# Project Proposal: Machine Learning Good Symbol Precedences*

Filip Bártek and Martin Suda

Czech Technical University in Prague, Czech Republic

## 1 Project Overview

Modern saturation-based Automated Theorem Provers (ATPs) such as E [13] or Vampire [4] rely on the superposition calculus [7] for their underlying inference system. Superposition is built around the paramodulation inference [12] crucially constrained by a simplification ordering on terms, which comes as a parameter of the calculus. Each of the two classes of simplification orderings used in practice, i.e., the Knuth-Bendix Ordering (KBO) [3] and the Lexicographic Path Ordering (LPO) [2], is mainly determined by a pair of symbol precedences—permutations on the predicate and function symbols, respectively.[1] Note that since a symbol precedence only makes sense in the context of a particular problem signature, it is genuinely problem-specific.

The choice of the precedences and thus of the simplification ordering may have a significant impact on how long it takes to solve a given problem. In a well-known example, prioritizing predicates introduced during the Tseitin transformation of an input formula [16] exposes the corresponding literals to resolution inference during early stages of the proof search, with the effect of essentially undoing the transformation and thus threatening with an exponential blow-up [11]. ATPs typically offer a few heuristics for generating the symbol precedences. For example, the successful `invfreq` heuristic in E orders the symbols by the number of occurrences in the input problem, prioritizing symbols that occur the least often for early inferences. Experiments with random precedences have shown that the existing heuristics often fail to come close to the optimum precedence [10], revealing there is a large potential for further improvements.

The ultimate goal of this project is to implement a system that, when presented with a First-Order Logic (FOL) problem, proposes symbol precedences that will likely lead to solving the problem quickly. We plan to use the techniques of supervised learning and extract such theorem-proving knowledge from successful (and unsuccessful) runs of the Vampire theorem prover when run over a variety of FOL problems equipped by randomly sampled symbol precedences. Our basic assumption is that there are abstract properties of signature symbols[2] which the learned model can utilize to determine their placement in good (from the perspective of the proving process) precedences. Moreover, we assume that by succeeding to solve already solvable problems fast, the learned knowledge will generalize to solving problems previously out of reach. This general assumption is shared with other projects for learning good theorem proving strategies from previous experience, such as the MaLeS system [6].

Note that while the domain of each precedence depends on the problem signature, we still aim to generalize the model across a wide range of FOL problems, learning theorem proving knowledge in a signature agnostic way. To this aim we are planning to use Graph Neural Networks (GNNs) [18] to extract abstract representations (feature embeddings) of the symbols

*Supported by the ERC Consolidator grant AI4REASON no. 649043 under the EU-H2020 programme.

[1] KBO is further parameterized by symbol weights, but our reference implementation in Vampire [4] uses for efficiency reasons only weights equal to 1 [5] and so we do not consider varying this parameter here.

[2] Such as the number the occurrences used by `invfreq` mentioned above.

while learning to distinguish good precedences from bad ones. The viability of using GNNs for learning in related theorem proving tasks has recently been established [17, 14, 9, 8] and it has been shown that a lot of problem structure can be captured in a way that does not hard-code symbol names into the model. This makes GNNs an ideal target architecture also for our work.

# 2 Problem-wise Predicate Precedence Learning

While keeping finer details of the overall architecture open for now, we start by focusing on learning good precedences for each problem in isolation. This preliminary task allows us to establish how much "signal to learn from" we can expect to be available. Moreover, learning from permutations presents already several interesting challenges to deal with. For one, there is a priori no obvious way how to characterise a permutation by real-valued features to serve as inputs for a learning algorithm, or how to do it in a way which does not presuppose a fixed domain size. Moreover, even with a regression model ready to predict the prover's performance under a particular precedence $\Pi$, we still need solve the task of finding the ideally optimal precedence $\Pi^*$ according to this model.

## 2.1 Initial Experiment

We based our initial experiment on the assumption that each pair of predicates $(p_1, p_2)$ contributes to the performance a precedence that orders $p_1$ before $p_2$ independently of the ordering of the other predicates in the precedence. This is in accord with how a typical human designed heuristics would be justified: by declaring that a certain class of symbols should be larger or smaller than others. Under this assumption we can aggregate the values from all of the up to $n!$ precedence-wise performance measurements (where $n$ is the number of predicates in the problem under consideration) into $n(n-1)$ pairwise predicate *preference* values.

