Granularity in Reward Shaping for Automated Theorem Proving Without Prior Knowledge

Marcelo Sena

June 9, 2025

Abstract

The automation of formal theorem proving using deep reinforcement learning (RL) presents a promising avenue for advancing mathematical and software verification. A critical challenge in this domain is the design of effective reward functions to guide an agent through the vast search space of possible proof steps. While dense rewards are known to be superior to sparse ones, the optimal granularity of this dense signal remains an open question. This project investigates this question within a simulated Lean 4 theorem-proving environment. We develop a policy-gradient-based RL agent and task it with proving two distinct, fundamental theorems: the commutativity of addition and the identity of addition with zero. The agent operates without any prior knowledge and within a fixed action space of 18 common and distractor tactics. The core of the project compares two dense reward paradigms: (1) a standard dense reward, which provides a uniform positive signal for any step that correctly advances the proof; and (2) a more fine-grained structure that provides larger rewards for more "critical" steps, includes a per-step time penalty to encourage efficiency, and applies stronger penalties for irrelevant actions. The results demonstrate that the denser reward does not necessarily achieve a higher proof success rate, suggesting that random exploration is critical in the presence of little knowledge, typical for example for proofs which are at the frontier of verifiable domains currently outside the scope of human knowledge.

Contents

1 Introduction

Interactive Theorem Provers (ITPs) like Lean [3], Coq, and Isabelle/HOL have revolutionized formal verification, enabling the creation of machine-checked proofs of immense complexity. Despite their power, the process of writing formal proofs remains a significant bottleneck, requiring substantial human expertise and effort. Automating this process is therefore a key objective in computer science and artificial intelligence.

Reinforcement learning (RL) offers a compelling framework for this task, modeling a prover as an agent that learns to navigate the complex state space of proof goals by selecting from a set of available proof tactics. A central challenge in applying RL to this domain is the "credit assignment problem": how to reward an agent for making progress in a long and intricate proof. The simplest approach, a sparse reward upon proof completion, often fails due to the immense difficulty of accidental discovery.

This has led to the exploration of dense reward structures, or reward shaping, where intermediate feedback is provided. While it is widely accepted that dense rewards are beneficial, the optimal design and granularity of these signals are not well understood. Does a simple signal for any forward progress suffice, or is a more nuanced function that reflects the "quality" of a step necessary?

This paper investigates this question of reward granularity. We construct a simulated Lean 4 environment where an RL agent is tasked with proving two different theorems. We implement and compare two distinct dense reward structures: a "standard" one that rewards any correct step, and one that incorporates domain heuristics like step importance and proof efficiency. Our primary hypothesis is that the denser reward may not necessarily improve the proof success rate, given the importance of random exploration in the presence of little knowledge. Our findings confirms this. These findings have ramifications for policy learning in domains that are currently beyond the scope of human knowledge, akin to the little prior knowledge of the agent in our environment.

2 Related Work

The application of machine learning to automated reasoning has evolved significantly. Early approaches often relied on supervised learning with handcrafted features to predict the next tactic, as seen in systems like TacticToe [2]. The advent of deep learning enabled more powerful models. [4], for instance, demonstrated that large language models can generate proofs by treating them as a sequence generation task.

The RL paradigm for ATP, which frames proving as a sequential decision-making game, has been explored in several major projects. The work on proving theorems in HOL-Light [1] showcased the power of combining graph neural networks for state representation with Monte Carlo Tree Search. More recently, 'LeanDojo' [5] has provided a comprehensive and scalable environment for training RL agents directly within Lean 4, establishing a modern benchmark for the field.

Our work situates itself within this RL-for-ATP context. However, instead of aiming for state-of-the-art performance on a broad corpus, our project adopts a more focused, analytical approach. We intentionally simplify the state representation and agent architecture to create a controlled laboratory for the express purpose of isolating and studying the impact of reward function design. While prior work has acknowledged the importance of rewards, our contribution is a direct, empirical comparison of different dense reward granularities, providing quantitative evidence for the impact of more informed reward shaping.

3 Methodology

Our experimental framework consists of a simulated Lean environment, a policy-gradient RL agent, and the two dense reward structures being compared.

