Boosting Automated Reasoning By Mining Existing Proofs

Thomas Gransden

Department of Computer Science,
University of Leicester
tg75@student.le.ac.uk

Abstract: Theorem provers are extremely useful tools for reasoning about complex mathematics and large scale systems. They play an important part in verifying the correctness of designs, particularly for safety-critical systems. The problem is that they still require a large amount of human intervention to successfully guide the proof. Our solution to this important limitation involves using data mining techniques to mine proof tactics from successful and unsuccessful proofs with the aim that they can be more widely applied. In this paper we motivate the problem, and describe our work in progress.

1 Introduction

Proof automation is an extremely desirable property in theorem proving. In First Order Theorem Proving, reasoning automatically has now become a reality with the advent of fully automated theorem provers that work with decidable fragments of First Order Logic. Automated Theorem Proving (ATP) is a very competitive area of research, with many automated proof systems now competing against each other in tournaments staged at international conferences. This shows the progress that has been made so far in the field of Automated Reasoning, but we also know that there are many more challenges ahead.

Although First Order Logic provides us with the potential for automation, we must accept a tradeoff with the amount of expressivity that the logic provides. A more capable logic for expressing complex problems is Higher Order Logic (HOL). HOL allows us to quantify over functions and predicates, whereas this cannot be done in FOL. This increase in expressivity leads to a major problem; HOL is much harder to automatically reason over due to undecidable algorithms and proof methods. Whilst trivial goals can be solved automatically, most complex proof developments require significant amounts of human intervention to elicit a successful proof.

As the field develops there is more formalised knowledge becoming available in the form of proof libraries. Whilst there is too much information to perform a brute force proof search, there is the potential for identifying interesting and useful patterns in these libraries. Our work is based on making use of these patterns by turning them into tactics which could be applied automatically. We can think of a tactic as a set of instructions that when applied to a proof state, they either advance the proof state, complete the proof or fail to do either of these.

2 Data Mining in Theorem Proving

There have been some attempts at applying data mining techniques in the context of theorem proving. Many first order theorem provers use heuristics to guide their proof search procedures. Much work has been carried out on improving these heuristics by learning from previous proof examples [1, 2, 3].

In terms of applying data mining techniques to higher order theorem proving, there have been a few attempts. Hazel Duncan's work [4] looked at searching through libraries of successful proofs to find commonly occurring sequences of proof steps. These proof steps were then formed into tactics, which were then incorporated into a small automatic theorem prover in Isabelle. The prover was naive in the sense that it simply tried each tactic that was found in a brute force way.

The tactics that Duncan produced were moderately effective in terms of their applicability. From the test set that she evaluated her tactics against, an average of 32% of them could have at least one of her tactics applied to them. A tactic that was deemed good in terms of applicability was used in 51% of the test set proofs, whereas a bad tactic couldn't be applied at all. A recommender system [6] was created that made use of the sequences of mined proof steps, providing hints to the user about what the next step of a proof could be, based on the existing proofs that were mined from the proof libraries.

A more recent attempt is called ML4PG [5]. This project attempts to show that it is possible to link Interactive Theorem Provers with machine learning tools, and that non-trivial patterns can be found. The software provides a link between the Coq theorem prover and the machine learning tools Matlab and Weka. The user can use ML4PG when they become stuck during a proof, and the tool provides hints to the user. The user must select libraries of existing proofs, and the level of proof that they would like to look for patterns in (proof tree, goal structures or proof steps).

The learning tool returns clusters of lemmas that show similarities to the difficult proof, and it is left to the user to inspect the proofs of the clustered lemma to see if there is a proof pattern that could help advance the problematic proof. In their conclusions, it would appear that clustering based on goal structures provides the best results. They also show that their tool can be used in a variety of proof situations such as industrial proofs and mathematical proofs. Overall, this work goes some way to showing that there is a place for machine learning in theorem proving, and that

useful results can be found.

3 Research Challenges

When combining data mining techniques with interactive theorem proving, there are naturally many challenges that arise. Whilst some of these are addressed more generally in [5] there are some that arise specifically from our work on implementing a tactic miner.

In Higher Order Logic, a particular goal could be solved by an unlimited combination of proof steps, which can include user specified variable instantiations. This makes it difficult infer useful models from specific examples in such a way that the distinguishing features of a proof can be extracted and applied more generally.

Providing a suitable interface for communication between the theorem prover and the mining algorithm is difficult. Ultimatley, the results of the tactic miner must be interpreted back into the language of the theorem prover being used, so we must consider how this can be done in a way that allows the theorem prover to make use of the tactics output from the mining algorithm. We must make use of mechanisms provided in some theorem provers that allow users to specify tactics in some language that the theorem prover can understand. An example of this is the Ltac language that Coq uses.

So far, proof tactic inference approaches have been unsupervised; tactics have only been observed from successful proofs. However, the theorem proving process also generates a lot of failed and discarded derivations. The ability to leverage these negative examples can significantly improve the accuracy and efficiency of mining algorithms. Therefore, one of our research challenges will be to develop a supervised tactic mining algorithm that will make use of this data.

Finally, when evaluating tactic mining algorithms there are a number of factors to consider. We must see if the algorithm is scalable, and also generic enough to be implemented within a number of theorem provers. We must also look at the quality of the algorithm and see how many examples were needed to learn tactics from, and we must also evaluate the applicability of the tactics that the algorithm outputs.

4 Current Work

Although we are focussed on developing the tactic mining approach at the moment, we must also consider the behaviour of a theorem proving system that implements our tactic mining algorithm. This behaviour is shown in Figure 1.

Firstly, we must take the proof libraries (of successful and unsuccessful proofs) and abstract them into a format that is appropriate for learning. Currently, we have converted successful Isabelle proofs into simple lists consisting of the proof steps used. However, we know that including some contextual information would be beneficial,

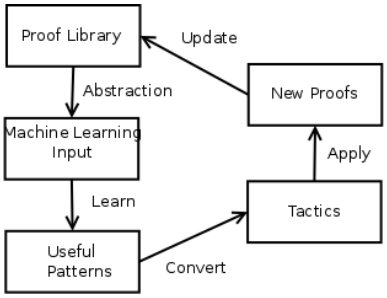

Figure 1: Proposed Technique

so that we can identify the state of the proof when a particular proof step was applied.

The next step will be to learn useful patterns from these existing proofs. We are currently at the stage of studying what kind of learning approach will give the best results. We envisage that something novel will have to be developed specifically for solving our problem.

Once we have identified the useful patterns in the proof libraries, we can either present them to the user and allow them to manually evaluate the sequences, or ideally we want to form tactics from them. By doing this we open up the possibility of automatically applying them to proof states.

Finally, when we complete any new proofs the proof library must be updated with the new information. This is important so that any future times that the algorithm is used, it can make use of the newly proved knowledge.

References

[1] James P. Bridge. Machine learning and automated theorem proving. Technical report, University of Cambridge, Computer Laboratory, November 2010.

[2] J. Denzinger, M. Fuchs, C. Goller, and S. Schulz. Learning from Previous Proof Experience. Technical report, Institut für Informatik, Technische Universität München, 1999.

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

[4] H. Duncan. *The Use of Data Mining for the Automatic Formation of Tactics*. PhD thesis, University of Edinburgh, 2007.

[5] Katya Komendantskaya and Jonathan Heras. Machine Learning in Proof General: Interfacing Interfaces. December 2012.

[6] Alison Mercer. PG Tips: A Recommender System for an Interactive Theorem Prover. Master's thesis, University of Edinburgh, 2006.