Tactic Characterizations by the Influences on Proof States *

Liao Zhang[1,3] and Lasse Blaauwbroek[2]

[1] Czech Technical University, Prague, Czech Republic
[2] Institut des Hautes Études Scientifiques, France
[3] University of Innsbruck, Austria

1 Introduction

When formalizing mathematics in an interactive theorem prover, such as the Coq [1] proof assistant, it is necessary to have an intuition on how the available proof actions change the proof state. In particular users may have an idea that to transform the current proof state to a different one, a particular tactic might be the right one to use.

In this paper, we regard the changes to a proof state made by the tactic application as the semantic of that tactic. The purpose of our study is to predict the tactic based on its semantic. Assume there is a triple $(ps, t, \{ps'\}_{1..n})$, where ps, t, $\{ps'\}_{1..n}$ are a Coq state, the tactic applied to ps by a Coq user, and the after states caused by the tactic application, respectively. We aim at building a machine learning model to predict a tactic t' such that ps transforms to $\{ps'\}_{1..n}$ by the application of t'. To ensure that t and t' lead to the same after states, we run t' in Coq and compare with t.

There are several motivations behind our project. First, the task can be directly applied for tactic suggestion given a human's intuition for the next state. For a Coq beginner, it is quite common that he can imagine the next state but cannot determine how to select a suitable tactic to reach the goal. However, understanding Coq's manual may be challenging for beginners. If he can copy the before state from the Coq editor, convert it to the imaginary after state, and input them into our system, we will be able to automatically suggest the tactics with the expected behavior. Meanwhile, a medium-level Coq programmer may want to discover a single tactic to substitute an awkward tactic sequence. Even for an expert, when he encounters an unfamiliar domain, he needs our system to advise likely helpful tactics.

Second, the task serves as an initial step to a new formal verification strategy. When a mathematician tries to prove a theorem, he first thinks of several intermediate goals and then fills the gaps by order. However, nowadays proof assistants cannot skip tactics between intermediate goals. We can extend our system to predict a tactic sequence from one state to another. Afterwards, the human expert can merely specify the states that he thinks are important to complete the proof and ask our system to erase the gaps.

Finally, since we encounter our own challenges in precisely characterizing the transition between before and after states, the approaches developed by us can also be applied to other machine learning domains. Take fault detection [3] for instance, we can apply our differential techniques to the images before and after the fault occurs. Then, the results can be input into a learning model to predict the category of fault.

*This work was supported by the ERC grant no. 714034 *SMART* and by the European Regional Development Fund under the project AI&Reasoning (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000466).

2 Tactic Characterizations

We characterize the semantic of tactics as features and Coq strings as the input for random forests [7] and GPT-2 [5], respectively. The feature extraction techniques on Coq terms are the same as our previous work [7]. Large-scale pretrained transformers such as GPT-2 have achieved significant progress in various domains [2]. We evaluate GPT-2 and random forests to make a comparison.

We consider three feature extraction approaches. The first approach computes the differences between the state features of ps and $\{ps'\}_{1..n}$. From ps, we extract a set of features F. We also extract n sets of features $\{F\}_{1..n}$ from $\{ps'\}_{1..n}$. If a feature f exists in F but does not in any F_i, we regard it as a disappeared feature. Conversely, if there is an F_i with a feature f that is not in F, then f is an appearing feature. The tactic characterization is the union of all disappeared and appearing features.

Second, we extract features from the newly defined existential variables in proof terms. In Coq, we write tactics to construct a proof script to prove a theorem. Actually, the tactics help to complete a proof term. The relationship between proofs and proof terms is based on the Curry-Howard correspondence [6]. An incomplete proof term may contain several existential variables. Some are defined, and others are undefined as holes. A tactic fills some holes with Coq terms and may generate several new holes. A proof term is completed once all the holes have been filled. We obtain the features from the terms defined in the holes by the tactic as its characterization.

Finally, we perform first-order anti-unification [4] on the before and after states to find the substitutions. A term g of two terms t_1 and t_2 is called a *generalization* if there are substitutions σ_1 and σ_2 such that $\sigma_1 g = t_1$ and $\sigma_2 g = t_2$. Anti-unification aims to find the *least general generalization* lgg such that for any generalization g' of $t1$ and t_2, there exists a substitution σ that makes $\sigma g' = lgg$. We extract the features from the Coq terms present in the substitutions(σ_1 and σ_2) and the lgg as the input to our model.

For GPT-2, we merely apply anti-unification to generate strings. We convert the lgg and substitutions to strings and input them into the model.

3 Experimental Evaluation

Our dataset is composed of the proof states $(158,494)$ of all the lemmas $(11,372)$ in the Coq's standard library. The lemmas were randomly divided into three subsets for training, validation, and testing in an 80-10-10 ratio. Each subset includes the states of the corresponding lemmas. For random forests, we optimize parameters on the training and validation partitions, which is depicted in Figure 1. Afterwards, we build models with the best hyper-parameters learned from the training dataset and make predictions for the test dataset. We also fine-tune the smallest GPT-2 for each characterization. Every model is executed for 25 epochs, and we store the snapshot with the best accuracy on the validation dataset to synthesize tactics for the test data. All the GPT models utilize the same parameters: a batch size of 32, no weight decay, and the learning rate of 0.0003 with a linear schedule and the first 20% steps for warming up. Figure 2 depicts the average training loss per step and validation accuracy during fine-tuning.

