Isabelle/RL Project Proposal: Reinforcement learning on the Isabelle proof assistant

Jonathan Julián Huerta y Munive[1]

Czech Institute of Informatics, Robotics and Cybernetics,
Czech Technical University in Prague,
Jugoslávských partyzánů 1580/3, Prague, Czechia
huertjon@cvut.cz

Abstract

Machine learning (ML) methods have shown great promise for aiding proof automation in interactive theorem provers (ITPs). However, one of the most popular and automated ITPs, Isabelle, has been mostly ignored. This extended abstract describes an ongoing project on creating infrastructure for carrying out ML experiments in the Isabelle ITP.

1 Introduction and related work

Interactive theorem provers (ITPs), or proof assistants, are complex software for mechanical verification of mathematical proofs. Mathematicians and engineers alike use these tools in a variety of applications from certification of difficult mathematical results to the verification of safety and security properties of software and hardware systems. The most developed provers like Coq, Lean, Isabelle and the HOL-family, integrate small inference kernels, prover integrated development environments (PIDEs), flexible syntax pipelines [26, 17], external reasoning tools (hammers) [4], code-generators [7] and search functionalities [21].

The recent success of machine learning (ML) methods has fueled the integration of ML tools into ITPs to aid in the interactive proving process. However, these approaches are diverse with approximately a different method employed for each prover. For instance, for finding sequences of next possible tactics in the HOL4 prover, the pioneering TacticToe tool uses a combination of a Monte Carlo tree search algorithm with a distance-weighted k nearest neighbour (kNN) classifier [6]. This has been recently followed up by the TacticZero reinforcement learning (RL) agent [27] that uses neural networks (NNs) instead of the kNN approach. Similarly, the latest implementation of the Tactician uses a graph neural network (GNN) to suggests Coq's next tactic [3, 19], which has been compared to a kNN and a large language model (LLM) without access to Coq's proof context. The HOList project [2] provides an environment for interacting with a simplified HOL Light ITP so that its users test various models, e.g. GNNs or two-tower neural network, for ranking possible actions [1, 16]. Finally, the recent interest in the Lean prover from the mathematical community has led to the integration of large language models (LLMs) into this ITP [20, 28] attracting various ML researchers to the prover [18, 25].

Yet, few of these ML methods have been applied to one of the most popular, well-engineered, and long-standing ITPs: the Isabelle proof assistant. In particular, the focus has been on conjecturing the next provable statement, either via autoformalistion and premise selection with LLMs [8, 9, 11] or via specialised templates for proofs by induction [14, 12, 13, 15], but no general integration of ML into the Isabelle proving process has been achieved like that of the Tactician or TacticToe. Moreover, these approaches have not been adapted to do reinforcement learning (RL), which leaves the question open on whether such an approach would be as successful as all of the above. The project proposed here, Isabelle/RL, aims to answer that question while providing an integrated environment for ML experiments in the Isabelle proof assistant.

2 Progress

At the moment of writing, the project has completed its earliest stage: producing functions for generating training data out of Isabelle. Understanding Isabelle's infrastructure for theorem proving has been necessary to achieve this result. Figure 1 shows an oversimplification of Isabelle's proof states which are very context dependent. That is, after each Isabelle command (represented as dots in the image) the Isabelle state changes. In particular, the syntax, lists of available theorems/facts, and proof methods may vary. Ideally, ML algorithms should learn how these changes affect the probability that a certain obligation is proven.

For an Isabelle `.thy` file, the project can find all proofs in the file, and print a JSON-representation of each step in the proof (see Figure 2). The JSON object contains a list of theorems used in the proof, the full list of proof methods, proof obligation, and available Isabelle commands. Moreover, the project can represent Isabelle terms in different formats: plain strings, raw abstract syntax trees (ASTs), and XMLs. The output is already more detailed and context-aware than that used for other Isabelle-focused approaches [8, 9, 11]. Thus, the capacity to produce better quantity and quality of data has increased, and presumably, previous approaches should improve because of this. However, the project still does not have filtering functions to select the most relevant of the methods and theorems to use as training data. For this objective, repurposing existing functions from Isabelle's *Sledgehammer* tool [4] could suffice. Although training a premise selection algorithm is now in the project's scope.

3 Future work

RL objective Recall that the standard model for agents in RL algorithms is a Markov Decision Process (MDP). An MDP $\langle S, A, T, R\rangle$ has a set of states S, actions A, and a transition T and reward R function. In the case of ITPs, and in particular the Isabelle/RL project, the states correspond to the context-dependent data shown above. The actions are the combinations of proof-commands and arguments that the prover accepts. The transition function assigns probability 1 to the effect of each action, and the reward function should measure certain progress in the proof and confidence in its completion. Since a concrete choice of these representations is difficult, the project will incrementally progress via various steps detailed below.