3.1 Simulated Multi-Theorem Environment

To test for generalization, the environment tasks the agent with proving one of two theorems, chosen at random at the start of each episode.

- **Target Theorems:**

1. `add_comm (x y : Nat) : x + y = y + x`
2. `zero_add (n : Nat) : 0 + n = n`

- **State Representation:** The state is a string representing the current primary proof goal (e.g., `x + y = y + x`). While simplistic, this is sufficient for the agent to distinguish between the limited number of states in our simulated proofs.
- **Action Space:** The agent selects from a fixed set of 18 tactics. This space includes the necessary tactics for both proofs (`rw [Nat.add_comm]`, `rfl`, `induction y`, `simp`, `rw [Nat.add_succ]`) as well as numerous plausible but incorrect "distractor" tactics to create a challenging search problem.

3.2 Agent Architecture

We implement a simple policy-gradient agent based on the REINFORCE algorithm. The agent's policy is represented by a neural network.

- **Network:** A Multi-Layer Perceptron (MLP) with one hidden layer of 64 units takes a one-hot encoding of the current state string as input. The output layer produces a probability distribution over the 18 possible tactics using a softmax activation.
- **Learning Algorithm:** The network is trained using the REINFORCE algorithm with a baseline to reduce variance. At the end of each episode, the policy is updated to increase the probability of actions that led to a high cumulative reward.

3.3 Reward Structures: A Comparative Study

The central comparison of this study is between two different dense reward functions.

1. **Standard Dense Reward:** This structure provides a simple, uniform positive reward for any step that correctly follows the predefined optimal proof trace.

$$r_t = \begin{cases} +10.0 & \text{if proof is complete} \\ +2.0 & \text{if step is correct} \\ -0.1 & \text{if step is incorrect} \end{cases}$$

2. **Goal-Proximity Dense Reward:** This is a more fine-grained, heuristic-driven structure. It introduces an efficiency penalty and provides a non-uniform reward based on the perceived importance of the step.

$$r_t = \begin{cases} +10.0 - 0.5 \times t & \text{if proof is complete} \\ +5.0 - 0.5 \times t & \text{if step is a critical correct step} \\ -1.0 - 0.5 \times t & \text{if step is incorrect} \end{cases}$$

where t is the current time step. This structure explicitly incentivizes the agent to not only find a correct proof, but to find the shortest one.

3.4 Training Hyperparameters

The following hyperparameters were used for all training runs to ensure a fair comparison.

Table 1: Key Training Hyperparameters

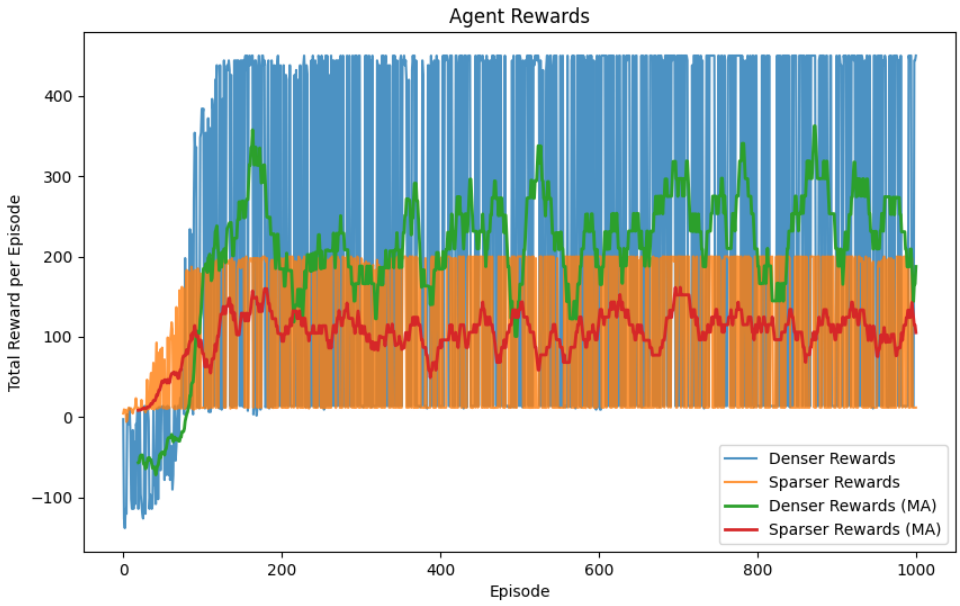
Hyperparameter	Value
Total Training Episodes	1000
Max Steps per Episode	100
Learning Rate	0.01
Discount Factor (γ)	0.99
Optimizer	Adam

4 Results

We evaluate the two reward structures based on their learning dynamics during training and the proof success rates after training.

