Extending Elpi functionality towards hierarchical reinforcement learning

Patrick Nicodemus

University of Pennsylvania

Recent work in automated theorem proving has focused on using neural networks or other statistical learning tools to guide proof search. Often, reinforcement learning combined with imitation learning has outperformed imitation learning alone.

In program synthesis and inductive logic programming, *hierarchical* reinforcement learning (HRL) has been deployed effectively to learn program synthesis incrementally over time. In [4], the authors show that it is possible to learn a library of programs over time, allowing a program synthesis model to learn to solve more difficult problems. Although HRL is a broad category, we will here use it to mean something more specific, what [3] calls "bias reformulation" and "dependent learning": throughout the reinforcement learning process, the model accumulates a library of programs in a cumulative manner, building a *hierarchy* of programs, each new program *dependent* on the ones below it; after each refactoring of the program library, the search tree for new programs has been reshaped (bias reformulation).

Proof tactic-script synthesis is closely related to program synthesis, except that program synthesis is usually driven by examples whereas proof synthesis involves constructing any term which inhabits the desired type. But HRL in proof synthesis has been limited so far; first steps were made by [22] and [18]. We propose to extend existing work in this direction, following the explicit call for this in [22]. For neurally guided proof search, the neural model needs to learn less information when existing skills are codified into compound tactics which are added explicitly to the search tree, or if existing tactics can be replaced with abstractions which fulfill the same purpose with greater generality (as in [7])

We propose to implement such a hierarchical RL system for Coq, using inductive logic programming to recognize reusable proof components. There is substantial existing work on recognizing proof patterns which we believe can be leveraged to form the "skill consolidation stage" of a hierarchical RL loop, where the tactic library is refactored.

The most relevant recent work is due to G. Grov and collaborators, [10], [9] [8], and his students, [5], [21]. In [8] and [5] the authors use ILP to identify and extract general proof strategies from existing tactic proofs in Isabelle, with emphasis on identifying preconditions for a tactic to be applied, using types to prune the search space, and using dependent learning to build up to more difficult problems. Although they do not address this specifically, it is clear that their work could be applied in a reinforcement learning loop to learn a library of tactics, giving a form of learned inductive bias.

AI4FM (Artificial Intelligence for Formal Methods) was a four-year research program to develop AI methods to extract reusable patterns from formal proofs; a late-project case study illustrating their results is [6].

Other work on learning new tactics from existing proofs is [2], [16]. Komendantskaya and Heras studied proof patterns in Coq but with a focus on interactive proof guidance rather than RL, [13], [14], [12], [11], [17], and in ACL2, [15].

We believe this existing body of work shows that proof pattern recognition is a mature subject and that the time has come to incorporate it into reinforcement learning for proof search.

However, the most common tactic language $\mathcal{L}$tac is not ideal for this task. As pointed out in [19] and [20], its grammar and semantics are not consistent (it has notational exceptions to make it easier to read and write, and heuristics to guess what the user means in ambiguous cases). Its tactics often prioritize user convenience over predictability of outcome. Both of these features make it a poor object language for serious program analysis and refactoring. Instead we propose to develop a proof-search environment and a theory of proof refactoring in *Elpi*, a dialect of λ-Prolog which has been a tactic language for Coq since 2015. Elpi will serve as both the *object language* (as tactic language, we will attempt to extract reusable components from Elpi programs, and otherwise abstract, generalize and compose Elpi tactics) and the *metalanguage* (we will use Elpi for the inductive logic programming component). As a logic programming language, Elpi is naturally suitable for proof search. Elpi has native support for managing free and bound variables which as well as support for constraint-logic programming, which makes it suitable for automated proof of theorems involving existential quantifiers. Frequently, papers on neurally guided proof-search write off the problem of term synthesis as being a hard problem outside the scope of their paper, and restrict tactic arguments to be a hypothesis, variable or constant. Taking Elpi as our tactic language gives a realistic chance of a proof search tool being able to instantiate an existential quantifier.

More fully, what we propose is the following:

- Write a complete proof search procedure for Coq in Elpi.
- Design a clean, robust interface for an external guidance system to interface with Coq and guide proof search using reinforcement learning. Following [23] as a model, instrument Elpi with "oracle" clauses to let it communicate with a model that learns to guide proof search, and similarly instrument the interpreter to save the search stack so that the whole search process can be steered by a top level control system which takes the whole search stack as input.
- Develop a theory of refactoring, generalizing, abstracting and composing Elpi programs, and design an algorithm to extract reusable program components from the trace of a successful proof search, drawing on work mentioned above, as well as [1]
- Develop an ILP algorithm to identify sufficient conditions under which tactics can be applied, as in [8], in order to prune the proof search space.

