10-2-2019

Machine Learning to Support an Interactive Theorem Prover

Salman Haider
DePauw University

Andy Le
DePauw University

Echo Wu
DePauw University

Brian T. Howard
DePauw University, bhoward@depauw.edu

Machine Learning to Support an Interactive Theorem Prover

Salman Haider, Andy Le, and Echo Wu
Brian Howard
Computer Science Department, DePauw University, Greencastle, IN, 46135

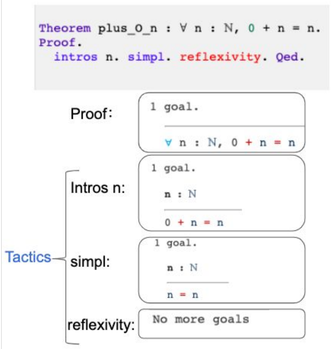
Abstract

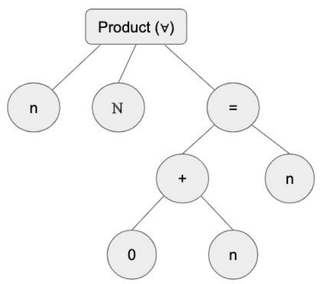
An **Interactive Theorem Prover (ITP)** is a computer program that can assist a human in creating a proof of a mathematical theorem or the correctness of a piece of software. At each step in the proof, the human has the computer apply a chosen "tactic" to attempt to make progress toward the goal. We are exploring the possibility of using **Machine Learning (ML)** to assist in this tactic selection. Building on recent work at UC Berkeley and Princeton, where they adapted the Coq ITP so that it could interface with ML libraries via the Python scripting language, we have been evaluating strategies to encode the current proof state to try to improve the accuracy of tactic prediction.

Interactive Theorem Proving

Interactive Theorem Proving (ITP) involves the use of a computer program to assist the development of formal proofs by human-machine collaboration. One of the popular ITP is **Coq**, which is a formal proof management system with a functional programming language -- **Gallina**. Moreover, a language of **Tactics** in Coq guide the process of developing the steps of the proof. We use **Abstract Syntax Tree (AST)** as a way to present the syntax of programming language as a tree-like structure.

```
Theorem plus_O_n : ∀ n : N, 0 + n = n.
Proof.
  intros n. simpl. reflexivity. Qed.
```



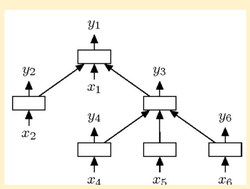


Machine Learning

Machine Learning: A technique to infer parameters to make a model fit patterns in a given set of data.

We are using Machine Learning to automate the interaction with **proof assistants.**

We use a **TreeLSTM** on ASTs to feed the input goals and premises. This type of model allows us to encode tree topologies (ASTs). The model is trained on a set of proofs and evaluated on the accuracy of tactic prediction.



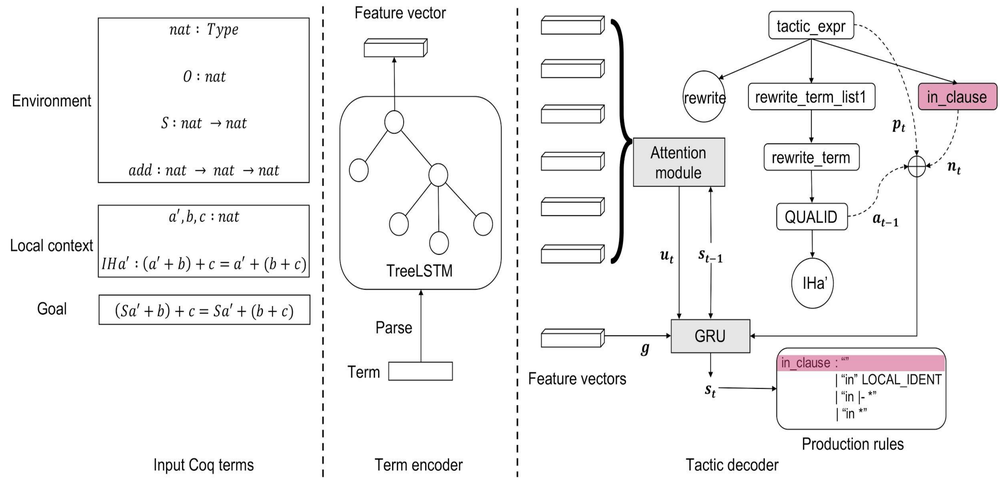
Previous Work

GamePad and **CoqGym** are prior recent projects on this topic. **GamePad** was released in the summer of 2018, **CoqGym** was a continuation of **GamePad**, released in the summer of 2019.

