Hardware accelerated intelligent theorem proving

Jan Heemstra
https://janheemstra.nl/

Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract

We present a connection-based tableaux theorem prover that performs inferences entirely on the GPU. Benchmarks on the m40 dataset show it performs worse than other provers in terms of proving power. In terms of inferences per second, however, it is on par with or even surpasses state-of-the-art provers on similarly priced and dated hardware, despite it lacking various important optimizations that are present in the other provers.

On top of this prover, we designed and implemented a heuristic neural network-based system that avoids evaluations during the computation of inferences, and instead precomputes the results of these evaluations.

1 Prover

Our prover is partially based on LeanCoP and MaLeCoP. LeanCoP [1] is a connection-based tableaux theorem prover based on predicate logic that can apply iterative deepening to a problem to derive a solution. MaLeCoP [2] builds on LeanCoP by adding machine learning heuristics. The Ph.D. thesis of Michael Rawson [3] describes how to perform the evaluation of the heuristics on the GPU.

LeanCoP and MaLeCoP both run on a CPU, while our prover runs on a GPU. To achieve this, our project differs from these two provers in some key areas. Instead of predicate logic, we opted to use lambda-free higher order logic [4, 5], for its simpler syntax and more powerful semantics. Furthermore, since implementing iterative deepening in a multi-threaded environment is relatively complex and requires synchronization to prevent multiple explorations of the same path, we opted to implement a simple random exploration. A future version may implement iterative deepening by repeatedly launching kernels, as is done in GPUexplore [6, 7]. Lastly, instead of evaluating heuristics during runtime, we precompute a heuristic for each pair of unifiable literals. This heuristic is created using machine learning on previously found proofs, and it approximates the relative probability of finding a proof when extending from the first literal in the pair to the second. In subsequent proof attempts, these are used as weights when randomly deciding which literal to extend to.

The prover is implemented in CUDA [8], and by default runs $2^{20} \approx 1$ million proof attempts. In each attempt, the prover randomly tries 64 proof steps. The prover does not terminate when a solution is found, so it may find more than one solution.

2 Lambda-free higher order logic

Whereas the connection method is usually implemented in the context of predicate logic, the only real requirement for the corresponding logic is the uniqueness of a most general unifier. Predicate logic has function symbols and predicate symbols which can have varying arities, which could cause unpredictable control flow.

For our prover, we examined the feasibility of using lambda calculus instead of predicate logic. First, since lambda calculus has no built-in way to create a constant such as the 0-arity

function in predicate logic, we introduce two types of atomic lambda terms: constants and variables. Lambda calculus generally does not have unique most general unifiers, as abstraction provides many ways to represent an expression. When we take abstraction away, and only keep application and lambda terms, the resulting logic becomes about as simple as it can be. We are left with the following:

1. Constants A, B, C, ...
2. Variables x, y, z, ...
3. Application $Q\,R$ for terms Q and R

Statements and theorems in predicate logic can be converted to lambda calculus by means of Currying [9]. This does however cause a slight increase in complexity, as the number of nodes in the syntax tree of the problem can in the worst case nearly double.

3 Bare Metal Theorem Prover (BMTP)

The current state-of-the-art first-order connection style theorem prover in terms of inferences per second is the Bare Metal Theorem Prover [10]. BMTP performs roughly 1.5 million inferences per second on a single core of an Intel(R) Xeon(R) Gold 6140 CPU. On an NVIDIA GeForce RTX 2080 Ti, our prover can perform about 70 million inferences per second. Given the fact that these pieces of hardware have similar price points and have been released in the same general period, this may seem like a huge win for our prover. However, where our prover uses all GPU cores, BMTP only uses a single CPU core. If we ran many instances for many problems, the gap between BMTP and our prover would close significantly. Furthermore, energy usage due to stressing a single CPU core is far lower than the energy usage of an entire GPU. Lastly, there are some discrepancies in the way inferences are counted in BMTP compared to our prover. Correcting these would bring the measured performance of BMTP to nearly equal to that of our prover.

On the other hand, BMTP is highly optimized and employs several optimization tricks that our prover thus far lacks. One such trick is to skip the first occurs check of an extension step. Since occurs checking is likely one of the most computationally intensive steps of the process, skipping them could result in a major performance boost.

4 Machine learning

Two versions of the prover, one with uniformly distributed random literal selection and one with machine-learned heuristics, were benchmarked on the m40 dataset [11]. The machine-learned heuristics improved the number of proven theorems over the uniform selection heuristic from 4783 to 5698, on a total of 32444.

Since the prover does not stop when it finds a solution, and since performance is unaffected by the heuristics, we can compare these two versions by the number of proofs they find. The improvement of the neural prover compared to the uniform prover is shown in figure 1.

