Reinforcement Learning for Term Rewrite Systems

Robin Rawiel
Osnabrück University
rrawiel@uos.de

Lukas Niehaus
Osnabrück University
luniehaus@uos.de

January 31, 2025

1 Introduction

In recent years, Large Language Models (LLMs) have been increasingly applied to mathematical and reasoning tasks.

These models take significant amounts of training and yet are still capable of dreaming up incorrect solutions or struggle with simple algebraic tasks. For this reason, we investigate methods to improve the performance of models on symbolic reasoning tasks. Term Rewrite Systems (TRS) offer an intuitive, reliable, verifiable and Turing-complete reasoning framework on formal languages.

With this work, we want to explore the potential of combining TRS with Reinforcement Learning (RL). We introduce *treewrite*, a Python library designed to facilitate the integration of TRS with neural networks, enabling efficient problem-solving through RL.

2 Related Work

TRS have long been used in a variety of applications, ranging from declarative programming languages [1] to Theorem Proving (TP) [2] and can even be applied to chemical transformation searching [4]. Recent advancements in machine learning, and in particular RL have had promising impacts on current research.

Recently, Xin et al. improved Deepseek [9], an LLM tree search architecture to present DeepSeek-Prover-V1.5 [13], an LLM for functional programming [7]. After training the LLM, they apply Reinforcement Learning from Proof Assistant Feedback (RLPAF) for fine-tuning.

Piepenbrock et al. [8], on the other hand, present a neural rewriting system along with the Stratified Shortest Solution Imitation Learning (3SIL) training method. They use the system to assist the Prover9 [6] Automated Theorem Prover (ATP) and significantly improve their performance. Consequently, we aim to build on this work and optimize our training process, by including synthetically generated episodes.

3 Methodology

We present our Python library *treewrite*, designed to solve a range of challenges that can be formulated as TRS. The *treewrite* library enables users to define grammars and rules for TRS and swiftly integrate them into RL environments using the *gymnasium* framework [12].

Our library offers several key features:

- **Grammar Definition:** Grammars can be defined using Extended Backus Naur-Form (EBNF) or similar methods, ensuring easy compatibility with existing formal languages.
- **Tree Parsing & Conversion:** Converts terms to parse trees and vice versa, while enabling seamless integration with neural networks through tensor representations.
- **Rule Application:** Facilitates the application of rewrite rules. Rules may be conditional and can be applied to specific nodes.
- **Tokenization Scheme:** Provides efficient tokenization based on grammar specifications, allowing for node selection during the rule application.
- **Random Sampler:** Generates random and balanced trees for the provided grammar, inspired by Lample et al. [3]. These random trees serve as initial states for the RL environment.
- **Type System:** We provide a type system to ensure correctness of the parse trees. Types can be in relation with each other, e.g. the type $\mathbb{N}$ is contained within the type $\mathbb{Z}$.
- **Context Management:** Some TRS require information to be saved in a context. Our library provides an optional context class to handle the types of individual variables, their boundedness, or other necessary information.

4 Proof-of-Concept

We implemented a Boolean algebra system within the *gymnasium* framework [12], where the goal is to transform terms into Conjunctive Normal Form (CNF). In the environment, a state is represented as a parse tree, while an action consists of a rule to be applied and the node in the tree, where the rule should be applied. The RL agent receives rewards based on successful transformations or penalties for invalid actions as well as actions that result in a state that has been seen during the same episode. After training with *stable-baselines3*'s [10] PPO [11] algorithm, we achieved a 100% success rate on random terms. The nature of this total success rate is due to the trivial nature of the problem. An agent only needs to apply five different rules [5], whenever applicable and will always reach the CNF in a fixed number of steps.

Ultimately, we aim to apply this method to more complex mathematical environments, including TP.

References

[1] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J.F. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002. Rewriting Logic and its Applications.

[2] Jieh Hsiang, Hélène Kirchner, Pierre Lescanne, and Michaël Rusinowitch. The term rewriting approach to automated theorem proving. *The Journal of Logic Programming*, 14(1):71–99, 1992.

[3] Guillaume Lample and François Charton. Deep learning for symbolic mathematics, 2019.

[4] Martin Mann, Heinz Ekker, and Christoph Flamm. The graph grammar library-a generic framework for chemical graph rewrite systems. In *Theory and Practice of Model Transformations: 6th International Conference, ICMT 2013, Budapest, Hungary, June 18-19, 2013. Proceedings 6*, pages 52–53. Springer, 2013.

[5] Kim Marriott and Peter J Stuckey. *Programming with constraints: an introduction*. MIT press, 1998.

[6] William McCune. Release of prover9. In *Mile high conference on quasigroups, loops and nonassociative systems, Denver, Colorado*, 2005.

[7] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, pages 625–635, Cham, 2021. Springer International Publishing.

[8] Jelle Piepenbrock, Tom Heskes, Mikoláš Janota, and Josef Urban. Guiding an automated theorem prover with neural rewriting. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning*, pages 597–617, Cham, 2022. Springer International Publishing.

[9] Tanya Piplani and David Bamman. Deepseek: Content based image search & retrieval, 2018.

[10] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[12] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff,

Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024.

[13] Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search, 2024.