A statistical relational learning challenge – extracting proof strategies from exemplar proofs

Gudmund Grov GGROV@STAFFMAIL.ED.AC.UK
University of Edinburgh, United Kingdom

Ekaterina Komendantskaya KATYA@COMPUTING.DUNDEE.AC.UK
University of Dundee, United Kingdom

Alan Bundy BUNDY@STAFFMAIL.ED.AC.UK
University of Edinburgh, United Kingdom

Abstract

Proof automation is becoming a bottleneck in both mechanised mathematics and particularly industrial use of formal development methods. In this challenge paper, we argue that statistical relational learning can be used to discover and extract proof strategies, which can aid with this automation. We outline what we believe are key problems and possible approaches to solve these questions.

1. Introduction

Modern theorem provers are used both to ensure that a proof is correct (so called proof-checking), and to perform the actual proof search. *Interactive theorem provers* (ITPs), are the provers that require user interaction to guide the proof. Examples of interactive theorem provers are Coq, Isabelle, HOL and HOL light. In such provers, the user typically works backwards from a goal towards the hypothesis or previously proven lemmas. For large systems, thousands of lemmas may be available in a well-developed theory, and hundreds of hypothesis may be present in a given proof state – with most of them having no significance for the particular proof (although it is hard to find out which are the important ones). A proof in such system is a finite combination of *tactics*, which can be seen as a function from a goal to a set of subgoals that has to be proven to justify the overall goal, thus forming a tree where the nodes are tactics and edges are the goal states. All such steps are verified using the underlying type theory.

Recently, automation within ITPs has greatly increased mainly due to integration of (automatic) resolution provers and SMT solvers. However, user interaction is still required to verify significant mathematical results or verification of industrial sized software and hardware systems. For industrial application of formal methods beyond niche markets, the cost of the resulting interactive proofs has become a bottleneck.

Many proofs in ITPs follow a similar pattern — and once the user has managed to discharge one proof, many proofs follow (almost!) the same combination of tactics. Automated discovery of such common proof patterns by utilising machine learning tools could potentially provide the much sought automatisation for statistically similar proof-steps; as was argued e.g. in (Denzinger et al., 1999; Denzinger & Schulz, 2000; Duncan, 2002; Lloyd, 2003; Tsivtsivadze et al., 2011; Urban et al., 2008). With this in mind our overall research hypothesis is that

> *it is possible to detect proof patterns where we can extract proof strategies which can automate the proofs of 'similar conjectures'.*

As was clarified in (Denzinger et al., 1999), applications of machine-learning assistants to mechanised proofs can be divided into *symbolic* (akin e.g. Inductive logic programming), *numeric* (akin *neural networks* or *kernels*), and *hybrid* (Lloyd, 2003; d'Avila Garcez et al., 2002; Getoor & Taskar, 2007), which combine symbolic and numeric methods. The advantages of the numeric methods over symbolic is tolerance to noise and uncertainty, as well as availability of powerful learning functions. For example, the standard multi-layer perceptrons with error

Appearing in *Workshop on Statistical Relational Learning 2012*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

back-propagation algorithm are capable of approximating any function from finite-dimensional vector spaces with arbitrary precision. In this case, it is not the power of the learning paradigm, but the feature selection and representation method that sets the limits. Symbolic methods, on the other hand, may be tuned better to account for the syntactic aspects of the theorem provers. It may also be more suitable to handle the relational nature of the properties, as well as better guiding the necessary generalisations.

We do not believe that this domain is suitable for the application of numeric or symbolic methods alone:

- the proofs contains a lot of noise and non-patterns (as well as different patterns), which is not handled well by symbolic machine learning techniques;
- the tactic combinations in a sequence are not identical within a pattern. We cannot thus simply reuse the tactic sequence, but we will need a generalisation of them, where symbolic methods have their strengths.

We can thus restate our research hypothesis as:

> *by utilising numeric machine learning techniques to detect proof patterns and generalise them into proof strategies by symbolic machine learning methods it is possible to automate the proofs of 'similar conjectures'.*

Our own experience and related work mentioned above make us classify various challenges in practical machine-learning of proof strategies into two main classes.