Machine learning models The project's data-generating functions enables the possibility of doing various ML-and-Isabelle experiments, for instance, premise selection (given a proof method), term generation (as witnesses of existential proofs), conjecturing (given a current obligation and available theorems). To achieve the RL environment, the project will first attempt to replicate the results of TacticToe and the Tactician while trying to generate custom-made premise and method selectors, as well as next-state predictors. The experiments will guide the choice of ML models, preprocessing, and encoding for each task. This is similar to comparisons of the performance of ML models on coding tasks [22]. The ultimate objective, however, is interfacing Isabelle to an OpenAI's Gymnasium environment for doing RL in it [23]. Towards this goal, the project already has an Isabelle/Scala class that initiates an Isabelle process, but this could be replaced with Dominique Unruh's *scala-isabelle* [24]. From here, the project already has successfully connected Scala with Python via the *Py4J* library [5].

Isabelle tool Finally, from the various ML experiments, the project aims to use the most efficient and effective models to build tools, e.g. recommending next steps in Isabelle's ITP process. Again, Even if the models cannot terminate a proof, the integration of a next-step recommendation tool could already simplify the ITP experience for many newcomers.

4 Acknowledgments

A Horizon MSCA 2022 Postdoctoral Fellowship (project acronym DeepIsaHOL and number 101102608) support this project. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the European Research Executive Agency can be held responsible for them.

References

[1] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, and Christian Szegedy. Learning to reason in large theories without imitation. *CoRR*, abs/1905.10501, 2019.

[2] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An environment for machine learning of higher order logic theorem proving. In *ICML 2019*, volume 97 of *PMLR*, pages 454–463. PMLR, 2019.

[3] Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. The Tactician - A seamless, interactive tactic learner and prover for Coq. In Christoph Benzmüller and Bruce R. Miller, editors, *CICM 2020*, volume 12236 of *LNCS*, pages 271–277. Springer, 2020.

[4] J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *JFR*, 9(1), 2016.

[5] Barthélémy Dagenais. Py4j. https://github.com/py4j/py4j, 2024.

[6] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. Tactictoe: Learning to prove with tactics. *JAR*, 65(2):257–286, 2021.

[7] F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In *10th Intl. Symp. on Functional and Logic Programming (FLOPS)*, volume 6009 of *LNCS*, pages 103–117, Heidelberg, 2010. Springer.

[8] Albert Qiaochu Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygózdz, Piotr Milos, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding hammers to integrate language models and automated theorem provers. In *NeurIPS 2022*, 2022.

[9] Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothée Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *ICLR 2023*. OpenReview.net, 2023.

[10] Daniel Matichuk, Toby C. Murray, and Makarius Wenzel. Eisbach: A proof method language for Isabelle. *J. Automated Reasoning*, 56(3):261–282, 2016.

[11] Maciej Mikula, Szymon Antoniak, Szymon Tworkowski, Albert Qiaochu Jiang, Jin Peng Zhou, Christian Szegedy, Lukasz Kucinski, Piotr Milos, and Yuhuai Wu. Magnushammer: A transformer-based approach to premise selection. *CoRR*, abs/2303.04488, 2023.

[12] Yutaka Nagashima. SeLFiE: Modular semantic reasoning for induction in Isabelle/HOL. *CoRR*, abs/2010.10296, 2020.

[13] Yutaka Nagashima. Faster smarter proof by induction in Isabelle/HOL. In Zhi-Hua Zhou, editor, *IJCAI 2021*, pages 1981–1988. ijcai.org, 2021.

[14] Yutaka Nagashima and Yilun He. PaMpeR: proof method recommendation system for Isabelle/HOL. In Marianne Huchard, Christian Kästner, and Gordon Fraser, editors, *ASE 2018*, pages 362–372. ACM, 2018.

[15] Yutaka Nagashima, Zijin Xu, Ningli Wang, Daniel Sebastian Goc, and James Bang. Template-based conjecturing for automated induction in Isabelle/HOL. In Hossein Hojjat and Erika Ábrahám, editors, *FSEN 2023*, volume 14155 of *LNCS*, pages 112–125. Springer, 2023.

[16] Aditya Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. *CoRR*, abs/1905.10006, 2019.

[17] Lawrence C. Paulson. *ML for the working programmer (2. ed.)*. Cambridge University Press, Cambridge, 1996. https://www.cl.cam.ac.uk/~lp15/MLbook/.

[18] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *ICLR 2023*. OpenReview.net, 2023.

[19] Jason Rute, Miroslav Olsák, Lasse Blaauwbroek, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. Graph2Tac: Learning hierarchical representations of math concepts in theorem proving. *CoRR*, abs/2401.02949, 2024.

[20] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv: Arxiv-2404.12534*, 2024.

