# Reaping the first fruits of the Isabelle/RL project

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**

The proposed presentation aims to update the audience on the achievements of the Isabelle/RL project presented at AITP 2024. In particular, the project has contributed an openly available framework for interacting in Python with the Isabelle proof assistant, a small-scale experiment evaluating the proof capabilities of 5 encoder-decoder transformers, and the use of the framework for creating a tool that automatically fixes (simple) broken verification proofs due to small changes in the upstream theory stack. The presentation also aims to discuss future research avenues from the existence of this framework.

The Isabelle/RL project, proposed in AITP 2024 [4], has grown from a simple data-extraction algorithm for the Isabelle interactive theorem prover (ITP) to a set of tools for interfacing with the prover, training and evaluating machine learning models, and proving simple Isabelle theorems with them. Despite this progress, the project's objectives have not been attained yet. This extended abstract discusses the project's achievements and upcoming steps.

**Achievements** The project has attained the following milestones:
1. A *data-extraction algorithm* with a Scala and a Python command line interface. The algorithm complements older Isabelle-focused approaches [6, 7] by supplying typing data for constants and variables which has been previously reported as missing [8]. To run the algorithm, users must specify a "read" directory and an "output" directory. The read directory must have at least one `.thy` file. Users can include a `ROOT` file describing an Isabelle session, or a `ROOTS` file listing several sessions with their own directories, `ROOT` files, and `.thy` files. The output directory will mimic the structure of the read directory. Additionally, it will create a directory for each `.thy` file, with the same name, and containing one `.json` file for each proof in the original `.thy` file. An example of a generated `.json` can be seen in Appendix A.
2. Python-based *training and evaluation loops* for Google's T5 [1, 9] and Gemma 3n [11] models. Chosen for their relatively small sizes and open availability, most users can download these models and run them locally. The project has produced various versions of the T5 models which are publicly available in a snapshot of the project's repository [5]. Specifically, for computational and time constraints, the project has not used the `large` (783M parameters), `xl` (2.85B), and `xxl` (11.3B) versions of the `google/flan-t5` family of models. Instead, it has focused on training from scratch and fine-tuning the `small` (77M) and `base` (248M) varieties. A comparison of the proving capabilities of these models appears in Table 1. The T5 models have a typical encoder-decoder architecture, very similar to that of the original transformer [12] with minor changes [9], while the Gemma 3n model (2B) has a Matryoshka architecture described as nesting a smaller model within a larger model. Upcoming work will evaluate the Gemma 3n model and compare the current results with more architectures.
3. Scala and Python *read-eval-print-loops* (REPLs) for interacting with the prover programmatically. The objective is to enable users to interface their models with Isabelle purely on the Python side, where popular libraries for machine learning abound (e.g. Hugging Face, Unsloth, Axolotl, Gymnasium). As before, the REPLs receive as input (at most) a "logic" (an Isabelle

session) and the name of a `.thy` file, enabling them to load Isabelle at the appropriate proof context. From there, simple commands (e.g. `apply`, `reset`, `undo`) enable users to execute or undo Isabelle "actions" by using as arguments typical Isabelle-user strings. Other methods provide context information (e.g. `last_proof_of`, `is_at_proof`, or `last_error`). As output, the REPL prints the output that users would usually see in Isabelle's output panel.
4. A depth-first search (DFS) algorithm for evaluating the models' ability to prove theorems from a specified split (train, validation, or test) of the generated data. Crucially, the DFS loop leverages the Python REPL and the trained/fine-tuned models. For each logic in the split, the loop loads the REPL in the appropriate Isabelle context and queries the model for the next Isabelle step. The breadth and depth of attempts in the DFS algorithm are controlled from an input `.json` configuration file. The results of the evaluation for the T5 models with 5 attempts up to 5-levels deep in the proof appear in Table 1. Future work will explore variations of these meta-parameters and apply the DFS loop to the recently fine-tuned Gemma 3n model.
6. The knowledge and tools used here have been crucial for developing the `super_fix` tool for automatically repairing large proof scripts leveraging Sledgehammer [10]. In a large proof development with several iterations, manual calls to Sledgehammer for fixing easy but broken proof steps do not scale. Instead, `super_fix` uses the Isabelle/RL project's libraries to traverse `.thy` files with failed proof steps and automatically replace them with updated versions. The tool has been used in a large mechanised proof of properties of a cache coherence protocol [10]. Extensions to the tool leveraging the LLMs are upcoming work.
7. The project has applied the data extraction algorithm to the AFP (2024). It has also used the DFS loop to perform a small-scale experiment comparing different training configurations for the T5 models. That is, it evaluates the models' ability to prove theorems via the DFS varying the training setup (from scratch or fine-tuning) and the data extend (whether to include premises and keywords or not). The project has published a snapshot of the models' weights and the algorithm for this experiment in a publicly available snapshot [5]. A summary of the models' behaviour appears below:

| **Model** | **Predictions** | **Progress steps** | **Attempts** | **Finished** | **Avg. time** |
|---|---|---|---|---|---|
| small_s | 4,185,571 | 62.48% | 24,071 | 6.63% | 48.92 s |
| finet_small_s | 2,055,838 | 37.29% | 24,740 | 11.46% | 40.09 s |
| small_spk | 1,859,629 | 49.01% | 27,597 | 8.55% | 29.84 s |
| base_s | 2,983,954 | 60.65% | 20,252 | 8.92% | 50.33 s |
| finet_base_spk | 2,315,016 | 52.57% | 38,532 | 16.71% | 53.28 s |