1. The first set of challenges comes from the complexity of the notion of a proof in higher-order interactive theorem provers: there are different levels at which one approaches the proof, which results in a hierarchy of proof-patterns; as we consider in §3.
2. The second set of challenges comes from the fact that relational and symbolic learning has to be intertwined, in order to balance statistical and logical patterns arising in mechanised proofs; §4 discusses that.

2. Proofs and proof strategies

We start with an illustration of what we mean by proofs and proof strategy, and give some examples using an Isabelle syntax. The respective proof trees are shown graphically in Figure 1[1]. We first develop our proofs in the manner that visually is similar to what a programmer sees when working with an interactive theorem prover; we also call it the level of tactics.

lemma lem1: $A \to A$
apply (rule impI)
apply assumption
done

Here, the first step in the proof reduces the goal to $A \vdash A$ which can be proved by the assumption. In

lemma lem2: $A \to (B \to A)$
apply (rule impI)
apply (rule impI)
apply assumption
done

this step has to be done twice, first to get the goal $A \vdash B \to A$, then $A, B \vdash A$ which can be discharged by an assumption. Finally, in

lemma lem3: $(B \wedge A) \to (A \wedge B)$
apply (rule impI)
apply (rule conjI)
apply (erule conjE)
apply assumption
apply (erule conjE)
apply assumption
done

the first step gives $B \wedge A \vdash A \wedge B$ and the second step gives two subgoals: $B \wedge A \vdash A$ and $B \wedge A \vdash B$, which both can be solved by removing $\wedge$ in the assumption, producing the goals $B, A \vdash A$ and $B, A \vdash B$. Both of them can be proven by an assumption.

We will now focus on one of the patterns in this proof, and that is the underlined steps above. These are actually examples of a well-known tactic often called *intros* which removes certain symbols of the goal[2]. What we are hoping for is a system which not only recognises that this is a pattern, even if the use of it is not identical in any of the proofs, but it also realises in which cases it can be applied and what the result of applying it is — i.e. a *strategy* looking something like:

```
PRE top_level_symbol ∈ {→, ∧}
  WHILE top_level_symbol ∈ {→, ∧}
    (apply (rule impI)) OR (apply (rule conjI))
```

[1]The examples are deliberately kept simple (propositional) to ease reading, and thus do not illustrate the issue with expressiveness and size.

[2]Dale Miller calls the result of applying intros 'normal proofs'.

POST top_level_symbol $\notin \{\rightarrow, \wedge\}$

We believe such generalisation, and the encoding of (features of) the expected input and output goals are the key to success for our ambitious research goal. Without encoding pre- and post-conditions we will not be able to know when to apply a learned tactic and when to stop a repeated application of it.

The examples we have considered so far hide some information that drives the proof engine: for example, the types and shapes of subgoals to which the tactics are applied, cf. Figure 1. Finally, the overall proof-structure, also called a *proof-tree*, is often hidden from the prover's interface, we illustrate it in Figure 1.

3. Hierarchical proof-pattern recognition

The learning of structured data is recognised as one of the paramount challenges in machine learning, and many sophisticated techniques have been suggested to address it (Bakir et al., 2007). However, in the case of higher-order proofs, we deal with a hierarchy of structures and patterns. Depending on this hierarchy, it is likely that different methods from the arsenal of (Bakir et al., 2007) will be useful at different stages.