Both **GamePad** and **CoqGym** include a tool for interacting with **Coq.**

GamePad was trained on 1,600 theorems leading to a proof of the Feit-Thompson Theorem in group theory, whereas **CoqGym** was trained on 71,000 theorems, covering a broad spectrum of both mathematics and program verification.

CoqGym generates complete tactics that can be used to obtain full proofs, whereas **GamePad** group all tactics into categories and only predicts the category, not the specific tactic.



(Image from Yang & Deng, "CoqGym")

Experiments

Our main focus of this summer project is to figure out ways to improve the **CoqGym** model to attain better tactic recommendations for humans to develop proofs. In order to achieve this, we conducted four experiments, as listed below:

0. Original CoqGym model	17%
1. Restrict to program verification proofs	17%
2. Trivial AST input	7%
3. Leave tokens in AST	17%
4. Generate tactic_list	(in progress)

0. We ran the original **CoqGym** model and got 17% accuracy on tactic prediction.

1. As mentioned previously, **CoqGym** was trained on over 71,000 theorems, covering a broad spectrum of both mathematics and program verification. Since we are only interested in the latter, we restricted the training and testing data to see if the machine can make more relevant connections. However, we did not see a significant change.

2. In order to make sure the inputs indeed have an impact on the model, we ran a version with trivial input. Rather than having a complex tree-like structure of data, we changed it to a single node. As a result, we ran into a notable drop, to 7% accuracy.

3. In the original **CoqGym** model, the tokens that hold module and identifier names were removed. Based on the second experiment, we hoped that more information will yield better results. Hence, we kept all these tokens and ran the model. Again, we did not see any changes.

4. The original **CoqGym** model calculated its accuracy based on a single prediction of the tactic. In this experiment, we will aim to change the model's prediction to a list of plausible tactics, rather than only one. Since a human is involved in the process of constructing proofs, we assume that they can cognitively choose the correct tactic from that list. We are still in the process of running this experiment, we are also the most optimistic about this one.

Conclusions and Future Work

Although we were able to confirm that the CogGym model is able to learn some patterns in the training data, we were unable to improve the tactic prediction accuracy beyond 17%. This is unlikely to be helpful as a hint tool in an interactive theorem prover.

One idea for future work is to modify the tactic decoder so that it generates a *list* of suggested tactics, where the loss score will be low if any tactic on the list matches the expected (ground truth) tactic.

If the model can be modified to increase the accuracy rate, then further work will focus on integrating the tactic suggestions into an interface for Coq, such as CoqIDE. The long-term goal is to provide a tool that will help both students and professional programmers in producing verified software.

Given a model with sufficient accuracy that we could integrate into a system for interaction with Coq, our next step would be to conduct user studies to see if the tactic suggestions are indeed useful, for a range of possible users: students who are just learning to use Coq, experienced programmers who are attempting to produce verified software, and current Coq users looking for an additional tool to assist in their work.

References

- Coq: The Coq Development Team (INRIA, et al.), `https://coq.inria.fr/`
- GamePad: Daniel Huang and Dawn Song (UC Berkeley), Prafulla Dhariwal and Ilya Sutskever (OpenAI), `https://github.com/ml4tp/gamepad`
- CoqGym: Kaiyu Yang and Jia Deng (Princeton), `https://github.com/princeton-vl/CoqGym`

Supported by the DePauw Science Research Fellows Program, the J. William Asher and Melanie J. Norton Endowed Fund in the Sciences, the Information Technology Associates Program and the Tenzer Technology Center, the DePauw Computer Science Department, and Google Cloud.