EPiC Series in Computing
Volume 51, 2017, Pages 29–32
ARCADE 2017. 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements

We know (nearly) nothing!
But can we learn?

Stephan Schulz

DHBW Stuttgart
schulz@eprover.org

Abstract

The greatest source of progress in automated theorem proving in the last 30 years has been the development of better search heuristics, usually based on developer experience and empirical evaluation, but increasingly also using automated optimization techniques. Despite this progress, we still know very little about proof search. We have mostly failed to identify good features for characterizing homogeneous problem classes, or for identifying interesting and relevant clauses and formulas.

I propose the challenge of bringing together inductive techniques (generalization and learning) and deductive techniques to attack this problem. Hardware and software have finally evolved to a point that we can reasonably represent and analyze large proof searches and search decisions, and where we can hope to achieve order-of-magnitude improvements in the efficiency of the proof search.

1 Introduction

During the last 30 years, automated theorem provers for first-order logic have reached relative maturity, with some progress in refining calculi, and significant progress towards more robust and efficient implementations. However, the greatest source of progress in automated theorem proving has been the development of better search heuristics. As an example, our recent paper [18] shows a 60% improvement in the number of proofs found (over the TPTP library [19]) for current clause selection heuristics compared to the simple heuristics that were state-of-the-art 30 years ago.

Most of the leading modern theorem provers are based on variants of the superposition calculus with optional negative literal selection [2]. This includes e.g. SPASS [21], Vampire [11, 7], Prover9 [9] and E [14, 15]. These provers are based on saturation, using variants of the given-clause algorithm. The proof state is represented as a set of first-order clauses, and a proof by contradiction is attempted by systematically deriving and adding new clauses to this state. Redundancy elimination (in particular rewriting and subsumption) is used to reduce the search state and thus the search space. The main heuristic choice points are selection of a good term ordering for a given proof task, selection of inference literals, or of a selection strategy, and selection of the *given clause*, i.e. the clause that is one of the premises of all generating inferences in a given iteration of the main loop.

Heuristics for these choice points have historically been developed based on developer experience and hunches, backed by, often extensive, experimental evaluation. However, increasingly automated or semi-automated mechanisms come into play. Clause selection heuristics for E have been created using hill-climbing methods [20] and genetic algorithms [12]. Vampire is using look-ahead to find good literal selection schemes [6]. However, while there is significant progress, results are still unsatisfactory. In typical cases, only a very small percentage of derived clauses contribute to the actual proof.

2 Challenge: Using induction to guide deduction

We live in one of two possible realities. In one of these realities, proof search in first-order logic is inherently unpredictable, even for classes of proof problems of particular interest to users. In the alternative reality, it is possible, at least in theory, to find strategies that are more likely than others to succeed on defined classes of problems.

There are several reasons to believe in the second case. The success of manual and semi-automatic tuning hints that proof search is not entirely unpredictable. There is some existing work to obtain search control knowledge automatically from examples of successful proofs, using patterns [3, 16, 4, 13] or neural networks [17, 8], with moderate success.

However, I believe that we have now reached a level of robustness and efficiency of software and capability of available hardware that such attempts should yield significantly greater rewards than in the past. On the deduction side, we can now efficiently extract proof objects and even largely complete proof search graphs from automated theorem provers [15]. On the machine learning side, there are now powerful standard algorithms and libraries [10, 1, 5] that can be used with relatively little effort. And on the hardware level, multi-gigabyte main memory has become routine even on personal workstations and laptops, and multi-core processors and modern graphics cards provide unprecedented amounts of computing power.

To make the most of these capabilities, we can be guided by a number of interesting research questions, including the following:

- How do we represent search control experiences? What are relevant features of the proof state, of formulas, clauses, terms, and inferences?

- What automated learning methods are likely to be useful? Deep Learning is attractive because it has already tackled many problems that have been hard to formalize. However, it requires significant hardware, and even then is quite slow. Also, it is non-trivial to understand and communicate knowledge from such highly distributed representations.

- Can we represent clauses, formula, proof states, proof problems by relatively small sets of simple features while maintaining the information necessary for proof guidance? If yes, how?

- Can we automatically cluster proof problems in a way that problems in a given cluster have similar search properties? If yes, can we characterize these clusters in a way that allows us to assign problems to them a-priori?

- Can we dynamically detect if any given proof attempt is unlikely to succeed and switch to a different strategy? Can we extract useful intermediate results from failed proof attempts and reuse them, maybe in a way similar to the the mechanisms used by some propositional CDCL solvers to keep learned lemmata during a restart?

