Thesis Proposal Review

July 14, 2023

1 Overview

This research focuses on the automation of Interactive Theorem Proving (ITP) systems through Artificial Intelligence and Machine Learning. ITP systems are central to modern formal verification efforts as they allow for more complex proofs than traditional Automated Theorem Proving (ATP) approaches[27, 10]. ATP systems generally consist of fully automated heuristic based search algorithms which scale poorly with the complexity of the task. ITP systems instead require a human to guide the proof system at a higher level of abstraction, and can utilise ATPs for proving smaller, more tractable subgoals. Proof guidance requires human expertise in both the area of reasoning and the proving system itself. This limits the scalability of these approaches, with large scale formalisation projects taking many years to complete[34] As a result, progress in the automation of this proof guidance could improve the scalability of formal verification approaches.

2 Literature Review

The application of AI and Machine Learning methods to Mathematical Reasoning is an emerging research area. There are different paradigms within this, and[21] presents a comprehensive overview with a summary of progress to date. The scope of this project, at present, is on the application of AI/ML methods to Interactive Theorem Proving (ITP).

2.1 Interactive Theorem Proving

Interactive Theorem Proving (ITP) is a method of formal verification where a human guides a proof system at a high level. Given a goal to prove, the human selects a *tactic* to apply to the goal, which defines the strategy to use. The tactic can also include *arguments*, which are usually previously proven theorems (*premises*) or variables in the goal (*terms*)[1]. The system then attempts to prove the statement with the provided tactic. It returns either an empty list indicating the goal has been proven, or a list of subgoals, the conjunction of which is equivalent to the original goal. Repeating this process leads to a proof tree, with leaves being goals and edges the tactics which generated them. The selection of a goal to prove from this tree is a key part of ITP guidance.

2.1.1 Systems

There are many ITP systems currently available. For our purposes, the main systems used to investigate ITP automation are HOL4 [31], HOL Light [9], Coq[28], Isabelle/HOL[29], Lean[2] and MetaMath[23]

2.1.2 Applications

ITP has been an essential tool for many large scale formal verification efforts. Some of the most noteworthy applications include:

- Fully verified compilers: The CakeML[32] project includes a verified compiler for Standard ML[24], a functional programming language. CompCert[19] is a C compiler which was verified using the Coq ITP. It has been used in safety critical applications such as flight controllers[3].
- seL4[18] is a microkernel verified using Isabelle/HOL and HOL4[11]. It has been used by NASA in their High Performance Spaceflight Computing (HPSC) program[26], among other high integrity applications [33, 22, 12].
- Verified correct vote-counting for large scale elections[25]
- Formalising pure mathematics. Some key examples are a formal proof of the Kepler Conjecture[7] and the four colour theorem[4].
- Provably correct hardware design[6, 17]

2.2 Current results in Automating ITP

2.2.1 Datasets and benchmarks

Progress in the area of ITP automation has been measured with benchmarks spanning several ITP frameworks. These benchmarks either measure direct proof performance through an ITP interface, or use datasets constructed from ITP proof traces. These proof traces are used to construct supervised learning tasks, such as premise selection or tactic prediction.

As discussed in [21], current benchmarks include

- HOList [1]

[1] Arbitrary arguments can also be provided, although in the context of AITP we usually restrict the arguments to premises and tactics.

- HOLStep [16]
- MIZAR [14, 5]
- CoqGym [36]
- PISA [15]
- LeanStep [8]
- Lean-Gym [30]
- mini-F2F [37]
- INT [35]
- IsarStep [20]
- GamePad [13]

2.2.2 Approaches

3 Proposed Research

3.1 Comparison of Embedding Architectures

3.2 Improving Proof Guidance

3.3 Open Source Experimentation Framework

- AITP large field. Subproblem automated ITP.
- many separate benchmarks across different ITPs.
- many separate models and approaches.
- Some approaches not openly available
- Want to be able to compare approaches easily between different ITP domains
  - Goal selection approaches (fringes, BFS, DFS, MCTS),
  - Embedding approach (Pre-trained + fine tuned LLM, GNN, Tree LSTM, DAG LSTM, Autoencoder)
  - tactic selection (Decoder LLM, LSTM, MLP)
  - tasks/ training approach ((end to end RL, supervised learning (premise selection, tactic prediction)))
- We present a unified framework for experimentation in automating ITP. Includes

  - Modular separation of data, experiments, models and environments.
  - Framework uses wandb and pytorch lightning. Allows for easy reproducibility and general logging, checkpointing and data loading capability.
  - HOL4 Reinforcement Learning environment, TacticZero algorithm (original, and extended for End-to-End embeddings, no current code available)
  - HOL4 Premise selection task (novel)
  - HOLStep Premise Selection task
  - HOList Environment (Deepmath-light, easy to run, language agnostic)
  - MIZAR 40 Premise Selection Task (no previous implementation)

- Using this, we present a comparison of embedding architectures across these and show..