Consider the examples of §2. One proof has the following layers of structure:

$\star$ The level of goal structures (i.e. the edges of Figure 1). For example, for every given sequence of subgoals, there is an apparent pattern in the structure of the formulas, as one proof-step changes another. This type of feature abstraction has been used for learning the inputs for automatic provers, e.g. (Urban et al., 2008; Denzinger & Schulz, 2000) – which has later been extended to interactive proofs (Tsivtsivadze et al., 2011); we will also call this kind of proof-pattern recognition the *term-pattern recognition.* As the related work shows, *geometric kernels* and similar methods are appropriate for investigating term-structures in the proofs. However, the overall proof structure and information of how the goal is produced are lost. Additional questions here are:

- Should we look at the goal in isolation or how they are changed by a tactic? E.g. a feature could be the size of a term, but it could also be a parameter indicating whether or not the term-size is reduced in a proof step.
- Should we look at the goal or include the hypothesis (and other background information) in the feature space? The examples we have shown are very small, and had at the most 2 assumptions – a more realistic example could have hundreds of assumptions, with some being just noise. Other background information, like properties we have proved about certain operators, are also important.
- Object- or meta-level properties? Object-level could refer to properties such as x has the value 5, while meta-level properties could e.g. relate the goal to an assumption or other known facts. For example, the *assumption* tactic requires an identical[3] term to the goal is in one of the hypothesis.

$\star\star$ The level of tactics (i.e. the nodes in Figure 1, usually given by proof script text directly). Sequences of tactics applied at every level of the proof bear some apparent patterns, as well. There is always a finite number of tactics for any given proof, and therefore, they may serve well as features for statistical learning. Previous work on learning proof strategies (Jamnik et al., 2002; Duncan, 2002) has taken this approach; we also call it *tactic-pattern recognition.* It is important to note that there may be proofs in which the goal structures do not bear any evident pattern; however, the sequence of applied tactics does. Also, as an additional complication, there is normally a great variety of tactic combinations that lead to a successful automated proof for one goal; and *vice versa*, different goals may have the same sequences of tactics in successful proofs. Moreover, tactics often have complex configurations, which can both be hidden or given as arguments (e.g. rules to apply or instantiations of variables). For tactic-pattern recognition, *variable-length hidden Markov models* (Duncan, 2002), rather than kernels seems more promising – in particular, recent tree-versions of variable-length hidden Markov models which we understand have been developed in the natural language learning domain. The down-side of such tactic-pattern recognition is that any knowledge of when and why a tactic is applied, as well as its result is lost (except with respect to other tactic applications).

$\star\star\star$ Finally, there is a level of a proof tree (i.e. the complete trees of Figure 1) – that shows relations between different proof branches and subgoals and gives a better view of the overall proof flow; this approach was tested in (Komendantskaya et al., 2012). This set of experiments was done using multi-layer neural networks and kernels; and we call it *proof-tree pattern recognition.* As opposed to tactic chain discovery, Markov chains do not seem convenient for this pur-

[3]Unifiable to be precise.

