate a dataset of 71 Con-NF samples with definition context as training for context-aware autoformalization. The context-aware model supervised fine-tuned on this dataset significantly improves RLMEval results semantically. For online reinforcement learning, we split RLMEval, using the FLT repository as validation, such that the model is tested on unseen definitions.

We find the KL-divergence penalty to be important to prevent distributional collapse, shown in Figure 3. As the type-check accuracy has low mean and high variance compared to the embedding reward, we also test scaling it by a factor of 3, called *TripleEmbed*, which leads to similar means for both reward components.

While the reward formulation with BEq on ground-truth data significantly outperforms in syntactic and semantic metrics, it is unclear whether it improves over supervised finetuning on the same data.

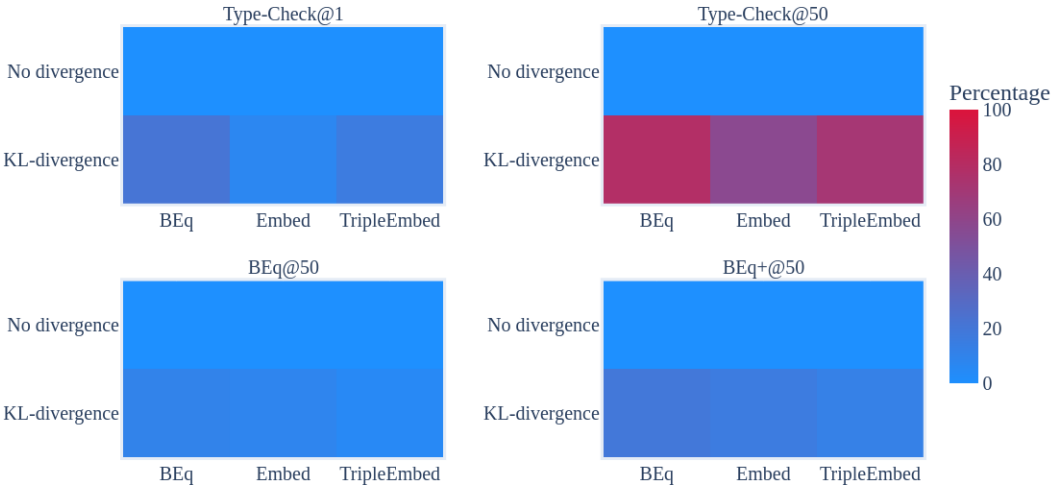

Figure 3: Heatmap of type-check accuracy in pass@1 and pass@50, as well as pass@50 BEq [Liu et al.] and BEq+ [Poiroux et al., 2025] scores for the six configurations on the FLT repository of RLMEval [Poiroux et al., 2025].

The reward function using our embedding model does not rely on ground-truth data. Consequently, we can even report results on all of RLMEval. This is not training on test data, as our method uses ground-truth statements only during evaluation.

Throughout training, we iteratively refine our outputs purely with type-check information and embedding similarity scores, which are also available during inference. Therefore, this approach would even be valid at inference time.

Our overall best model uses a batch size of 768 and the *TripleEmbed* reward. The final results in Table 2 indicate a strong improvement in type-check accuracy, and a slight increase in BEq and BEq+ scores across RLMEval.

4 Future works

We aim to extend this work on conjecture autoformalization to online reinforcement learning on proof autoformalization. Two possible methods for this arise. Proof autoformalization can be decomposed into individual steps, such as sketching and ATP in Draft, Sketch, and Prove [Jiang et al., 2023], and a reward can be assigned to each step separately. This might be type-check accuracy of the proof sketch, and a binary reward for successful proofs for ATP. Alternatively, proof autoformalization could be treated as an end-to-end problem, only rewarding 1 once the whole proof has been correctly formalized.

Further, different semantic reward formulations could lead to a more pronounced increase in BEq results. Separate critic models or LLMs as judges are promising avenues that can be explored in the future.

References

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang,

Table 1: Pass@50 rates for type-check, BEq, and BEq+ for Kimina's autoformalizer and our supervised model [Wang et al., 2025] on the six repositories of RLMEval [Poiroux et al., 2025].

Model	Dataset	Type-Check	BEq	BEq+
6*Kimina	PFR	83.45%	0.00%	0.00%
	Carleson	82.88%	0.00%	0.00%
	FLT	30.36%	0.00%	3.57%
	FLT3	76.19%	2.38%	3.57%
	Prime	90.91%	2.02%	3.03%
	Testing	43.55%	0.00%	0.00%
6*SFT	PFR	71.72%	8.28%	8.97%
	Carleson	66.66%	3.60%	3.60%
	FLT	64.29%	10.71%	19.64%
	FLT3	94.05%	29.76%	33.33%
	Prime	95.96%	13.13%	15.15%
	Testing	68.55%	4.84%	5.65%

Table 2: Our final reinforcement learning model achieves comparable pass@50 BEq and BEq+ scores, and significantly higher type-check (TC) rates on RLMEval [Poiroux et al., 2025].

Model	TC@1	TC@50	BEq	BEq+
SFT	17.29%	76.41%	10.66%	12.60%
RL	**26.01**%	**85.46**%	10.82%	12.76%

Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. URL `http://arxiv.org/abs/2501.12948`.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs. In *The Eleventh International Conference on Learning Representations*, 2023.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. URL `http://arxiv.org/abs/2312.14925`.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

The LeanProver-Community. A read-eval-print-loop for Lean 4, June 2025. URL https://web.archive.org/web/20250602110502/https://github.com/leanprover-community/repl.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. Rethinking and improving autoformalization: Towards a faithful metric and a dependency retrieval-based approach. URL https://openreview.net/forum?id=hUb2At2DsQ.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 Theorem Prover and Programming Language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, page 625–635, Berlin, Heidelberg, 2021. Springer-Verlag.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. Improving autoformalization using type checking, February 2025. arXiv:2406.07222 [cs].

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. DeepSeek-Prover-V2: Advancing Formal Mathematical Reasoning via Reinforcement Learning for Subgoal Decomposition, April 2025. arXiv:2504.21801 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. URL http://arxiv.org/abs/2402.03300.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer Reinforcement Learning, June 2025. URL https://web.archive.org/web/20250602133645/https://github.com/huggingface/trl.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-Prover Preview: Towards Large Formal Reasoning Models with Reinforcement Learning, April 2025. arXiv:2504.11354 [cs].

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. URL https://openreview.net/forum?id=BJgnXpVYwS.