- modular and extensible central benchmark and experimentation platform.

3.4 Extension to large scale verification projects

Most work only on formalising pure maths, remains to be seen how it does on a larger project.

4 Progress and Timeline

A summary of the current progress is given below:

- Reimplemented TacticZero algorithm.

- Generation of HOL4 premise selection pretraining dataset

- Implemented Graph Neural Network (GNN), Transformer, Structure Aware Attention (SAT) and novel Relation Attention embedding algorithms

- Tested performance of GNN and Transformer end to end embedding architectures on HOL4 TacticZero task. Demonstrated significant improvement (50%+)

- Experimentation framework implemented which includes:
  - Modular separation of data, experiments, models and environments.
  - Framework uses wandb and pytorch lightning. Allows for easy reproducibility and general logging, checkpointing and data loading capability.
  - HOL4 Reinforcement Learning environment, TacticZero algorithm (original, and extended for End-to-End embeddings)

  - HOL4 Premise selection task
  - HOLStep Premise Selection task
  - HOList Environment
  - MIZAR 40 Premise Selection Task

- Refactored HOList benchmark to be easier to run, with PyTorch integration, and not need Google Dependencies.

References

[1] Kshitij Bansal et al. *HOList: An Environment for Machine Learning of Higher-Order Theorem Proving.* en. arXiv:1904.03241 [cs]. Nov. 2019. URL: http://arxiv.org/abs/1904.03241 (visited on 04/06/2023).

[2] Leonardo De Moura et al. "The Lean Theorem Prover (System Description)". In: ed. by Amy P. Felty and Aart Middeldorp. Vol. 9195. Book Title: Automated Deduction - CADE-25 Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 378–388. ISBN: 978-3-319-21400-9 978-3-319-21401-6. DOI: 10.1007/978-3-319-21401-6_26. URL: http://link.springer.com/10.1007/978-3-319-21401-6_26 (visited on 06/26/2023).

[3] Ricardo Bedin França et al. "Formally verified optimizing compilation in ACG-based flight control software". en. In: Feb. 2012. URL: https://inria.hal.science/hal-00653367 (visited on 06/26/2023).

[4] Georges Gonthier. "The Four Colour Theorem: Engineering of a Formal Proof". en. In: *Computer Mathematics*. Ed. by Deepak Kapur. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 333–333. ISBN: 978-3-540-87827-8. DOI: 10.1007/978-3-540-87827-8_28.

[5] Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. "Four Decades of Mizar: Foreword". en. In: *Journal of Automated Reasoning* 55.3 (Oct. 2015), pp. 191–198. ISSN: 0168-7433, 1573-0670. DOI: 10.1007/s10817-015-9345-1. URL: http://link.springer.com/10.1007/s10817-015-9345-1 (visited on 06/26/2023).

[6] Aarti Gupta. "Formal hardware verification methods: A survey". en. In: *Formal Methods in System Design* 1.2 (Oct. 1992), pp. 151–238. ISSN: 1572-8102. DOI: 10.1007/BF00121125. URL: https://doi.org/10.1007/BF00121125 (visited on 06/26/2023).

[7] Thomas Hales et al. "A FORMAL PROOF OF THE KEPLER CONJECTURE". en. In: *Forum of Mathematics, Pi* 5 (Jan. 2017). Publisher: Cambridge University Press, e2. ISSN: 2050-5086. DOI: 10.1017/fmp.2017.1. URL: https://www.cambridge.org/core/journals/forum-of-mathematics-pi/article/formal-proof-of-the-kepler-conjecture/78FBD5E1A3D1BCCB8E0D5B0C463C9FBC (visited on 06/26/2023).