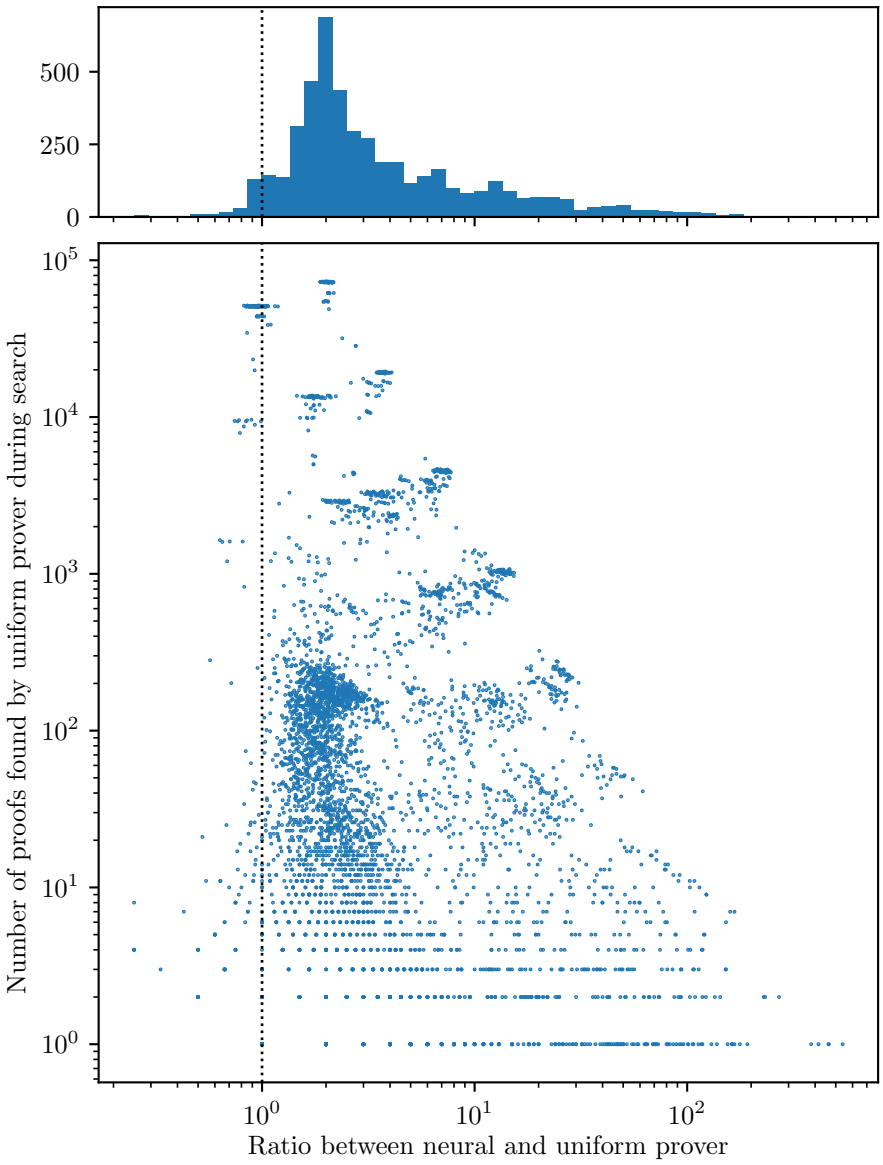

Figure 1: Histogram and scatter plot of the performance of the uniform prover and the relative performance of the neural prover. Each dot represents one theorem; the vertical position is the number of proofs found by the uniform prover, the horizontal position is the ratio between the number of proofs found by the neural prover and the uniform prover. Dots to the right of the dotted line indicate an improvement of the neural prover over the uniform prover.

The code for this project is available at https://github.com/CombustibleLemonade/CudaTP.

References

[1] Jens Otten. Lean connection-based theorem proving. http://www.leancop.de/, 20/04/2023.

[2] Josef Urban, Jirí Vyskocil, and Petr Stepánek. MaLeCoP Machine Learning Connection Prover. In Kai Brünnler and George Metcalfe, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 20th International Conference, TABLEAUX 2011, Bern, Switzerland, July 4-8, 2011. Proceedings*, volume 6793 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2011.

[3] Michael Rawson. *Applications of Machine Learning to Automated Reasoning*. PhD thesis, The University of Manchester, 2021.

[4] Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes, and Uwe Waldmann. Superposition for lambda-free higher-order logic. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning*, pages 28–46, Cham, 2018. Springer International Publishing.

[5] Petar Vukmirović, Jasmin Christian Blanchette, Simon Cruanes, and Stephan Schulz. Extending a brainiac prover to lambda-free higher-order logic. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 192–210, Cham, 2019. Springer International Publishing.

[6] Anton Wijs and Dragan Bošnački. GPUexplore: Many-Core On-the-Fly State Space Exploration Using GPUs. In Erika Ábrahám and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 233–247, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[7] Anton Wijs and Muhammad Osama. Gpuexplore 3.0: Gpu accelerated state space exploration for concurrent systems with data. In Georgiana Caltais and Christian Schilling, editors, *Model Checking Software - 29th International Symposium, SPIN 2023, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 188–197, 2023.

[8] Nvidia. Cuda toolkit documentation. https://docs.nvidia.com/cuda/, 10/05/2023.

[9] Henk Barendregt and Erik Barendsen. Introduction to Lambda Calculus, 2000.

[10] Cezary Kaliszyk. Efficient low-level connection tableaux. In Hans De Nivelle, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 102–111, Cham, 2015. Springer International Publishing.

[11] Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reason.*, 55(3):245–256, 2015.