Learning to Advise an Equational Prover*

Chad E. Brown[1], Bartosz Piotrowski[1,2], and Josef Urban[1]

[1] Czech Technical University, Prague,
[2] University of Warsaw, Poland

We describe a simple first-order equational theorem prover, `aimleap`, capable of interacting with an advisor. The prover takes 87 (fixed) quantified unit equations as axioms (all of which are true in AIM loops [4]). The conjecture is then given as an equation and `aimleap` attempts to find a proof by applying paramodulation using one of the axioms to one term occurrence in either the left or right side of the current goal. An external advisor can be used to filter and rank possible paramodulation steps. We report results of using an advisor trained using XGBoost [1] on data described below.

Proof Search We briefly describe the way `aimleap` searches for a proof of $s = t$. Note that s and t may contain constants and variables, where variables are allowed to be instantiated. Let $\mathcal{A}$ be a set of known equations. The search is limited by a bound n (maximum allowed distance) which is now always initialized to 10. We also use an abstract time limit (number of proof search inferences during) a that is set to 100 by default.

1. If s and t are unifiable, then report success.
2. If $n = 0$, then report failure.
3. Compute a finite set of *paramodulants* $s_i = t_i$. These are defined as rewrites of $s = t$ by a single equation from $\mathcal{A}$. Choose only one representative for each equivalence class of paramodulants w.r.t. renaming of variables.
4. Order these paramodulants using an advisor, filtering out those which the advisor deems to require more than $n - 1$ paramodulation steps to complete the proof, and for each one ask if $s_i = t_i$ is provable in $n - 1$ steps. I.e., this is a simple best-first search.

Primary Dataset Veroff has obtained a large number of AIM proofs using Prover9 [5]. Analysing some of these proofs it was possible to obtain 3468 equations provable from the 87 axioms within 2-10 paramodulation steps. This gave us our initial data for training and testing. Roughly half of the examples, 47.3%, was the result of 2 paramodulation steps. Another 25% resulted from 3 paramodulation steps and 10.2% resulted from 4 paramodulation steps. The remaining 18.5% resulted from 5-10 paramodulation steps, with fewer examples as the number of steps increased. We later augmented the training data with roughly 10,000 synthetically generated data, while still restricting testing to the original 3468 equations.

Rote Learner As a sanity test, an "oracle" advisor was used first. This advisor returns the known distance[1] for goals and subgoals that were seen in the training proofs. For other equations it returned a high distance of 50 (essentially forcing these equations to be pruned from the paramodulant options). We call this the "rote learning" (memorizing) advisor. As expected, the prover could reprove all 3468 problems given the full memorized information. In some cases, shorter proofs than the training proofs were found.[2] In 132 cases, new proofs involving only one paramodulation step were found. The rote learner also gave us a way to create negative training data by recording each (redundant) step where the rote learner returned 50. We have also split the 3468 problems into ten parts and used a rote learner that is allowed to look up only the values from the other nine parts. `aimleap` was only able to prove 800 (21.9%) problems in this cross-validation scenario.

*Supported by the ERC Consolidator grant no. 649043 AI4REASON

[1] Note that the *known distance* may be an over-approximation of the *real (minimal) distance*.

[2] Even though we only used the memorized data, they may still lead to alternative proofs.

Constant Distance We also tested an advisor that always returns a constant distance $d \in \{1, ..., 9\}$ whenever the left and right hand sides of the current goal are not equal (and 0 if equal). When $d = 9$, `aimleap` considers all possible 1 step proofs and some 2 step proofs,[3] approximating breadth-first search. Since just over half the problems actually do have a 1 or 2 step proof, this can solve 1739 problems (50.1%). When $d < 9$, the maximum number of problems solved was 138 (4%). Most of the search is then spent in greater depths and the abstract time of 100 runs out before trying many alternatives that may lead to a short proof.

Training the Advisor For providing machine-learned advice we used XGBoost. Training examples from the data set were fed into the model as features of pairs of terms and the corresponding distance between them. We used ENIGMA [2]-style features, i.e., paths of lengths 1-3 from the parse tree, with numbers of their occurrences. The more significant hyperparameters of XGBoost we used were as follows: objective function – mean squared error, number of boosting rounds – 1000, maximal depth of a decision tree – 10, learning rate – 0.1.

Accuracy of the Advisor We trained the model and measured its performance using the cross-validation split. The model on testing parts achieved average root mean square error (RMSE) 1.1 and average accuracy of 0.59. This means that the model with our features is able to estimate the distances between terms reasonably well. This in turn suggests its usability as an advisor for the prover.

Search Results using the Advisor Again, we run evaluation respecting the cross-validation split. The prover was given time limit of 60 s. Using the advice from the trained XGBoost models (one per cross-validation partition) we obtained proofs of 299 problems, which is 9% out of all 3468. This number is optimistic in a sense of being better than the average number of problems solved with the constant advice (but not better when $d = 9$). Because the interaction with the advisor slows the prover significantly, we believe that improvement of the implementation can make the proving performance considerably better. What is worth noting though is the number of problems solved with the advisor which were not solved by the rote learner – 135. There are also 18 problems solved with the advisor which were not solved with any d.

First-Order Automated Provers For further comparison we gave the problems to three automated provers with a timeout of 60 s: Prover9 [5], E [7] and Waldmeister [3]. E proved 2684 problems (77.4%), Waldmeister 2170 problems (62.6%) and Prover9 2037 problems (58.7%). The number of problems which were solved by `aimleap` with the machine-learned advice and not solved by these provers is 113, 92 and 49 for E, Prover9 and Waldmeister, respectively.

Conclusion and Future Work We have described an equational dataset based on the AIM project and an equational prover (`aimleap`) capable of interacting with an advisor. This provides a framework for testing the degree to which an advisor (typically one trained using machine learning techniques) is helpful during proof search. With perfect advice all the problems are easily provable by `aimleap`, however they are not all easy for standard ATPs. The current results for machine-learned advice, although preliminary, seem to be interesting and provide a baseline for further experimentation. One direction we will be investigating is using neural networks for *conjecturing* the intermediate steps. Sequence-to-sequence neural models have recently shown interesting performance in related tasks, such as predicting literals in Tableaux proofs [6], or even constructing terms from symbols. `aimleap` is already prepared for this mode of interaction.

[3] This is because the abstract time is 100 and the number of equations only 87. Deeper proof searches are pruned as soon as $d > n$.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM, 2016.

[2] Karel Chvalovský, Jan Jakubuv, Martin Suda, and Josef Urban. ENIGMA-NG: efficient neural and gradient-boosted inference guidance for E. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2019.

[3] T. Hillenbrand, A. Jaeger, and B. Löchner. Waldmeister - Improvements in Performance and Ease of Use. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, number 1632 in Lecture Notes in Artificial Intelligence, pages 232–236. Springer-Verlag, 1999.

[4] Michael K. Kinyon, Robert Veroff, and Petr Vojtechovský. Loops with abelian inner mapping groups: An application of automated deduction. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *LNCS*, pages 151–164. Springer, 2013.

[5] W. McCune. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`, 2005–2010.

[6] Bartosz Piotrowski and Josef Urban. Guiding theorem proving by recurrent neural networks. *CoRR*, abs/1905.07961, 2019.

[7] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th LPAR, Stellenbosch*, volume 8312 of *LNCS*. Springer, 2013.