[8] Jesse Michael Han et al. "Proof Artifact Co-training for Theorem Proving with Language Models". In: (2021). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2102.06203. URL: https://arxiv.org/abs/2102.06203 (visited on 06/26/2023).

[9] John Harrison. "HOL Light: An Overview". en. In: *Theorem Proving in Higher Order Logics*. Ed. by Stefan Berghofer et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 60–66. ISBN: 978-3-642-03359-9. DOI: 10.1007/978-3-642-03359-9_4.

[10] John Harrison, Josef Urban, and Freek Wiedijk. "History of Interactive Theorem Proving". en. In: *Handbook of the History of Logic*. Vol. 9. Elsevier, 2014, pp. 135–214. ISBN: 978-0-444-51624-4. DOI: 10.1016/B978-0-444-51624-4.50004-6. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780444516244500046 (visited on 06/26/2023).

[11] G. Heiser. "The seL4 Microkernel – An Introduction". In: 2020. URL: https://www.semanticscholar.org/paper/The-seL4-Microkernel-%E2%80%93-An-Introduction-Heiser/829ac81db0e9b1448644576b3115a9cca60265d9 (visited on 06/26/2023).

[12] Gernot Heiser, Gerwin Klein, and June Andronick. "seL4 in Australia: from research to real-world trustworthy systems". en. In: *Communications of the ACM* 63.4 (Mar. 2020), pp. 72–75. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3378426. URL: https://dl.acm.org/doi/10.1145/3378426 (visited on 06/26/2023).

[13] Daniel Huang et al. "GamePad: A Learning Environment for Theorem Proving". In: (2018). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.1806.00608. URL: https://arxiv.org/abs/1806.00608 (visited on 06/26/2023).

[14] Jan Jakubův et al. *MizAR 60 for Mizar 50*. en. arXiv:2303.06686 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2303.06686 (visited on 05/02/2023).

[15] Albert Qiaochu Jiang et al. "LISA: Language models of ISAbelle proofs". en. In: *6th Conference on Artificial Intelligence and Theorem Proving*. 2021.

[16] C. Kaliszyk, François Chollet, and Christian Szegedy. "HolStep: A Machine Learning Dataset for Higher-order Logic Theorem Proving". In: *ArXiv* (Mar. 2017). URL: https://www.semanticscholar.org/paper/HolStep%3A-A-Machine-Learning-Dataset-for-Logic-Kaliszyk-Chollet/f1318d15d7d8ab626b92d0c70dbdc5b5d37e223f (visited on 06/26/2023).

[17] Christoph Kern and Mark Greenstreet. "Formal Verification In Hardware Design: A Survey". In: *ACM Transactions on Design Automation of Electronic Systems* 4 (Dec. 2002). DOI: 10.1145/307988.307989.

[18] Gerwin Klein et al. “seL4: formal verification of an OS kernel”. In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. SOSP ’09. New York, NY, USA: Association for Computing Machinery, Oct. 2009, pp. 207–220. ISBN: 978-1-60558-752-3. DOI: `10.1145/1629575.1629596`. URL: `https://doi.org/10.1145/1629575.1629596` (visited on 06/25/2023).

[19] Xavier Leroy. “The CompCert C verified compiler: Documentation and user’s manual”. In: (Sept. 2014).

[20] Wenda Li et al. “IsarStep: a Benchmark for High-level Mathematical Reasoning”. In: (2020). Publisher: arXiv Version Number: 2. DOI: `10.48550/ARXIV.2006.09265`. URL: `https://arxiv.org/abs/2006.09265` (visited on 06/26/2023).

[21] Pan Lu et al. “A Survey of Deep Learning for Mathematical Reasoning”. en. In: ().

[22] Everton De Matos and Markku Ahvenjärvi. “seL4 Microkernel for Virtualization Use-Cases: Potential Directions towards a Standard VMM”. en. In: *Electronics* 11.24 (Dec. 2022), p. 4201. ISSN: 2079-9292. DOI: `10.3390/electronics11244201`. URL: `https://www.mdpi.com/2079-9292/11/24/4201` (visited on 06/26/2023).

[23] Norman Megill and David A. Wheeler. *Metamath: A Computer Language for Mathematical Proofs*. en. Google-Books-ID: dxqeDwAAQBAJ. Lulu.com, 2019. ISBN: 978-0-359-70223-7.

