Learning to Generate Abstractions for an Equational Solver

Guy Axelrod, Devdatt Dubhashi, Moa Johansson, Andrea Silvi, Nicholas Smallbone, and Sandro Stucki

Chalmers University of Technology, Gothenburg, Sweden

Abstract

We propose, and plan to present early results on, an approach to using LLMs for the generation of useful abstractions in the context of equational reasoning. Specifically, given a partial proof search, we consider the task of providing definitions to the Twee solver with the aim of speeding up convergence. We outline a pipeline for generating supervised fine-tuning data (via synthetic conjectures and symbolic abstraction heuristics) and for continuously improving the LLM's suggestions through iterative refinement with feedback from Twee (via Reinforcement Learning).

Like other automated theorem provers for equational logic, Twee uses a DISCOUNT loop to implement unfailing Knuth-Bendix completion. In addition, it supports goal direction by adding axioms containing constants which define subterms of the goal. It is shown that this simple form of goal direction can lead to significant speedups for certain problems (a similar "goal definition" feature exists as an option for E). Generalizing slightly, we wish to consider the setting in which we allow ourselves to introduce axioms that contain (possibly functional) constants defining arbitrary, syntactically valid terms. We call such axioms *abstractions*.

Given the successes of simple, goal-derived abstractions, a natural question to ask is: for a given axiom set, goal and partial proof search in Twee, which more general abstractions will augment search in a way that leads to faster convergence? The idea of introducing definitions or intermediate lemmas to ease proof search has a long history in automated reasoning. Veroff's hints technique [11], for instance, showed that providing previously discovered lemmas as hints can dramatically aid provers in tackling hard problems in loop theory.

We propose to approach the automation of this task via the iterative finetuning of a LLM, in which training signals are shaped by feedback from Twee. As a baseline, both for the purposes of comparison and pretraining, we will survey and adapt various previous symbolic approaches suitable to this task [12, 14, 13, 7, 1]. Various learning-based approaches have also been proposed [6, 3, 5]. Similar in spirit to our approach is the lemma suggestion of [5]. In their experiments on the Abelian Inner Mapping (AIM) conjectures in loop theory [4], a reinforcement learning agent (using a technique called 3SIL) learned to propose rewrite steps. It was then used to rewrite conjectures to derive useful lemmas which, when augmenting Prover9, enabled the proof of about 8% more theorems than Prover9 alone. This demonstrates the promise of learned lemma generation in equational domains.[1] However, to our knowledge, no prior work has directly trained an LLM to generate definition-style abstractions to aid an equational prover[2]. We aim to provide preliminary explorations in this area, with the broader intention of evaluating the capabilities of LLMs in extracting relevant insights from proof logs. Our proposed pipeline consists of the following steps:

1. **SFT:** Perform Supervised Fine-Tuning on a base LLM to generate a syntactically valid abstraction given a prompt consisting of an axiom set, conjecture, and partial goal-directed

[1] We note that many of these previously studied techniques focus on lemma discovery (often in stronger logics), a task which can be specialized to our specific task of abstraction introduction in equational theories.

[2] Though, techniques from program synthesis and code refactoring may be relevant here – for example [2].

Twee run. To construct such an instruction-tuning dataset, we first build an input instance as follows: synthetically generate syntactically valid axioms; select some provable conjectures by running Prover9 in production mode on these axioms; perform a partial run of Twee on these axioms and conjectures. This input is then paired with desired outputs (i.e. abstractions) which are collected from the twee run using symbolic approaches, such as frequent subterm mining.

2. **RL:** Fixing some axiom set and dataset of conjectures, we enter an iterative improvement loop. In each iteration, we do the following:

   For each conjecture and corresponding partial twee run, prompt the LLM to sample multiple abstractions. For each sample, rerun Twee, now augmented with the abstraction. For the runs that converge, we derive some reward (e.g. based on the length of the derived proof) which is used to perform GRPO on the LLM [8].

   We iterate until performance saturates.

In Step 2, we rely on the presence of an existing dataset of conjectures in some fixed equational theory. This leads to highly specialized abstraction generators, which may be useful in the context of hard open problem, such as those of Verroff's AIM project [4]. Alternatively, we may also wish to produce a more general purpose tool suitable for proof guidance in arbitrary equational theories. In this case, we foresee the possible use of a curriculum learning approach, in which the LLM is dually trained to generate axiom sets which provide dense training signal for the task of abstraction generation. We hope to test this more general approach on TPTP's UEQ problems set [10], and possibly enter the CASC competition [9].

We plan to report on our preliminary implementation and results at the workshop, and we welcome feedback and suggestions from the community.

References

[1] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. Automating inductive proofs using theory exploration. In *Proceedings of the 24th International Conference on Automated Deduction*, CADE'13, page 392–406, Berlin, Heidelberg, 2013. Springer-Verlag.

[2] Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251):20220050, 2023.

[3] Cezary Kaliszyk and Josef Urban. Learning-assisted theorem proving with millions of lemmas. *Journal of Symbolic Computation*, 69:109–128, 2015. Symbolic Computation in Software Science.

[4] Michael Kinyon, Robert Veroff, and Petr Vojtěchovskỳ. Loops with abelian inner mapping groups: An application of automated deduction. In *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, pages 151–164. Springer, 2013.

[5] 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.

[6] Michael Rawson, Christoph Wernhard, Zsolt Zombori, and Wolfgang Bibel. Lemmas: Generation, selection, application. In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 153–174, Cham, 2023. Springer Nature Switzerland.

[7] R. Sekar, I. V. Ramakrishnan, and Andrei Voronkov. *Term indexing*, page 1853–1964. Elsevier Science Publishers B. V., NLD, 2001.

[8] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[9] G. Sutcliffe. The CADE ATP System Competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

[10] Geoff Sutcliffe and Christian Suttner. The tptp problem library. *Journal of Automated Reasoning*, 21:177–203, 1998.

[11] Robert Veroff. Solving open questions and other challenge problems using proof sketches. *Journal of Automated Reasoning*, 27:157–174, 2001.

[12] Jiří Vyskočil, David Stanovský, and Josef Urban. Automated proof compression by invention of new definitions. In Edmund M. Clarke and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 447–462, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[13] Christoph Wernhard and Zsolt Zombori. Exploring metamath proof structures. *AITP*, 2024, 2024.

[14] Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. egg: Fast and extensible equality saturation. *Proc. ACM Program. Lang.*, 5(POPL), January 2021.