DeepIsaHOL progress report: current machine learning for 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

I report on the progress of the DeepIsaHOL project for doing modern machine learning (ML) on the Isabelle interactive theorem prover (ITP) data. The project is ongoing and focused on creating infrastructure for carrying out ML experiments with Isabelle data such as fine-tuning large language models (LLMs) or converting Isabelle into a reinforcement learning (RL) environment. The project's ultimate objective is to provide tools based on these experiments to aid Isabelle users in the proving process. This report describes the project's current state and the tools it has generated, including a generic data extraction algorithm from Isabelle's theory files, its Scala and Python interfaces, simple (Python and Scala) read, eval, print, loops (REPLs) for interacting with Isabelle programmatically, and initial use of the framework's data for training a Google's FLAN-T5 small LLM. The project remains open and welcomes ITP and ML enthusiasts to collaborate on it.

Motivation Nowadays, integrations of machine learning (ML) and interactive theorem provers (ITPs) abound in the literature [6, 23, 2, 17, 1, 10, 18, 24, 22]. Hence, the usefulness of ML methods in ITPs has been successfully evidenced to the point that some tools have already been widely adopted. For instance, the Sledgehammer tool [3] uses a combination of a k-nearest-neighbour (kNN) algorithm and a plethora of automated theorem provers (ATPs) for finding the premises that prove many of the Isabelle proof assistant's higher order logic (HOL) statements. However, the expected impact and wide-adoption in ITPs of the more recent ML methods, such as large language models (LLMs), remains to be seen.

There are several factors in the current state of the art that hinder a straightforward comparison of the published methods, and therefore, the ITP users' ability to choose a tool for their workflows. Firstly, the different implementations of the ITP-ML integrations are hard to navigate and assess. Part of this stems from the fact that ITPs themselves have diverse implementations and a solution for one does not immediately translate into a ready-to-use approach in another. Thus, given that ITPs are tools difficult to master, most users are limited to integrations implemented in their own prover of expertise. Another difficulty is due to the target learning in the ITP-ML integration. While some approaches focus on premise selection [12] (like Sledgehammer), others tackle proof-tactic selection [13, 14, 15], autoformalisation [10], conjecturing, proof reconstruction or a (partial) combination of all of these [6, 2, 17, 23, 24]. This variety of methods and integrations makes it difficult to compare them. A community-provided solution is a generic benchmark [25] comprising the formalisation of the same mathematical problems in different provers. Yet, the scope of the benchmark is proof-reconstruction which leaves out single-step approaches like premise or proof-tactic selection. Moreover, despite showing good performance in such a benchmark, recent approaches present typical ML problems: the models do not generalise nor adapt to using recently defined user-tactics, the benchmarks

might be part of the training of the LLM—making their use as evaluation metrics unsound—, and there is not enough data to train the models as in other domains.

Solution A more robust solution is to turn ITPs into environments and make them part of the models' training and evaluation. Such an integration gives the models access to the proof context that users see when interacting with the prover, enabling the models to create relationships between the proofs' states and the user-written text. Also, those integrations provide the models with the data without making it publicly available for general-purpose LLMs, allowing the creation of robust and task-specific evaluations for the different learning targets. Furthermore, a well-designed interface between the model and the assistant could allow faster deployment of custom-made ML-based proof methods that benefit ITP users. Systems like the Tactician for Coq [2], HOL-list for HOL Light [1], and LeanDojo for Lean [24] have already active teams developing and enhancing these generic integrations. However, despite being one of the most used proof assistants and having one of the largest repositories of proofs, Isabelle has been mostly left out. The Portal to Isabelle (PISA) project [8] that started such an integration has not been updated since the first half of 2023. As a response to this situation, the DeepIsaHOL project [7] reports work in progress on: a generic data extraction algorithm from Isabelle's theory files, its Scala and Python interfaces, simple (Python and Scala) read, eval, print, loops (REPLs) for interacting with Isabelle programatically, and initial use of the framework's generated data for training Google's FLAN-T5 small LLM [16, 5] from scratch.

Implementation Isabelle's core—or Isabelle/ML (for meta-language)—, manages the prover's internal state, the logical inference certification and the prover's data. The remaining Isabelle tools receive and use the core's data, and follow its lead. A consequence is that Isabelle has its own Isabelle/ML parser and its interface is its own (prover-)IDE. Respecting this philosophy, the DeepIsaHOL project created data-retrieving functions in Isabelle/ML that take as input an Isabelle `.thy` file, and produce a JSON object for each proof in the `.thy` file. To create a link between well-established ML Python libraries like Hugging Face's transformers or OpenAI's gymnasium [20], the project leverages the fact that Isabelle's communication with external tools is via the Scala programming language. In this case, the project used the `scala-isabelle` library [21] for lifting Isabelle/ML functions to the Scala level. From here, the `Py4j` library, connecting python with the Java virtual machine, enables the usage of these functions at the Python level too. Due to these integrations, users of the project's libraries can create a database from Isabelle's `.thy` files, start an interactive session with Isabelle, and even run the project's scripts for training T5 LLMs on the generated data (see usage examples in the repository [7] and the data example in the Appendix B).

Conclusion and future work The project is well-positioned to develop training algorithms for premise and tactic selection, term generation (as witnesses for existential proofs), conjecturing, proof-reconstruction, and reinforcement learning. Another sensible step could be to reproduce results from previous works and compare them. For instance, a contrast between the kNN and neural network approaches for premise selection only appears in the Tactician for Coq [2, 17]. It would be interesting to learn if a different choice of neural network and data-structure will lead to similar results as those obtained there. Yet, the project's main objectives are to provide an RL environment based on Isabelle, and to create a tool for its users. A valid method-and-premises suggestion tool would already be valuable for novice users. An invitation to Isabelle or ML experts to collaborate on DeepIsaHOL, or to provide feedback that accelerates its progress remains open.

References

[1] 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.

[2] 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.

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

[4] Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *ICLR 2019*. OpenReview.net, 2019.

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. https://arxiv.org/abs/2210.11416, 2022.

[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] Jonathan Julian Huerta y Munive. DeepIsaHOL. https://github.com/yonoteam/DeepIsaHOL, November 2023.

[8] Albert Q. Jiang, Wenda Li, Jesse Michael Han, and Yuhuai Wu. Lisa: Language models of isabelle proofs. *6th Conference on Artificial Intelligence and Theorem Proving*, 2021.

[9] 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.

[10] 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.

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

[12] 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.

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

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

[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] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

[24] 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.

[25] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

A 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.

B Example of Printed Data

An example of a JSON object generated generated by the project's data extraction functions appears in Figure 1. The data extracted in Figure 1 is more detailed and context-aware than that of other Isabelle-focused approaches [9, 8, 12]. Moreover, the algorithm extracts more proofs and proof-steps than previous works. This is because the functions not only extract proof-data from typical "lemma" and "theorem"-heading proofs, but they also do so for various Isar proofs and from class, function and type definitions.

```
"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 1: JSON object generated via the project's data-retrieving functions.