[24] Robin Milner et al. *The Definition of Standard ML*. en. The MIT Press, 1997. ISBN: 978-0-262-28700-5. DOI: `10.7551/mitpress/2319.001.0001`. URL: `https://direct.mit.edu/books/book/2094/the-definition-of-standard-ml` (visited on 06/26/2023).

[25] Lyria Moses et al. “No More Excuses: Automated Synthesis of Practical and Verifiable Vote-Counting Programs for Complex Voting Schemes”. In: Oct. 2017, pp. 66–83. ISBN: 978-3-319-68686-8. DOI: `10.1007/978-3-319-68687-5_5`.

[26] *NASA Launches Space Cyber-resilience into a New Era with the seL4 Microkernel - DornerWorks*. URL: `https://dornerworks.com/blog/sel4-nasa-space-sbir-phase-ii/` (visited on 04/05/2023).

[27] M. Saqib Nawaz et al. “A Survey on Theorem Provers in Formal Methods”. In: (2019). Publisher: arXiv Version Number: 1. DOI: `10.48550/ARXIV.1912.03028`. URL: `https://arxiv.org/abs/1912.03028` (visited on 06/26/2023).

[28] Christine Paulin-Mohring. “Introduction to the Coq Proof-Assistant for Practical Software Verification”. en. In: *Tools for Practical Software Verification: LASER, International Summer School 2011, Elba Island, Italy, Revised Tutorial Lectures*. Ed. by Bertrand Meyer and Martin Nordio. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 45–95. ISBN: 978-3-642-35746-6. DOI: `10.1007/978-3-642-35746-`

6_3. URL: https://doi.org/10.1007/978-3-642-35746-6_3 (visited on 06/26/2023).

[29] Lawrence C. Paulson, ed. *Isabelle*. en. Vol. 828. Lecture Notes in Computer Science. Berlin/Heidelberg: Springer-Verlag, 1994. ISBN: 978-3-540-58244-1. DOI: 10.1007/BFb0030541. URL: http://link.springer.com/10.1007/BFb0030541 (visited on 06/26/2023).

[30] Stanislas Polu et al. "Formal Mathematics Statement Curriculum Learning". In: (2022). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2202.01344. URL: https://arxiv.org/abs/2202.01344 (visited on 06/26/2023).

[31] Konrad Slind and Michael Norrish. "A Brief Overview of HOL4". In: *Theorem Proving in Higher Order Logics*. Ed. by Otmane Ait Mohamed, César Muñoz, and Sofiène Tahar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 28–32. ISBN: 978-3-540-71067-7.

[32] Yong Kiam Tan et al. "The verified CakeML compiler backend". en. In: *Journal of Functional Programming* 29 (Jan. 2019). Publisher: Cambridge University Press, e2. ISSN: 0956-7968, 1469-7653. DOI: 10.1017/S0956796818000229. URL: https://www.cambridge.org/core/journals/journal-of-functional-programming/article/verified-cakeml-compiler-backend/E43ED3EA740D2DF970067F4E2BB9EF7D (visited on 06/26/2023).

[33] Steven H. VanderLeest. "Is formal proof of seL4 sufficient for avionics security?" In: *IEEE Aerospace and Electronic Systems Magazine* 33.2 (Feb. 2018), pp. 16–21. ISSN: 0885-8985. DOI: 10.1109/MAES.2018.160217. URL: https://ieeexplore.ieee.org/document/8332372/ (visited on 06/26/2023).

[34] Freek Wiedijk. "The De Bruijn Factor". In: 2000. URL: https://www.semanticscholar.org/paper/The-De-Bruijn-Factor-Wiedijk/8039ba5de3cc5f23ea3e8ccd4211a8242f18a95e (visited on 06/26/2023).

[35] Yuhuai Wu et al. "INT: An Inequality Benchmark for Evaluating Generalization in Theorem Proving". In: (2020). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2007.02924. URL: https://arxiv.org/abs/2007.02924 (visited on 06/26/2023).

[36] Kaiyu Yang and Jia Deng. "Learning to Prove Theorems via Interacting with Proof Assistants". In: (2019). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1905.09381. URL: https://arxiv.org/abs/1905.09381 (visited on 06/26/2023).

[37] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. "MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics". In: (2021). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2109.00110. URL: https://arxiv.org/abs/2109.00110 (visited on 06/26/2023).