In addition, existing neural proof search performs premise selection either by using a fixed set of hand-coded features, a learned set of features, or a neural network embedding. We are not aware of any existing work where the proof-search automaton has the ability to simply directly search its environment for all theorems matching with a given pattern, i.e., the goal contains the expression $n + 1$ and the automaton searches the library for theorems of the form $(\_ + 1 = \_)$. We suggest that providing such a search function to the model reduces the amount it has to learn, as instead of memorizing a theorem it can simply search based on the pattern it needs and this theorem will come up. This is not the same as fetching theorems based on a heuristic derived from the number of tokens in common; the idea is that the model learns over time to query the existing base of theorems for formulas having specific properties.

References

[1] Sebastijan Dumancic, Tias Guns, and Andrew Cropper. "Knowledge Refactoring for Inductive Program Synthesis". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.8 (May 18, 2021), pp. 7271–7278. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v35i8.16893. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16893 (visited on 04/24/2023).

[2] Hazel Duncan. "The Use of Data-Mining for the Automatic Formation of Tactics". UK: School of Informatics, University of Edinburgh, 2007. 151 pp. URL: http://gpbib.cs.ucl.ac.uk/gp-html/hazelthesis.html.

[3] Kevin Ellis et al. "Bias Reformulation for One-Shot Function Induction". In: *Frontiers in Artificial Intelligence and Applications*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2014, p. 7. ISBN: 978-1-61499-419-0-525. DOI: 10.3233/978-1-61499-419-0-525. URL: https://dspace.mit.edu/handle/1721.1/102524.

[4] Kevin Ellis et al. *DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning*. June 15, 2020. arXiv: 2006.08381 [cs]. URL: http://arxiv.org/abs/2006.08381 (visited on 04/23/2023). preprint.

[5] Colin Farquhar. "Meta-Interpretive Learning of Proof Strategies". Edinburgh, Scotland: Heriot-Watt University, May 2022. 147 pp. URL: https://www.ros.hw.ac.uk/handle/10399/4706 (visited on 03/28/2023).

[6] Leo Freitas et al. "Verifying the Heap: An AI4FM Case Study". In: *Contributions to {AI4FM} 2013*. ITP-WS-13. Technical Report, Heriot-Watt University, pp. 7–9. URL: http://www.macs.hw.ac.uk/cs/techreps/docs/files/HW-MACS-TR-0100.pdf.

[7] Thibault Gauthier et al. "TacticToe: Learning to Prove with Tactics". In: *Journal of Automated Reasoning* 65.2 (Feb. 2021), pp. 257–286. ISSN: 0168-7433, 1573-0670. DOI: 10.1007/s10817-020-09580-x. arXiv: 1804.00596 [cs]. URL: http://arxiv.org/abs/1804.00596 (visited on 04/23/2023).

[8] Gudmond Grov et al. "Typed Meta-Interpretive Learning for Proof Strategies". In: 25th Annual Conference on Inductive Logic Programming. Kyoto, Japan, pp. 17–32. ISBN: 978-3-319-40566-7. DOI: 10.1007/978-3-319-40566-7.

[9] Gudmund Grov, Aleks Kissinger, and Yuhui Lin. "A Graphical Language for Proof Strategies". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Ken McMillan, Aart Middeldorp, and Andrei Voronkov. Red. by David Hutchison et al. Vol. 8312. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 324–339. ISBN: 978-3-642-45220-8. DOI: 10.1007/978-3-642-45221-5_23. URL: http://link.springer.com/10.1007/978-3-642-45221-5_23 (visited on 04/24/2023).

[10] Gudmund Grov and Ewen Maclean. *Towards Automated Proof Strategy Generalisation*. June 9, 2013. arXiv: 1303.2975 [cs]. URL: http://arxiv.org/abs/1303.2975 (visited on 04/24/2023). preprint.

[11] Jónathan Heras and Ekaterina Komendantskaya. "ML4PG in Computer Algebra Verification". In: *Intelligent Computer Mathematics*. Ed. by Jacques Carette et al. Red. by David Hutchison et al. Vol. 7961. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 354–358. ISBN: 978-3-642-39320-4. DOI: 10.1007/978-3-642-39320-4_28. URL: http://link.springer.com/10.1007/978-3-642-39320-4_28 (visited on 04/24/2023).