[21] Yiannos Stathopoulos, Angeliki Koutsoukou-Argyraki, and Lawrence Paulson. SErAPIS : A concept-oriented search engine for the Isabelle libraries based on natural language. pages 1–13, 03 2020. https://www.cl.cam.ac.uk/~lp15/papers/Alexandria/Serapis.pdf.

[22] Weisong Sun, Chunrong Fang, Yun Miao, Yudu You, Mengzhe Yuan, Yuchen Chen, Quanjun Zhang, An Guo, Xiang Chen, Yang Liu, and Zhenyu Chen. Abstract syntax tree for programming language understanding and representation: How far are we? *CoRR*, abs/2312.00413, 2023.

[23] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. https://zenodo.org/record/8127025.

[24] Dominique Unruh. scala-isabelle. https://github.com/dominique-unruh/scala-isabelle, 2024.

[25] Sean Welleck and David Renshaw. LLMLean. https://github.com/cmu-l3/llmlean, 2024.

[26] Makarius Wenzel, Clemens Ballarin, Stefan Berghofer, Jasmin Blanchette, Timothy Bourke, Lukas Bulwahn, Amine Chaieb, Lucas Dixon, Florian Haftmann, Brian Huffman, Lars Hupel, Gerwin Klein, Alexander Krauss, Ondrej Kuncar, Andreas Lochbihler, Tobias Nipkow, Lars Noschinski, David von Oheimb, Larry Paulson, Sebastian Skalberg, Christian Sternagel, and Dmitriy Traytel. The Isabelle/Isar reference manual, 2023. https://isabelle.in.tum.de/doc/isar-ref.pdf.

[27] Minchao Wu, Michael Norrish, Christian Walder, and Amir Dezfouli. Tacticzero: Learning to prove theorems from scratch with deep reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *NeurIPS 2021*, pages 9330–9342, 2021.

[28] Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

A Figures

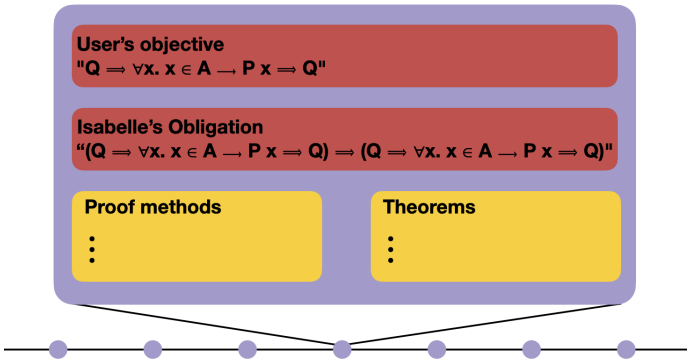

Figure 1: Isabelle's proof state

```
"state": {
 "term": "∀s'. s' ∈ X s ∪ Y s → True ⟹ X s ∪ Y s = UNIV
   ⟹ (wlp (λs.X s ∪ Y s) (λs. True) s &&& X s ∪ Y s = UNIV)"
 "hyps": [
  {"name": "∀s'. s' ∈ (if T s then X s else Y s)"},
  {"name": "Q 0"}
 ],
 "variables": [
  {"Type0": "Q, T :: 's ⇒ bool"},
  {"Type1": "s :: 's"},
  {"Type2": "Y, X :: 's ⇒ 's set"},
  {"Type3": "wlp :: ('s ⇒ 's set) ⇒ ('s ⇒ bool) ⇒ 's ⇒ bool"}
 ],
 "constants": [
  {"Type0": "All :: ('s ⇒ bool) ⇒ bool"},
  {"Type1": "(∈) :: 's ⇒ 's set ⇒ bool"},
  {"Type2": "If :: bool ⇒ 's set ⇒ 's set ⇒ 's set"},
  {"Type3": "(⟹), (&&&) :: prop ⇒ prop ⇒ prop"},
  {"Type4": "Pure.prop :: prop ⇒ prop"},
  {"Type5": "UNIV :: 's set"},
  {"Type6": "(=) :: 's set ⇒ 's set ⇒ bool"},
  {"Type7": "True :: bool"},
  {"Type8": "(∪) :: 's set ⇒ 's set ⇒ 's set"},
  {"Type9": "(⟶) :: bool ⇒ bool ⇒ bool"},
  {"Type10": "0 :: 's"},
  {"Type11": "Trueprop :: bool ⇒ prop"}
 ],
 "type variables": [
  {"Sort0": "'s :: zero"}
 ],
 "methods": [
  {"name": "Quotient.partiality_descending_setup"},
  {"name": "Transfer.transfer_prover_start"},
          ⋮
  {"name": "HOL.simp"},
  {"name": "HOL.safe"},
  {"name": "HOL.rule"},
  {"name": "HOL.auto"},
  {"name": "Pure.-"}
 ],
 "thms": [...],
 "action": "using assms(2,3)"
}
```


Figure 2: JSON object generated via the project's data-retrieving functions.