Automating proofs of lattice inequalities in Coq with reinforcement learning and duality

Anshula Gandhi[1,2], Favio E. Miranda-Perea[2], and Lourdes del Carmen González Huesca[2]

[1] MIT International Science and Technology Initiatives, Massachusetts Institute of Technology
[2] Departamento de Matemáticas, Facultad De Ciencias, Universidad Nacional Autónoma de México

Abstract

We have developed a reinforcement learning agent in Coq that, without the aid of neural networks or human training data, can prove basic lattice equalities and inequalities. Additionally, since the duality lemma is often heavily utilized in lattice theory, it was natural to develop a Coq tactic that can apply the duality lemma and to equip the agent with this tactic, allowing the agent to prove theorems in a single step if its dual statement has already been proven in the Coq context. Proof by duality is heavily employed in pen-and-paper proofs in lattice and order theory, and this work is a first step to bringing that proof tactic to formalized, automated, and artificially intelligent proofs.

Introduction Certain types of statements are equivalent to their duals in many fields of mathematics, including logic (involving interchanging conjunctions and disjunctions), geometry (involving interchanging points and lines), and lattice theory (involving interchanging meets and joins). As a result, mathematicians frequently rely upon duality in their pen-and-paper proofs of theorems in these fields. In this paper, we apply the duality lemma in the context of lattice theory.

We seek to lay the foundation to apply duality in the realm of computer-assisted proofs: including formally verified proofs and artificially intelligent proof search. We develop a reinforcement learning agent [1] equipped with the definition of a lattice (formalized by its order-theoretic, rather than algebraic, definition), the duality lemma, and basic Coq theorem-proving actions.

We share two main contributions:

- a Python reinforcement learning agent for lattice theory
- a Coq duality tactic for lattice theory.

Reinforcement Learning Agent for Lattice Theory We develop an infrastructure for a neural-network-free Python-based reinforcement learning agent to conduct proof search in lattice theory. In particular, the Q-learning reinforcement learning environment [11] is designed as follows:

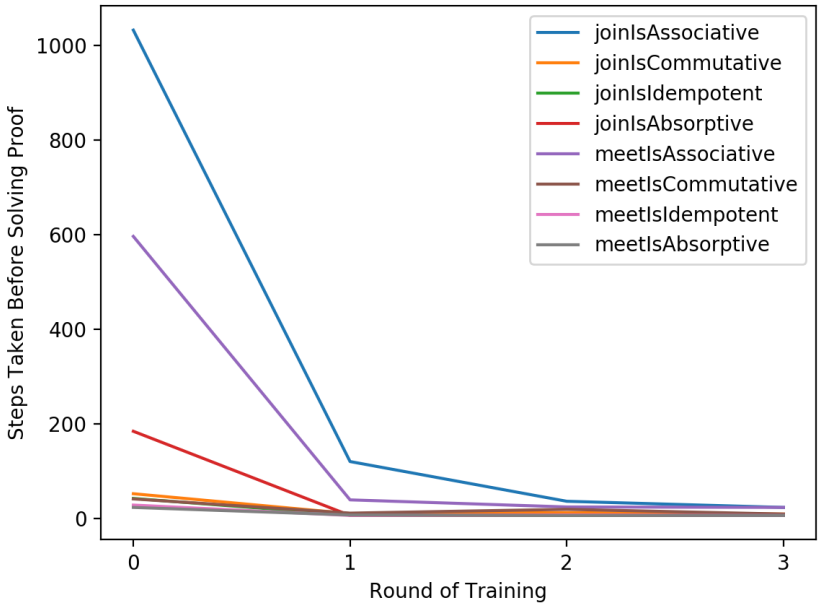
- *States:* The environment state is a string containing a Coq expression of the current subgoal, e.g. `a ⊓ b = b ⊓ a`. Theoretically, the state space could become arbitrarily large.
- *Actions:* Each action the agent takes translates to a Coq tactic. The action space is limited to 11 actions: `reflexivity`, `split`, `apply antisymmetric`, `apply join_definition`, `apply meet_definition`, `apply join_is_upper_bound`, `apply meet_is_upper_bound`, `apply transitivity with y:=(a ⊔ b)`, `apply transitivity with y:=(b ⊔ c)`, `apply transitivity with y:=(a ⊓ b)`, `apply transitivity with y:=(b ⊓ c)`.

[1] The code for this project builds off a few sources. The Python reinforcement learning framework is built off a project of the first author at MIT's Brains, Minds, and Machines lab, while the Coq implementation of lattices was inspired by the work of D. James and R. Hinze [7], and the code that interfaces between Python and Coq was built off the work done by D. Huang et al. [6].

- *Rewards:* The agent receives a reward of -1 when it attempts to take an action that produces a Coq error, a reward of 1 when it finishes the theorem, and 0 otherwise. Somewhat counter-intuitively, we find this sparse rewards system more effective for lattice theory than some other denser rewards systems, since it generalizes more easily to a greater variety of lattice inequalities.