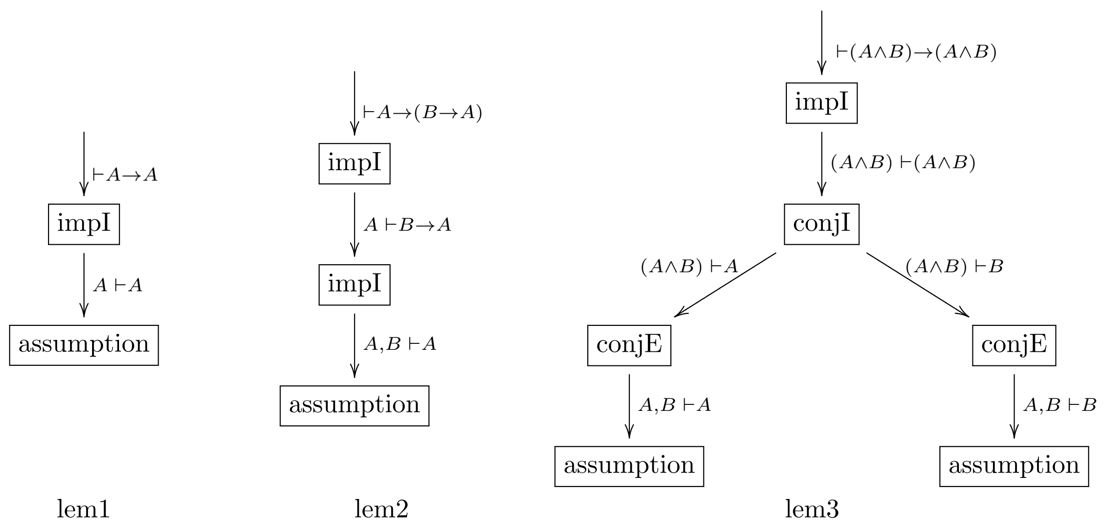

Figure 1. Proof trees

pose. Again, it may happen that, for a set of computer proofs, there is no apparent pattern in either subgoal structure or the tactic sequence, but the proof-tree structure is common.

It is still an open research question to find the best approach to grasp these rich hierarchies: is it best to keep the learning process open, and look for patterns at all levels? or is it best to data-mine patterns that span all three layers? And what impact such decisions may have on the tasks that machine-learning can perform within the interactive proof-search? It is also important to note that one would expect a pattern to be a sub-sequence of tactic applications, or a sub-tree of a proof trees.

The research questions, from a machine learning perspective, are:

- Is it fruitful to data-mine every structural level independently, thus not making premature assumptions about their connections, and allow the statistical methods to re-discover such connections when they exist?

- Is it best to apply different statistical methods (Bakir et al., 2007) at every level, or is it best to use one (perhaps hybrid) method on all stages?

- Can we independently apply symbolic methods and combine the results? For example, HR (Colton, 2002) is a tool which has been successfully used to discover new concepts; ILP have been successful in generalising relational data; while (Krumnack et al., 2007) can handle higher-order data, which is required in order to generalise terms found in most interactive provers. Can we then combine the results of these different symbolic techniques?

- Can we borrow any machine-learning methods applied to different domains that require not only the mix of statistical and relational models, but also hierarchical models of learning (as e.g. in the natural language domain) ?

4. Statistical and relational proof-pattern recognition: the balance

The second challenge in our current work is balancing the statistical and relational components. Assuming the right machine-learning tools (e.g. from (Bakir et al., 2007)) are used and tuned for pattern-recognition at every level of the proof hierarchy, and with very good statistical performance, cf. e.g. (Komendantskaya et al., 2012), the following essential tasks may not be achieved without interference of the relational methods:

- Extraction of common features from a rich variety of proofs. Our initial experiments show that unlike in case of e.g. face recognition, simple solutions for automated feature extraction will not apply. There is too much of variety in the length and complexity of automated proofs; and tracking the features literally seen in the proof trees will extend the number of features, and hence the dimensions of the feature space to unreasonable sizes, cf. (Komendantskaya et al., 2012). Therefore, some initial pre-processing of the automated proofs will be needed, so that extracted features are no longer literal samples from the proofs, but are proof-features in a more abstract sense. This initial abstraction can only be achieved by symbolic methods; cf. Figure 2.

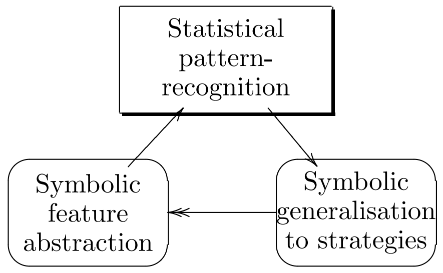
Statistical pattern-recognition
Symbolic feature abstraction
Symbolic generalisation to strategies

Figure 2. Balance of symbolic and relational methods in proof pattern recognition

- Generalisation of common proof trees to proof strategies. Suppose a proof-family is statistically discovered, cf. (Komendantskaya et al., 2012). In many cases, and unlike e.g. image recognition, the bare fact that a proof belongs to some family or cluster is not immediately and necessarily a practical knowledge. The necessary step is to ask: Which properties of this proof family lead to its success or failure, in other words, is there a general strategy that can be used for future proof discovery? Note also that this general strategy may concern one or more hierarchical levels, depending on the proof-pattern discovered statistically.

We illustrate the generalisation process as follows.

