Towards neuro-symbolic conjecturing

Sólrún Halla Einarsdóttir, Moa Johansson, and Nicholas Smallbone

Chalmers University of Technology, Gothenburg, Sweden.
{slrn, moa.johansson, nicsma}@chalmers.se

Abstract

Theory exploration systems automatically generate mathematical conjectures, by exploring a set of terms of interest. This search is expensive for large theories, as the set of terms becomes large. We describe ongoing work in combining data-driven and symbolic methods for automated conjecturing, where the data-driven part should identify which kinds of conjectures are likely to be useful and restrict the symbolic search to those ones. As a first step, we have extracted a dataset of lemma templates from Isabelle's Archive of Formal Proofs[1].

1 Introduction

Theory exploration is a symbolic technique for automated conjecturing based on testing [7]. It has been used to successfully discover, for example, lemmas needed in automated (co)-inductive provers [5, 1]. Our theory exploration system QuickSpec [7], takes as input a number of functions and datatypes, and builds terms of increasing size. The search space is managed by using already discovered properties to avoid larger terms that could be reduced or subsumed by something already known. While this works well for smaller signatures (up to around 10–20 functions) and terms up to about 10 symbols, it eventually runs into exponential blow up. To address this, we developed a variant of QuickSpec called RoughSpec [2], which restricts the search space to properties of specific shapes using *templates*. For example, the template $?F(?F(X,Y),Z) = ?F(X,?F(Y,Z))$ describes an associative binary function $?F$. Currently, the human user decides which templates to use.

We plan to instead select templates automatically using a data-driven approach. As a first step towards this goal, we have collected and started to analyze a large dataset of lemmas from Isabelle's Archive of Formal Proofs. The long term aim is to build a neuro-symbolic system for conjecturing, where given a theory, a machine learning system selects the most promising templates, and a symbolic system fills in the templates to produce conjectures, discarding any conjecture which is trivial, trivially false, or already known. Our hypothesis is that this approach combines the best of the machine learning and symbolic approaches: machine learning to learn which parts of the search space to focus on, and symbolic methods to reason about and evaluate specific conjectures.

2 A Library of Lemma Templates

In our previous work [2], our theory exploration system RoughSpec required the user to provide templates for the properties they were looking for. We also provided a small collection of "default" templates describing some properties we guessed might be useful for theory exploration based on our intuitions and experience. This collection included templates for properties such as commutativity and distributivity. In order to gain a more robust empirical understanding of

[1] https://www.isa-afp.org/index.html

what kinds of templates are useful and to provide a dataset for data-driven experiments we have mined equational lemma templates from the Archive of Formal Proofs(AFP). The AFP contains 676 entries from 425 authors, containing almost 200,000 lemmas and more than 3 million lines of code. The entries consist of proof formalizations from a variety of areas of Computer Science, Logic and Mathematics and we believe they are a good source of interesting and useful lemmas, as the lemmas we find there are handwritten as part of proofs that Isabelle users have seen a reason to formalize.

2.1 Preliminary results

We have collected and performed preliminary analysis on a dataset containing 22,767 equational lemmas, extracted from 2169 different theory files from 611 AFP entries. For each extracted lemma we generated a template representation of the lemma statement, showing the statement's term structure with function and variable names abstracted away but using integer labels to keep track of function symbols and variables that occur more than once. The dataset along with the code used to generate it is available at: `https://github.com/solrun/LibraryOfLemmas`.

These 22,767 lemmas are captured by 6567 different templates. In Figure 1 we can see that a small number of templates occur very frequently while the majority occur very seldom with 4099 templates occurring only once. The 10 most frequent templates together describe 3057 lemmas or 13.5% of the lemmas in our set, while more than 50% of the lemmas can be described using only 266 of the 6567 templates. This supports our hypothesis that only a smaller number of templates is needed to discover many lemmas using template-based conjecturing.

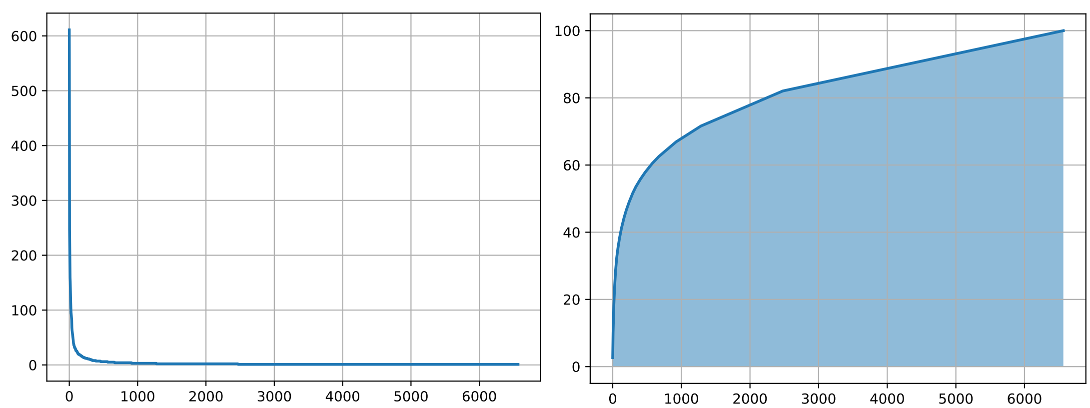

Figure 1: Left: Number of lemmas per template, sorted by frequency. Right: Cumulative percentage of lemmas in the dataset covered by most frequent templates.

