New approaches to ATPs

Frieder Simon

University of Oxford, UK
National Institute of Informatics, Japan

Abstract

In this paper a high-level outline of two research directions is presented, aimed at improving current, automated proof systems.

1 Introduction

When it comes do *doing* mathematics—as opposed to *communicating* mathematics, for which LaTeX, citeable article databases and other instruments have become the standard—there do not exist general-purpose tools: Various symbolic algebra programs, numerical analysis suites or other more specialized tools are employed, but these are typically heavily domain-dependent. Compared to these, at the other end of the spectrum there are automated theorem provers (ATPs), which work in full generality, and could be considered to be the ultimate tool to a working mathematician; though currently they are far from being able to attain human level proving performance.

Research in ATP can roughly be divided in two approaches: One works by encoding mathematics in a formal language and reasoning on a formal level. Recently, machine learning has been successfully used in this regard to augment existing ATP approaches (e.g., lemma selection) by improving the proof search mechanism through learning; see, e.g., [7, 6] for reinforcement learning or [2] deep learning approaches.

The other approach works by completely discarding modeling mathematics with a purely formal language, and building provers whose focus is not on soundness, but on proving more advanced statements, accepting a non-zero probability that those proofs might be wrong. Here we mention [5] as a modern candidate. (We will discuss why relaxing the soundness requirement might be meaningful at the end of the next section.)

In this text we will outline how advancements in either direction might be made.

2 Proposed research

Regarding the first approach, building on the literature review[1] on reinforcement learning methods in automated theorem proving, we propose to investigate end-to-end reinforcement learning pipelines for proof learning. It is noticeable that the reward one can assign for a completed proof can be delayed for a very long time, since for given a statement no bounds on the proof length can be derived a priori. This feature appears not to be sufficiently exploited by current methods. In contrast, in the reinforcement learning community there has been recent progress in learning from highly delayed rewards [4]. In line with current popular benchmarks, the authors of [4] evaluate their approach on a set of Atari games. We propose the delayed reward system be adapted to ATPs in order to account for the problem of highly varying proof lengths, from which it can be hard to learn a proof strategy by standard reinforcement methods. As an initial starting point, we will adapt and evaluate the methods from [4] to the theorem proving domain, which is likely to come with its own challenges and obstacles. Large databases of (formalized) mathematical texts abound and can be helpful in assessing the agents choice.

In: C. Kaliszyk, E. Brady, J. Davenport, W.M. Farmer, A. Kohlhase, M. Kohlhase, D. Müller, K. Pąk, and C. Sacerdoti Coen (eds.): Joint Proceedings of the FMM and LML Workshops, Doctoral Program and Work in Progress at the Conference on Intelligent Computer Mathematics 2019 co-located with the 12th Conference on Intelligent Computer Mathematics (CICM 2019), Prague, Czech Republic, July 8–12, 2019, published at http://ceur-ws.org

[1] carried out as part of my internship at the National Institute of Informatics

Examples of such are, e.g., [10] or proof libraries associated to interactive theorem provers, such as fragments from Isabelle's *Archive of Formal Proofs* (`http://afp.sf.org`), in addition to the existing libraries specifically curated for reinforcement learning, that come with the mentioned papers above.

Regarding the second approach we propose to build a productivity tool, a mathematical knowledge miner, that could be integrated in an ATP of the second kind, similar to [5]. The goal of the mathematical knowledge miner is to map the content of a piece of stand-alone mathematical text, such as a research article, onto a knowledge representation formalism, such as an ontology or a knowledge graph, and to thereby provide an understanding of the main theorem of the text from context it is embedded in. The starting point of learning such a mapping could be an approach as in [9] (combined with [1]). Different avenues in related fields of natural language processing have to be explored in order to find the optimal approach; this will make heavy use of modern deep learning methods from the field of natural language processing.

In case of knowledge graphs, every node could be either a definition, a theorem/lemma/proposition or an example and arrows should indicate how these are related to each other. Because this is an ambitious undertaking, it would be appropriate to first focus on an easy case, where the article in question is not a modern research article, but an article as they can be found in the American Mathematical Monthly [3] magazine, which caters to a wide, mathematically literate audience.