Both *rule impI* and *rule conjI* from §2 perform a new goal introduction step and we need to generalise this into a new strategy *intro*, which can be seen as introducing an OR branch in the search (either apply *rule impI* or apply *rule conjI*). As a result of such generalisation all underlined steps in the examples can be seen as an application of *intro*. However, as discussed in §2, the conditions in which to apply it, as well as the results of applying it, has to be found – and can only be achieved by symbolic methods.

Moreover, suppose the generalisation above is done and applied to the examples. Now it comes the time to iterate again the process outlined in Figure 2, that is, the features of the new generalised proofs need to be abstracted, then patterns need to be statistically discovered, after which another round of generalisation can take place. For example, after the first learning round, the first steps become *intro*, *intro;intro* and *intro;intro*, respectively. Here, we see that *intro* is applied many times, and thus a generalisation would be a repetition *intro**, with a corresponding termination condition.

5. Conclusions

We have considered the three modes of proof-pattern learning, which are *term-pattern*, *tactic-pattern*, and *proof-tree pattern* recognition. We identified the set of challenges in each, as well as the challenge of combining them efficiently. Finally, we considered the challenge of balancing the symbolic (relational) and statistical tools in proof strategy learning. Our future work is to experiment and implement the described tools.

Acknowledgments

We would like to thank the partners within the AI4FM project and members of the MRG group in Edinburgh for valuable discussion. The work has been funded by EPSRC grants *AI4FM: using AI to aid automation of proof search in formal methods* (EP/H024204/1) and *machine learning coalgebraic automated proofs* (EP/J014222/1).

References

Bakir, G., Hofmann, T., Scholkopf, B., Smola, A., Taskar, B., and Vishwanathan, S.V.N. *Predicting Structured Data.* MIT, 2007.

Colton, S. *Automated Theory Formation in Pure Mathematics.* Springer, 2002.

d'Avila Garcez, Arthur, Broda, K. B., and Gabbay, D. M. *Neural-Symbolic Learning Systems: Foundations and Applications.* Springer-Verlag, 2002.

Denzinger, J., Fuchs, M., Goller, C., and Schulz, S. Learning from previous proof experience: A survey. Technical report, Fakultat fr Informatik, Technische Universitat Munich, 1999.

Denzinger, Jörg and Schulz, Stephan. Automatic acquisition of search control knowledge from multiple proof attempts. *Inf. Comput.*, 162(1-2):59–79, 2000.

Duncan, Hazel. *The use of Data-Mining for the Automatic Formation of Tactics.* PhD thesis, University of Edinburgh, 2002.

Getoor, L. and Taskar, B. *Introduction to Statistical Relational Learning.* MIT Press, 2007.

Jamnik, Mateja, Kerber, Manfred, and Benzmller, Christoph. Automatic learning of proof methods in proof planning. Technical report, L. J. of the IGPL, 2002.

Komendantskaya, E., Almaghairbe, R., and K.Lichota. ML-CAP: the manual, software and experimental data sets, 2012. http://www.computing.dundee.ac.uk/staff/katya/MLCAP-man/.

Krumnack, Ulf, Schwering, Angela, Gust, Helmar, and Kühnberger, Kai-Uwe. Restricted higher-order anti-unification for analogy making. In Orgun, Mehmet A. and Thornton, John (eds.), *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Artificial Intelligence*, pp. 273–282. Springer, 2007. ISBN 978-3-540-76926-2.

Lloyd, J.W. *Logic for Learning: Learning Comprehensible Theories from Structured Data.* Springer, Cognitive Technologies Series, 2003.

Tsivtsivadze, E., Urban, J., Geuvers, H., and Heskes, T. Semantic graph kernels for automated reasoning. In *Proc. 11th SIAM Int. Conf. on Data Mining*, pp. 795–803. SIAM / Omnipress, 2011.

Urban, Josef, Sutcliffe, Geoff, Pudlák, Petr, and Vyskocil, Jirí. Malarea sg1- machine learner for automated reasoning with semantic guidance. In *IJCAR*, LNCS, pp. 441–456. Springer, 2008.