Hint Selection and Prioritization

Robert Veroff[1][*], Josef Urban[2][*], and Michael Kinyon[3][†]

[1] University of New Mexico, Albuquerque, New Mexico, U.S.A.
[2] Czech Technical University, Prague, Czech Republic
[3] University of Denver, Denver, Colorado, U.S.A.

Abstract

The method of proof sketches has been an effective strategy for proving challenging theorems—including open questions—with theorem provers such as Prover9[6]. In studies with multiple conjectures, however, a very large number of hint clauses can cause significant distractions, resulting in a failed search. We present refinements to the original proof sketches method[10], including new methods for selecting and prioritizing hints to help minimize these distractions.

1 Introduction

Our search for proofs of difficult theorems with the automated theorem prover Prover9 [6] generally involves sequences of runs that rely heavily on the use of hints [9] and on the method of proof sketches [10]. Under the hints strategy, a generated clause that *matches* (subsumes) a user-supplied hint clause is given a high priority in the proof search. A *proof sketch* for a theorem T is a sequence of clauses *sufficient* to prove T. Proof sketches are a natural and potentially effective source of hints in a search for a proof.

In [10], we consider how the generation and use of proof sketches, together with the sophisticated strategies and procedures supported by an automated reasoning program such as Prover9, can be used to find proofs of challenging theorems, including open questions. The general approach is to find proofs with additional assumptions and then to systematically eliminate these assumptions from the input set, using all previous proofs as hints. It also can be effective to include as proof sketches proofs of related theorems in the same area of study.

Much of our work is in areas of mathematics that can be organized into *theory hierarchies* defined by a base theory and the repeated addition of independent axioms. Theory hierarchies provide a natural source for extra assumptions that have been especially effective in our work. For a specific example, a lattice theory hierarchy is described in [7]. It begins with axioms for lattice theory, is extended with properties such as compatibility ($(x \vee y)' = x' \wedge y'$) and modularity ($x \vee (y \wedge (x \vee z)) = (x \vee y) \wedge (x \vee z)$) and ends with axioms for Boolean algebra.

2 Hint Selection and Prioritization

In a study involving several related theorems the number of accumulated hints can get very large. Having too many hints that are provable but not useful for finding a proof of the current theorem can cause a significant distraction to a search. In this case, careful management of hints becomes especially important.

[*]Partially supported by the *AI4REASON* ERC Consolidator grant 649043, the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/ 0000466 and the European Regional Development Fund.

[†]Partially supported by Simons Foundation Collaboration Grant 359872.

One way to deal with the problem of too many hints is to limit the number of proofs that contribute hints, for example, by tightening the definition of “related” theorem. Another way that has been especially effective is to prioritize hints and to have the search prefer hint matchers of higher priority hints. The intention is that hints that are more likely to be helpful in the current search are given a higher priority.

Our methods for prioritizing hints include ordering proofs, ordering clauses within a proof, and ordering clauses between proofs.

Ordering proofs: Prefer hints from most recent proofs. For example, in a sequence of proofs resulting from the systematic elimination of extra assumptions, the most recent proofs are likely to be the most relevant to a sought after proof.

Ordering clauses within a proof: Prefer hints with higher clause numbers. The reasoning is that the higher numbered clauses in a proof are closer to completing a derivation of the target clause (generally the empty clause). This is especially significant when we are trying to eliminate an extra assumption from a previous proof (as opposed to proving a different goal).

A variation of this method prefers hints that are closer to the target clause by inference distance. Hints from several proof sketches can easily be combined by merging the clauses by their inference distances from the empty clause in their respective proofs. That is, all of the clauses that are n steps away in their respective proofs are preferred over any proof clauses that are $m > n$ steps away.

Ordering clauses between proofs: Prefer proof clauses that appear in more proofs in the current study. Furthermore, we can use this frequency count criterion to limit the number of hints by including only hints that appear in some minimum number of proofs.