Table 1 shows the results on the test data. Unsurprisingly, only learning from before states performs worst since it contains little information of the influences of the tactic. The best accuracy achieved by GPT-2 is 10.47% better than that of random forests. This confirms the power of the state-of-the-art neural network. Anti-unification does not work well for random

Figure 1: Results of hyper-parameter tuning for random forests. The accuracy denotes how often we predict a tactic that is the same as the tactic in the dataset.

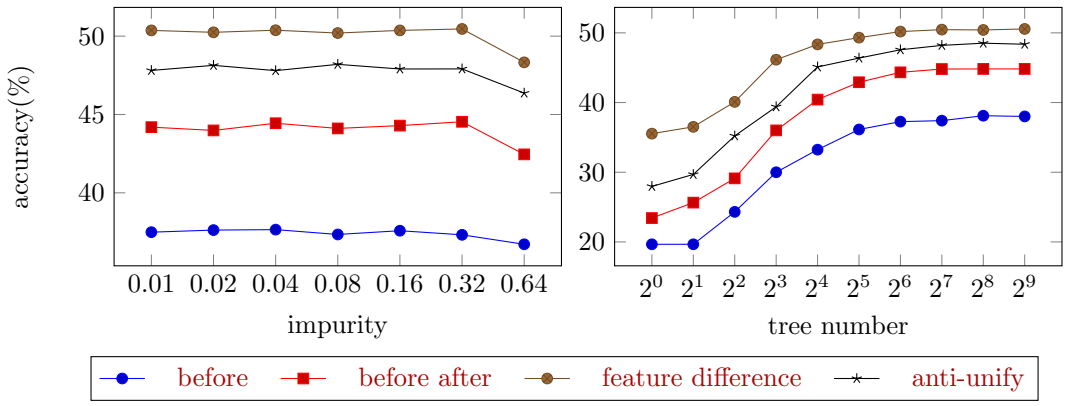
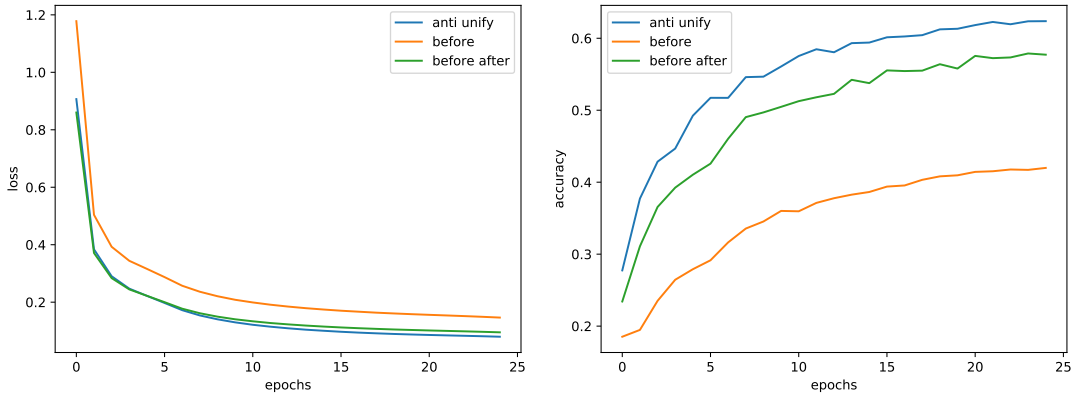


Figure 2: Training loss and validation accuracy of GPT-2.



(a) training loss (b) validation accuracy

Table 1: Results on the test dataset. "Same tactic" denotes that the prediction is exactly the same as the tactic in the library. "Same change" checks how often the prediction makes the same transformation.

model	accuracy(%)	before	before after	feature difference	proof term	anti unification
random forests	same tactic	36.917	44.563	49.723	47.480	47.727
	same change	43.225	52.166	59.344	56.024	55.507
GPT-2	same tactic	39.154	56.215			60.300
	same change	45.356	65.319			69.814

forests but obtains excellent performance for GPT-2. The reason may be that converting anti-unification to appropriate features is challenging.

References

[1] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

[2] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[3] Dubravko Miljković. Fault detection methods: A literature survey. In *2011 Proceedings of the 34th international convention MIPRO*, pages 750–755. IEEE, 2011.

[4] Gordon D Plotkin. A note on inductive generalization. *Machine intelligence*, 5:153–163, 1970.

[5] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[6] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Elsevier, 2006.

[7] Liao Zhang, Lasse Blaauwbroek, Bartosz Piotrowski, Prokop Černỳ, Cezary Kaliszyk, and Josef Urban. Online machine learning techniques for coq: A comparison. In *International Conference on Intelligent Computer Mathematics*, pages 67–83. Springer, 2021.