The novelty of the proposal does not lie in creating such knowledge graphs or ontologies per se, as similar solutions that classify the contents of a research in terms already exist, see, e.g., [8]. Rather it lies in using them in the context of an ATP. Such a "compressed view" of a mathematical text, captured by a knowledge graph or an ontology, would cover a new, middle ground between the formulation of the contents of the article in natural language and the formulation of the contents in a finely grained formal language, as would be necessary if one would want current ATPs to parse the article.

It is conceivable, in order to build an ATP of the second kind, that a succession of tools that provide different higher level features of a mathematical document should first be devised, as exemplified by a knowledge graph or ontology tailored to ATP needs. Building modules that enable deep neural networks to be more successful at understanding numeracy, a starting point being [11], would be a further possibility for a bottom-up approach to work towards an ATP. Once such a succession of tools leading up to a prover would be in place, it seems reasonable to assume that one could do high-level reasoning in such an ontology of mathematical objects and infer high-level information. One—perhaps controversial—aspect of this approach will be to not require such a prover to be sound. This might be unusual, but it is an approach that is worth exploring: Requiring soundness is one of the reasons current ATPs are designed in a purely formal language, thereby producing only completely rigorous theorems (unless the Trusted Computing Module an ATP uses to do the inference has bugs). But experience shows that this approach leads to ATPs that have difficulty proving formalized versions of advanced theorems (as they can be found, for example, in graduate textbooks). Another reason not to insist on soundness is that mathematicians are also not concerned by being presented with a proof that might not be sound, since they can check soundness for themselves. Rather, it is a matter of much greater importance to allow the ATP to come up with any proof at all, that can be easily read (and checked) by a human.

3 Summary

We conclude this vision paper by underlining that machine learning is the prevalent method that will be used to devise new approaches to advance ATPs. On one hand this could be accomplished by improving current ATP systems that work on a fine-grained formal language, encoding mathematics. On the other hand machine learning approaches could help pave the way towards non-formal ATPs.

Being still at the beginning of my doctoral research, it is clear that realizing the described approaches is an ambitious undertaking, given the difficulty of the domain of automated theorem proving. However, these approaches have not yet been fully explored, so there is reason to be optimistic that at least one of the approaches will be fruitful.

References

[1] L. Abzianidze, LANGPRO. Natural Language Theorem Prover, *Proceedings of the 2017 EMNLP System Demonstrations*, pp 115–120

[2] A. Alemi, F. Chollet, N. Een, G. Irving, C. Szegedy, J. Urban, DeepMath - Deep Sequence Models for Premise Selection, *Advances in NIPS*, Vol. 29, 2016, pp. 2235–2243

[3] American Mathematical Monthly, https://www.maa.org/press/ ...

[4] J.A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, S. Hochreiter, *RUDDER. Return Decomposition of Delayed Rewards*, https://arxiv.org/abs/1806.07857

[5] M. Ganesalingam, T. Gowers, A Fully Automatic Theorem Prover with Human-Style Output, *Journal of Automated Reasoning February*, Vol. 58(2), 2017, pp 253–291

[6] D. Huan, P. Dhariwal, D. Song, I. Sutskever, GamePAD. A Learning Environment for Theorem proving, *ICLR* 2019, https://openreview.net/forum?id=r1xwKoR9Y7

[7] C. Kaliszyk, J. Urban, H. Michalewski, M. Olšák, Reinforcement Learning of Theorem Proving, *Proceedings of the 32nd International Conference on NIPS, 2018*, pp 8836–8847

[8] M. Liakata, S. Saha, S. Dobnik, C. Batchelor, D. Rebholz-Schuhmann, Automatic recognition of conceptualization zones in scientific articles and two life science applications, *Bioinformatics*, Vol. 28(7), 2012, pp 991–1000

[9] Y. Shen, Z. Lin, C.-W. Huang & A. Courville, Neural Language Modelling by Jointly Learning Syntax and Lexicon, *ICLR* 2018, https://openreview.net/forum?id=rkgOLb-0W

[10] G. Sutcliffe, C. Suttner, The TPTP Problem Library for Automated Theorem Proving, *Journal of Automated Reasoning*, Vol. 21(2), 1998, pp 177–203

[11] A. Trask, F. Hill, S. Reed, J. Rae, C. Dyer, P. Blunsom, Neural Arithmetic Logic Units, *Advances in NIPS 2018*, Vol. 31, 2018, pp. 8035–8044