These methods and examples of their application will be presented in some detail.

3 Some Results

Support for hint prioritization has been added to Prover9, and the new methods have been applied to numerous problems. For example, much of our most recent work has been on a large, long-term project in loop theory [5]. The *AIM* project (for Abelian inner mapping groups) includes numerous open questions of mathematical interest. These are difficult problems; many having proofs that are several tens of thousands of steps long. It has become increasingly difficult to make progress in this study.

The new methods have led to new results and a growing library of relevant hints. For example, before the inclusion of hint prioritization into Prover9, we had 549 proofs of “mathematically interesting” properties in 117 output files.[1] Together the files contribute 167K distinct hint clauses for later runs (47K when we eliminate those that appear in only 1 output file). As of November 2018, we have 641 proofs in 149 files contributing 2.3 million distinct hints (90K when we eliminate the hints that appear in at most 2 output files). This continues to be a work in progress.

4 Related and Future Work

ENIGMA [3, 4] is a learning-based method used to influence given clause selection in E Prover [8]. This works by (i) learning to classify given clauses from previous successful searches as “useful” or “un-useful”, and (ii) applying the classifier in a new search to the generated

[1]Many of the output files contain proofs of multiple goals.

clauses. Analogously, an ENIGMA classifier could be applied to candidate hint clauses before a new search is initiated.

ProofWatch [1] and ENIGMAWatch [2] are learning-based methods for dynamically modifying the given selection priorities of hint-matching clauses during an E Prover search. In particular, the progress made toward completing each of several supplied proofs is monitored and used to focus on the more completed proofs in ProofWatch. In ENIGMAWatch, the proof-completion vectors from ProofWatch are additionally used as features characterizing the proof state and added to the features used for training ENIGMA classifiers. We have started experiments with replaying the Prover9 hint-guided proofs in E equipped with these methods.

References

[1] Z. Goertzel, J. Jakubuv, S. Schulz, and J. Urban. ProofWatch: Watchlist guidance for large theories in E. In J. Avigad and A. Mahboubi, editors, *Interactive Theorem Proving - 9th International Conference, ITP 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10895 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2018.

[2] Z. Goertzel, J. Jakubuv, and J. Urban. ProofWatch Meets ENIGMA: First Experiments. G. Barthe, K. Korovin, S. Schulz, M. Suda, G. Sutcliffe, M. Veanes, editors, *LPAR-22 Workshop and Short Paper Proceedings*, volume 9 of *Kalpa Publications in Computing*, pages 15–22. EasyChair, 2018.

[3] J. Jakubuv and J. Urban. ENIGMA: efficient learning-based inference guiding machine. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[4] J. Jakubuv and J. Urban. Enhancing ENIGMA given clause guidance. In F. Rabe, W. M. Farmer, G. O. Passmore, and A. Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

[5] M. Kinyon, R. Veroff and P. Vojtěchovský. Loops with abelian inner mapping groups: an application of automated deduction. In M. P. Bonacina and M. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune, volume 7788 of Lecture Notes in Artificial Intelligence*, pages 151–164. Springer, 2013.

[6] W. McCune. Prover9, version 2009-11A. `www.cs.unm.edu/~mccune/prover9/`

[7] M. Rose and K. Wilkinson. Application of model search to lattice theory. Tech. Report ANL/MCS-P900-0801, Argonne National Laboratory, Argonne, IL, 2001.

[8] S. Schulz. System description: E 1.8. In K. L. McMillan, A. Middeldorp, and A. Voronkov, editors, *LPAR*, volume 8312 of *Lecture Notes in Computer Science*, pages 735–743. Springer, 2013.

[9] R. Veroff. Using hints to increase the effectiveness of an automated reasoning program: case studies. J. Automated Reasoning **16**:223–239, 1996.

[10] R. Veroff. Solving open questions and other challenge problems using proof sketches. J. Automated Reasoning **27**:157–174, 2001.