[12] Jónathan Heras and Ekaterina Komendantskaya. *Proof Pattern Search in Coq/SSReflect*. Feb. 1, 2014. arXiv: 1402.0081 [cs]. URL: http://arxiv.org/abs/1402.0081 (visited on 04/24/2023). preprint.

[13] Jónathan Heras and Ekaterina Komendantskaya. "Recycling Proof Patterns in Coq: Case Studies". In: *Mathematics in Computer Science* 8.1 (Mar. 2014), pp. 99–116. ISSN: 1661-8270, 1661-8289. DOI: 10.1007/s11786-014-0173-1. URL: http://link.springer.com/10.1007/s11786-014-0173-1 (visited on 04/24/2023).

[14] Jónathan Heras and Ekaterina Komendantskaya. *Statistical Proof Pattern Recognition: Automated or Interactive?* Mar. 5, 2013. arXiv: 1303.1419 [cs]. URL: http://arxiv.org/abs/1303.1419 (visited on 04/24/2023). preprint.

[15] Jónathan Heras et al. "Proof-Pattern Recognition and Lemma Discovery in ACL2". In: *Logic for Programming, Artificial Intelligence, and Reasoning*. Ed. by Ken McMillan, Aart Middeldorp, and Andrei Voronkov. Red. by David Hutchison et al. Vol. 8312. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 389–406. ISBN: 978-3-642-45221-5. DOI: 10.1007/978-3-642-45221-5_27. URL: http://link.springer.com/10.1007/978-3-642-45221-5_27 (visited on 04/24/2023).

[16] M. Jamnik. "Automatic Learning of Proof Methods in Proof Planning". In: *Logic Journal of IGPL* 11.6 (Nov. 1, 2003), pp. 647–673. ISSN: 1367-0751, 1368-9894. DOI: 10.1093/jigpal/11.6.647. URL: https://academic.oup.com/jigpal/article-lookup/doi/10.1093/jigpal/11.6.647 (visited on 04/24/2023).

[17] Ekaterina Komendantskaya, Jónathan Heras, and Gudmund Grov. "Machine Learning in Proof General: Interfacing Interfaces". Version 2. In: (2012). DOI: 10.48550/ARXIV.1212.3618. URL: https://arxiv.org/abs/1212.3618 (visited on 04/24/2023).

[18] Zhening Li et al. *LEMMA: Bootstrapping High-Level Mathematical Reasoning with Learned Symbolic Abstractions*. Nov. 15, 2022. arXiv: 2211.08671 [cs]. URL: http://arxiv.org/abs/2211.08671 (visited on 04/24/2023). preprint.

[19] *Ltac2 Documentation*. URL: https://coq.inria.fr/refman/proof-engine/ltac2.html.

[20] Pierre-Marie Pedrot. "Ltac2 - Tactical Warfare". In: The Fifth International Workshop on Coq for Programming Languages CoqPL 2019. Lisbon, Portugal, Jan. 19, 2019. URL: https://www.xn--pdrot-bsa.fr/articles/coqpl2019.pdf.

[21] Iain Whiteside. "Refactoring Proofs". Thesis or Dissertation. Edinburgh, Scotland: The University of Edinburgh, Nov. 28, 2013. 254 pp. URL: https://era.ed.ac.uk/handle/1842/7970 (visited on 10/22/2013).

[22] Zsolt Zombori and Josef Urban. "Learning Complex Actions from Proofs in Theorem Proving". In: *5th Conference on Artificial Intelligence and Theorem Proving. AITP 2020*. 5th Conference on Artificial Intelligence and Theorem Proving. AITP 2020. Aussois, France, 2020. URL: http://aitp-conference.org/2020/abstract/paper_11.pdf.

[23] Zsolt Zombori, Josef Urban, and Chad E. Brown. "Prolog Technology Reinforcement Learning Prover". In: *IJCAR 2020: Automated Reasoning*. International Joint Conference on Automated Reasoning. Paris, France: Springer, 2020, pp. 489–507. ISBN: 978-3-030-51053-4. DOI: 10.1007/978-3-030-51054-1_33. URL: https://link.springer.com/chapter/10.1007/978-3-030-51054-1_33#citeas.