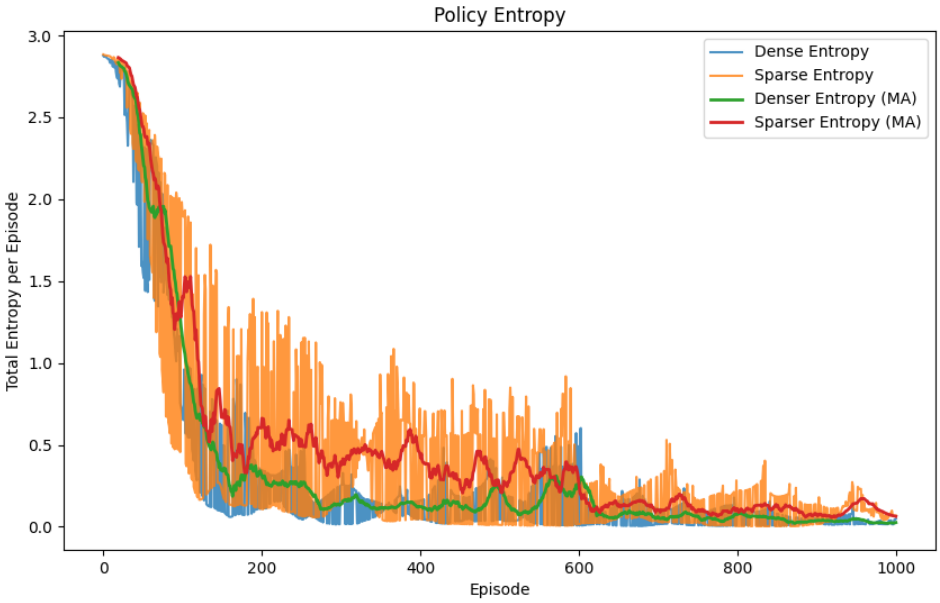
4.1 Learning Dynamics

Figure 4.1 plots the moving average of the total reward per episode for both agents over the 1,000 training episodes.



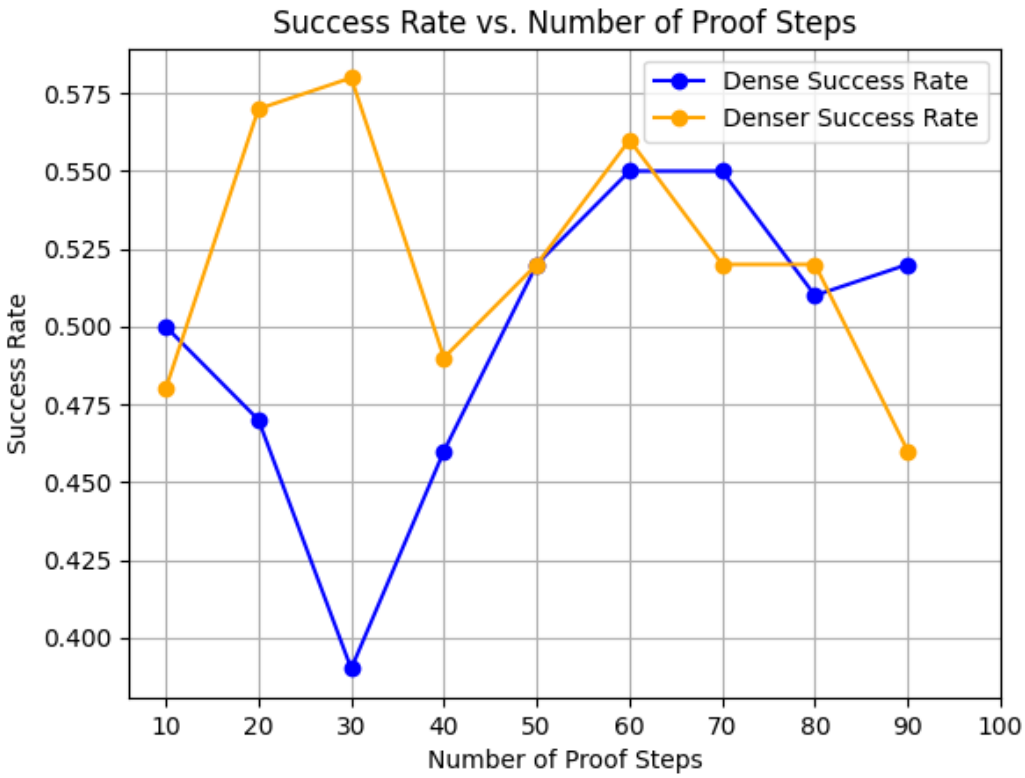
The plot clearly shows that the agent trained with the goal-proximity reward structure learns faster. It reaches a stable, high-reward policy within approximately 100 episodes, even if with more noise. The standard dense agent learns less noisily, with a final policy being more stable.