3 Conclusion

Anyone who has seriously studied mathematics will have made the experience that there are at least two levels of learning to reason in a given domain. On the one hand, one can consciously learn and understand the definitions, theorems, and proofs. But, as is obvious from observing first-year students during open-book exams, this knowledge is not sufficient to effectively perform mathematics. There is a second, mostly subconscious level of learning to perform the right steps at the right place during reasoning, of using the right piece of knowledge in a constructive way.

I believe that to achieve human-level performance on hard problems, theorem provers likewise must be equipped with *soft* knowledge, in particular soft knowledge automatically gained from previous proof experiences. I also suspect that this will be one of the most fruitful areas of research in automated theorem proving. And one of the hardest.

References

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Chen Zhifeng, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin Sanjay Ghemawat, Andrew Harp Ian Goodfellow, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Manjunath Kudlur Lukasz Kaiser, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker Kunal Talwar, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, , and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Leo Bachmair and Harald Ganzinger. Rewrite-Based Equational Theorem Proving with Selection and Simplification. *Journal of Logic and Computation*, 3(4):217–247, 1994.

[3] Jörg Denzinger and Stephan Schulz. Learning Domain Knowledge to Improve Theorem Proving. In M.A. McRobbie and J.K. Slaney, editors, *Proc. of the 13th CADE, New Brunswick*, volume 1104 of *LNAI*, pages 62–76. Springer, 1996.

[4] Jörg Denzinger and Stephan Schulz. Automatic Acquisition of Search Control Knowledge from Multiple Proof Attempts. *Journal of Information and Computation*, 162:59–79, 2000.

[5] Fèlix-Antoine Fortin, François-Michel De Rainville, Marc-Andrè Gardner, Marc Parizeau, and Christian Gagnè. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[6] Kryštof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the selection. In Nicola Olivetti and Ashish Tiwari, editors, *Proc. of the 8th IJCAR, Coimbra*, volume 9706 of *LNAI*, pages 313–329. Springer, 2016.

[7] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Proc. of the 25th CAV*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

[8] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep Network Guided Proof Search. *arXiv preprint arXiv:1701.06972*, 2017. (also to appear in Proc. 21st LPAR, 2017).

[9] William W. McCune. Prover9 and Mace4. http://www.cs.unm.edu/~mccune/prover9/, 2005–2010. (acccessed 2016-03-29).

[10] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[11] Alexandre Riazanov and Andrei Voronkov. The Design and Implementation of VAMPIRE. *Journal of AI Communications*, 15(2/3):91–110, 2002.

[12] Simon Schäfer and Stephan Schulz. Breeding theorem proving heuristics with genetic algorithms. In Georg Gottlob, Geoff Sutcliffe, and Andrei Voronkov, editors, *Proc. of the Global Conference on Artificial Intelligence, Tibilisi, Georgia*, volume 36 of *EPiC*, pages 263–274. EasyChair, 2015.

[13] Stephan Schulz. Learning Search Control Knowledge for Equational Theorem Proving. In F. Baader, G. Brewka, and T. Eiter, editors, *Proc. of the Joint German/Austrian Conference on Artificial Intelligence (KI-2001)*, volume 2174 of *LNAI*, pages 320–334. Springer, 2001.

[14] Stephan Schulz. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.

[15] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th LPAR, Stellenbosch*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.

[16] Stephan Schulz and Felix Brandt. Using Term Space Maps to Capture Search Control Knowledge in Equational Theorem Proving. In A. N. Kumar and I. Russell, editors, *Proc. of the 12th FLAIRS, Orlando*, pages 244–248. AAAI Press, 1999.

[17] Stephan Schulz, Andreas Küchler, and Christoph Goller. Some Experiments on the Applicability of Folding Architecture Networks to Guide Theorem Proving. In D.D. Dankel II, editor, *Proc. of the 10th FLAIRS, Daytona Beach*, pages 377–381. Florida AI Research Society, 1997.

[18] Stephan Schulz and Martin Möhrmann. Performance of clause selection heuristics for saturation-based theorem proving. In Nicola Olivetti and Ashish Tiwari, editors, *Proc. of the 8th IJCAR, Coimbra*, volume 9706 of *LNAI*, pages 330–345. Springer, 2016.

[19] Geoff Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[20] Josef Urban. Blistr: The blind strategymaker. In Georg Gottlob, Geoff Sutcliffe, and Andrei Voronkov, editors, *Proc. of the Global Conference on Artificial Intelligence, Tibilisi, Georgia*, volume 36 of *EPiC*, pages 312–319. EasyChair, 2015.

[21] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. SPASS Version 3.5. In Renate Schmidt, editor, *Proc. of the 22nd CADE, Montreal, Canada*, volume 5663 of *LNAI*, pages 140–145. Springer, 2009.