The set of training and testing problems are the same 8 problems. In particular, we train on a set of eight core equalities of lattice theory (namely, associativity, commutativity, idempotence, and absorption properties of meet and join both). Because methods for simplifying lattice inequalities usually follow the same pattern, the agent needs a minimal amount of training data before being able to generalize. As the agent keeps training, it incurs fewer and fewer penalties, and takes fewer and fewer steps to solve the theorem. After just three rounds of training, the agent can solve theorems in a seemingly minimal number of steps (that is, the same number of steps as a clean human-derived proof).

Figure 1: Sample training progress



This is a snapshot of the Q-table after training on a set of core lattice inequalities, correctly representing the theorem-proving actions to apply to certain inequalities. These results are achieved solely through self-play – without human training data.

State	Recommended Action
$a \leq a$	`reflexivity`
$(a \sqcap b \leq b) \wedge (a \sqcap b \leq a)$	`split`
$(a \sqcap b) \sqcap c \leq a$	`apply transitivity with (y := a ⊓ b)`
$a \sqcup b \leq b$	`apply join_is_sup`
$(a \sqcap b) \sqcap c = a \sqcap (b \sqcap c)$	`apply antisymmetric`

Duality Tactic for Lattice Theory In order to further optimize the performance of the agent, we develop a Coq tactic[2] that applies the duality lemma to prove theorems.

```
Ltac dualize_goal :=
   apply complement_both_sides;
   repeat (rewrite demorgans_law_meet || rewrite
        demorgans_law_join).

Ltac duality :=
   dualize_goal;
   auto with mylatticeproofs.
```

In particular, the above tactic takes a lattice equality or inequality, takes the complement of both sides, repeatedly applies DeMorgan's law to unfold the statement into its dual, and if the dual statement already exists in the Coq database `mylatticeproofs`, asserts that the dual statement proves the original statement.

We prove the correctness of this tactic for all boolean lattices, including formally verifying that DeMorgan's law holds for boolean lattices, and thus that the dual statement is equally valid.[3] Importantly, while this tactic is recursive in nature, it does not require implementation of inductive type classes describing lattice elements, as other previous research has done when developing recursive tactics in lattice theory [7]. As a result, this tactic can more easily be modified to generalize to other fields of mathematics that rely upon duality. After the reinforcement learning agent is equipped with this Coq tactic, it can prove theorems in as little as one step (by using the Coq command `duality`). The agent usually doesn't take more than three rounds of training to converge to this one-step proof.

Conclusion Notably, the implementation of this *reinforcement learning agent is such that it is general enough that it can be minimally modified to be applied to proof search in different fields*: the reward function is not specific to lattice proofs and thus is generalizeable, and only the agent's lattice-theory-specific actions would have to be replaced with actions more suitable to other fields. Similarly, the *duality tactic is general enough that it can be minimally modified to prove duality in other areas of math* including logic, category theory, or geometry, due to its minimal reliance on lattice-specific inductive type classes.

Reinforcement learning seems particularly promising for proving math theorems. Neural networks excel at approximating functions and therefore are able to classify types of algebraic structures [5], approximate how many steps are left in a proof [6], or suggest proof steps based on human training data [3, 4]. Reinforcement learning agents exercise a complementary set of advantages: namely, they can explore a set of theorem proving actions and discover entire proofs through self-play [1, 2, 8, 12]. The triumph of self-playing reinforcement learning agents in games such as Go and Dota 2 demonstrates the power of self-playing machines in video games [10, 9] though the testing of self-playing machines in mathematical theorem proving is still in its early stages [2]. We demonstrate the potential of such an artificially intelligent agent to prove lattice inequalities through reinforcement learning and duality, and lay the framework to extend this intelligence to more fields of mathematics.

[2]The full code for this tactic is available at github.com/anshula/duality-tactic-for-lattice-theory

[3]Duality also applies generally to all free lattices. However, the dual of a particular lattice element is not well defined for free lattices, which makes a Coq implementation of duality for free lattices a challenge for future work.

References

[1] Kshitij Bansal, Sarah Loos, Markus Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning*, pages 454–463, 2019.

[2] Alhussein Fawzi, Mateusz Malinowski, Hamza Fawzi, and Omar Fawzi. Learning dynamic polynomial proofs. *arXiv preprint arXiv:1906.01681*, 2019.

[3] Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Tactictoe: Learning to reason with HOL4 tactics. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 125–143. EasyChair, 2017.

[4] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Learning to prove with tactics. *CoRR*, abs/1804.00596, 2018.

[5] Yang-Hui He and Minhyong Kim. Learning algebraic structures: Preliminary investigations. *arXiv preprint arXiv:1905.02263*, 2019.

[6] Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. Gamepad: A learning environment for theorem proving. *arXiv preprint arXiv:1806.00608*, 2018.

[7] Daniel WH James and Ralf Hinze. A reflection-based proof tactic for lattices in coq. In *Trends in Functional Programming*, pages 97–112, 2009.

[8] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems*, pages 8822–8833, 2018.

[9] OpenAI. How to train your openai five. https://openai.com/blog/how-to-train-your-openai-five/, 2019.

[10] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[11] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

[12] Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. *arXiv preprint arXiv:1905.09381*, 2019.