Table 1 shows the top 10 most frequently occurring templates in our dataset, where # lemmas represents the number of lemmas matching the template, # thys is the number of different theory files it occurs in and # sessions is the number of different AFP sessions it occurs in. Template holes are represented by a question mark followed by a capitalized name, while variables are represented by a capitalized name. Among these common templates, we see a lot of similarity: many of the templates in Table 1 resemble each other, differing only in the number of variables or the order of application. Our hypothesis is that we can define a smaller set of "supertemplates" describing families of similar templates that can be generalized to describe the different family members, where using those templates and their generalizations we can

	Template	# lemmas	# thys	# sessions
1	?F (?G X Y) = ?H (?F X) (?F Y)	611	261	172
2	?F X = ?G (?H X)	566	265	169
3	X = ?F (?G X)	340	191	139
4	?F X = ?F (?G X)	280	149	118
5	X = ?F ?G X	247	136	98
6	?F (?G X Y) Z = ?H (?F X Z) (?F Y Z)	233	90	70
7	X = ?F X ?G	210	132	103
8	?F X (?G Y Z) = ?H (?F X Y) (?F X Z)	194	90	74
9	?F = ?G (?H X)	192	65	56
10	?F = ?G ?H X	184	110	85

Table 1: Top 10 most frequently occurring templates in the dataset.

generate a large proportion of the lemmas we need. For example (1) (6) and (8), (3) (5) and (7), and (9) and (10) should be grouped together and described by common "supertemplates". This would further reduce the space of templates to be searched over by RoughSpec.

In our previous work [2], we defined a set of 10 default templates capturing very common properties which we found useful in our case studies. Comparing these to the most frequent templates as shown above, we see that 4 out of our 10 default templates are also in the 10 most frequently occurring templates in our dataset. Of the remaining six default templates one did not show up at all, and the other occur in places 20–388. The second most commonly occurring template in the dataset, ?F X = ?G (?H X), is in a style we had previously disregarded as being too general to be suited to template-based theory exploration, but seeing how common this exact form of equivalence template seems to be we will definitely try out using it in future experiments. The differences between our collection of default templates and the most common templates in the dataset show the value of collecting a dataset for empirical evaluation.

Extending the dataset. We are currently expanding this dataset to also contain non-equational lemmas, such as conditionals, inequalities, and predicates. We also plan to extend the template language to cover e.g. lambda abstractions and quantifiers. With these extensions we should be able to cover all the lemmas in the AFP. Adding more data concerning for example the topic of the theory where the lemma in question is defined and used or the function definitions involved may also prove necessary in order to use this dataset to learn what templates are useful in various theorem proving contexts.

3 Future steps and related work

Having compiled a library of lemma templates, our next steps will be to analyze the data available there and apply it in the context of theory exploration and theorem proving. Our aim is to develop machine-learning based methods to make helpful template suggestions for a given set of functions. We envisage this implemented as a machine learning model trained to predict likely useful templates, given a suitable representation of a theory (i.e. a set of definitions of datatypes, functions and perhaps already known properties). These templates are then passed to a symbolic theory exploration systems (RoughSpec), which instantiates, tests and possibly proves properties before presented to the user. This has some similarities to neuro-symbolic program synthesis systems like DreamCoder [3], which use a neural network to predict a symbolic program, given a set of input-output examples. We could also use clustering methods to group together templates that are often seen together in the same theory formalization and then suggest templates based on lemmas that have already been defined by the user or found by exploration.

There have been recent attempts to use large language models for conjecturing tasks [8, 6]. A problem here is that the output typically contains a mixture of interesting theorems, non-theorems that "look like" theorems as well as many copies and alpha-renamings of lemmas occurring in the training data. Symbolic theory exploration methods are usually better at targeting more specifically novel conjectures, but struggle with large scale theories instead.

We believe that the most promising way forward is a combination of neural and symbolic methods, where the neural part makes suggestions of potential analogies to similar theories seen before, while they symbolic part fills in the details in such a way that redundant conjectures are avoided. Heras et al. demonstrated a prototype system similar to what we propose, for suggesting lemmas by analogy via templates [4]. Here, the user is supposed to be wanting to prove a particular conjecture, and asks the system for analogous prior theorems. If such a similar theorem exists (determined by data-driven methods), and its proof uses a lemma, then the lemma was generalized into a template. This template was then instantiated using symbols relevant to the new proof attempt at hand. Our proposed system differs in that we do not want to rely on a particular proof attempt, but rather suggest interesting conjectures based on the functions and datatypes in scope, and how they are defined.

References

[1] S. H. Einarsdóttir, M. Johansson, and J. Å. Pohjola. Into the infinite - theory exploration for coinduction. In *Proceedings of AISC 2018*, pages 70–86, 01 2018.

[2] S. H. Einarsdóttir, N. Smallbone, and M. Johansson. Template-based theory exploration: Discovering properties of functional programs by testing. In *IFL 2020: Proceedings of the 32nd Symposium on Implementation and Application of Functional Languages*, IFL 2020, page 67–78, New York, NY, USA, 2020. Association for Computing Machinery.

[3] K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, and J. B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, page 835–850, New York, NY, USA, 2021. Association for Computing Machinery.

[4] J. Heras, E. Komendantskaya, M. Johansson, and E. Maclean. Proof-pattern recognition and lemma discovery in ACL2. In *Proceedings of LPAR*, 2013.

[5] M. Johansson, D. Rosén, N. Smallbone, and K. Claessen. Hipster: Integrating theory exploration in a proof assistant. In *Proceedings of CICM*, pages 108–122. Springer, 2014.

[6] M. N. Rabe, D. Lee, K. Bansal, and C. Szegedy. Mathematical reasoning via self-supervised skip-tree training. In *Proceedings of ICLR*, 2021.

[7] N. Smallbone, M. Johansson, K. Claessen, and M. Algehed. Quick specifications for the busy programmer. *Journal of Functional Programming*, 27, 2017.

[8] J. Urban and J. Jakubův. First neural conjecturing datasets and experiments. In *Proceedings of CICM*, 2020.