Table 1: Evaluation of T5 models on automatically proving theorems

The column *Progress steps* tells the percentage of model predictions (first column) that were accepted by the ITP but that do not necessarily finish the proof. The column *Finished* displays the percentage of automatically completed proofs out of all the attempted ones (third column). The last column indicates the average amount of time that each model spent processing the attempted proofs. The `finet` prefix indicates that the model was fine-tuned on the project's data. If the prefix is not available, the model has only been pre-trained. The two sizes `small` (77M parameters) and `base` (248M parameters) also appear in Table 1. Finally, the suffix `spk` indicates that the data includes, besides the user-proof up to that point, the user-seen *state*, the *premises* that the user used to complete the proof, and the available *keywords* at that point. This is in contrast to the suffix `s` that only adds the user-*state* to the proof up to that point.

**Next steps** The project is currently focused on calling next-step suggestion external functions, such as those by the trained models, via the Isabelle proof assistant. The objective is to

simplify the tooling process so that future developments in automated theorem proving can be immediately adapted as an Isabelle tool. To that purpose, the project has already created a Python server and a Standard ML client that are being adapted to query the trained language models via Isabelle. Currently, one needs to provide the path to the current file and the line number (where the prediction must be made) as an Isabelle tactic. The model recommends the next step and sends it back to Isabelle, which displays it in its output panel. We are in constant communication with the Isabelle developers who have already provided extensions to Isabelle that facilitate reimplementing our methods via official Isabelle tools.

The small-scale experiment is easy enough to set up that trying more recently optimised models with the framework, such as Gemma 3 [11] or DeepSeek R1 [2], should be fairly simple and executed in parallel with other objectives of the project. The project has already fine-tuned a Gemma 3n model, and it is in the process of evaluating it in the DFS loop.

Finally, the project's initial objective was to turn Isabelle into a reinforcement learning environment. The amount of data generated from the project is not representative of all the knowledge that Isabelle users acquire to tackle the provers' interactive commands. The expectation is that a reinforcement learning setting will enable a machine learning model to estimate a distribution based on more varied data.

As such, the project intends to implement a Markov Decision Process (MDP) $\langle S, A, T, R\rangle$ such that the states ($S$) correspond to the context-dependent data used in the above evaluations, the actions ($A$) correspond to user actions (combinations of proof-commands and arguments that the prover accepts), the (learned) transition function ($T$) assigns a "probability" of success to each action, and the reward function ($R$) awards the most points for early proof completions but still positive points for non-repeated proof progress. Iterative improvements to the reward function will be guided by the study and analysis of its previous versions.

# 1 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] 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.

[2] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. https://arxiv.org/abs/2501.12948.

[3] Huerta y Munive, Jonathan Julián. Isabelle/RL project proposal: Reinforcement learning on the Isabelle proof assistant. AITP2024. EasyChair, 2024.

[4] Huerta y Munive, Jonathan Julián. Snapshot of contributions from the DeepIsaHOL project, March 2025. https://doi.org/10.5281/zenodo.15080049.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Chengsong Tan, Alastair F Donaldson, Huerta y Munive, Jonathan Julián, and John Wickerson. The burden of proof: Automated tooling for rapid iteration on large mechanised proofs. 2025. https://doi.org/10.1109/FormaliSE66629.2025.00010.

[10] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024. https://arxiv.org/abs/2403.08295.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

# A Appendix: Format and content of generated data

```
{
  "proof": {
    "steps": [
      {
        "step": {
          "action": "lemma in_empty_table[simp]: \"\<not> x \<in>
              empty_table\"",
          "user_state": "",
          "term": "x \<notin> empty_table \<Longrightarrow> (x \<notin>
              empty_table)",
          "hyps": [],
          "proven": [],
          "variables": [{"Type0": "x :: 'a"}],
          "constants": [
            {"Type0": "Pure.prop :: prop \<Rightarrow> prop"},
            {"Type1": "empty_table :: 'a set"},
            ...
            {"Type5": "(\<Longrightarrow>) :: prop \<Rightarrow> prop \<
                Rightarrow> prop"}
          ],
          "type variables": [{"Sort0": "'a :: type"}]
        }
      },
      { ... }  // Other steps omitted for brevity
    ],
    "apply_kwrds": [
      {"name": "\\<proof>"},
      {"name": "sorry"},
      {"name": "..."},  // Many apply-style commands omitted
    ],
    "isar_kwrds": [
      {"name": "define"},
      {"name": "assume"},
      {"name": "..."},  // Many Isar-style commands omitted
    ],
    "methods": [
      {"name": "Metis.metis"},
      {"name": "Transfer.transfer_prover_start"},
      {"name": "..."},  // Many methods omitted
    ],
    "deps": [
      {"thm": {"name": "Pure.protectIPure.protectI", "term": "PROP ?A \<
          Longrightarrow> (PROP ?A)"}},
      {"thm": {"name": "HOL.eq_reflectionHOL.eq_reflection", "term": "?x
           = ?y \<Longrightarrow> ?x \<equiv> ?y"}},
      ... // More dependencies omitted
    ]
  }
}
```