We also compare the entropy of both policies, and the denser reward training achieves a lower entropy policy, as expected.



4.2 Quantitative Evaluation of Final Policies

After training, we evaluated the final deterministic policy of each agent over fresh proof trajectories. We allowed the agents to find the proof in a maximum of N steps, where we vary N from 10 to 90. The results are summarized in the plot below.



The denser reward is better at proving when the number of allowable steps is short, which is not totally surprising given the time step penalty we provided during training. However, with a larger number of steps allowed, the sparser trained agent achieves a comparable performance and being sometimes even better. This suggests that the sparser reward, which effectively enables a random exploration of the state space, can be more beneficial than a denser reward, even when the reward is providing some signal. This is likely a consequence of the large state space the agent faces in this environment, which is typical of environments that RL faces at the frontier of human knowledge, where there is little prior expertise.

5 Discussion

The experimental results strongly suggest that for complex, multi-task symbolic reasoning problems, the detailed structure of the reward function is important. Moreover, learning better exploration policies is critical. Lower entropy policies may not necessarily be beneficial, as it is preventing exploration of otherwise profitable trajectories.

6 Conclusion and Future Work

This project successfully designed and executed a controlled experiment to compare the granularity of dense reward signals for an RL agent in a simulated theorem-proving environment. We demonstrated the non-triviality of reward shaping and state space explorations in domains where the RL agent has zero prior knowledge.

6.1 Future Work: Incorporating Backtracking

A key limitation of our current agent is its inability to recover from mistakes. Once an incorrect tactic is applied, the agent is stuck in a potentially failed proof state and must wait for the episode to terminate. A

more sophisticated agent should have the ability to backtrack or "undo" previous steps.

This could be implemented by adding a special undo action to the agent's action space. The state representation would need to be augmented to include a history of recent proof states. A key research question would then be how to design the reward for this new action. A small negative reward for undo could teach the agent to avoid mistakes in the first place, while still allowing it to escape from unproductive proof branches. This would enable more complex and strategic exploration, moving the agent's capabilities closer to that of a human mathematician.

Contributions

This project was conceived and executed by Marcelo Sena. All aspects, including the design of the environment, implementation of the RL agent, execution of experiments, and writing of this report, were handled by the sole author.

References

[1] Bansal, K., et al. (2019). *HOList: An Environment for Machine Learning of of Higher Order Logic Theorem Proving.* Proceedings of the 36th International Conference on Machine Learning (ICML).

[2] Ganku, G., et al. (2017). *TacticToe: Learning to Prove with Tactics*

[3] de Moura, L., et al. (2021). *The Lean 4 Theorem Prover.*

[4] Polu, S., et al. (2020). *Generative Language Modeling for Automated Theorem Proving.* arXiv preprint arXiv:2009.03393.

[5] Yang, K., et al. (2023). *LeanDojo: An Environment for Machine Learning of Formal Theorem Proving in Lean 4.* Advances in Neural Information Processing Systems (NeurIPS).