In our first implementation, we choose the preference value of a pair of predicates $(p_1, p_2)$ to be an empirical estimate of the expected number of saturation loop iterations of a successful Vampire run that orders $p_1$ before $p_2$. We obtain this value by running Vampire with 1000 uniformly random predicate precedences for each of the training problems, attributing a high constant value to the runs that time out. The number of saturation loop iterations in a successful run is generally a good measure of the quality of a precedence, because if we reduce the number of saturation loop iterations for a wide variety of problems, we are likely to solve previously unsolved problems.

Once we have access to good pairwise preference values $\mathit{pref}(p_1, p_2)$, we proceed to construct a predicate precedence $\Pi^*$ that approximately optimizes the cumulative preference value using a greedy algorithm proposed by Cohen et al. [1]. Essentially, the algorithm always looks for a predicate $p$ such that $\sum_{p' \in \mathit{Remaining}} \mathit{pref}(p, p')$ is the smallest/largest and moves such $p$ from *Remaining* to the next position in the (from left to right) constructed precedence.

To summarize the initial experiment design, we evaluate the feasibility of precedence learning from pairwise preference values, trying to answer the following question:

> To what extent is it possible to construct a good predicate precedence if we know a good preference value for each pair of predicate symbols?

In the talk, we will present encouraging experimental results showing that when using the averaged pairwise preference values the greedy algorithm from Cohen et al. [1] reaches or surpasses the performance of Vampire's equivalent of E's `invfreq` heuristic on 85 out of 113 problems

from TPTP [15] (we selected problems that exhibit an especially high variation in number of saturation loop iterations), reducing the number of saturation loop iterations by an average (geometric mean) factor of 0.47.

## References


[1] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *CoRR*, abs/1105.5464, 2011. URL http://arxiv.org/abs/1105.5464.

[2] Samuel N. Kamin and Jacques Lévy. Two generalizations of the recursive path ordering. 1980.

[3] D. E. Knuth and P. B. Bendix. *Simple Word Problems in Universal Algebras*, pages 342–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. ISBN 978-3-642-81955-1. doi: 10.1007/978-3-642-81955-1_23. URL https://doi.org/10.1007/978-3-642-81955-1_23.

[4] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 1–35, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39799-8. URL http://www.vprover.org/cav2013.pdf.

[5] Laura Kovács, Georg Moser, and Andrei Voronkov. On transfinite knuth-bendix orders. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*, pages 384–399. Springer, 2011. ISBN 978-3-642-22437-9. doi: 10.1007/978-3-642-22438-6_29. URL https://doi.org/10.1007/978-3-642-22438-6_29.

[6] Daniel Kühlwein and Josef Urban. Males: A framework for automatic tuning of automated theorem provers. *J. Autom. Reasoning*, 55(2):91–116, 2015. doi: 10.1007/s10817-015-9329-1. URL https://doi.org/10.1007/s10817-015-9329-1.

[7] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning (in 2 volumes)*, pages 371–443. 2001.

[8] Miroslav Olšák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. *CoRR*, 2019.

[9] Aditya Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. Graph representations for higher-order logic and theorem proving. *CoRR*, abs/1905.10006, 2019. URL http://arxiv.org/abs/1905.10006.

[10] Giles Reger and Martin Suda. Measuring progress to predict success: Can a good proof strategy be evolved? In *AITP 2017*, 2017.

[11] Giles Reger, Martin Suda, and Andrei Voronkov. New techniques in clausal form generation. In Christoph Benzm\"uller, Geoff Sutcliffe, and Raul Rojas, editors, *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41 of *EPiC Series in Computing*, pages 11–23. EasyChair, 2016. doi: 10.29007/dzfz. URL https://easychair.org/publications/paper/XncX.

[12] G. Robinson and L. Wos. Paramodulation and theorem-proving in first-order theories with equality. In Jörg H. Siekmann and Graham Wrightson, editors, *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pages 298–313, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg. ISBN 978-3-642-81955-1. doi: 10.1007/978-3-642-81955-1_19. URL https://doi.org/10.1007/978-3-642-81955-1_19.

[13] 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.

[14] Daniel Selsam and Nikolaj Bjørner. Guiding high-performance SAT solvers with unsat-core predictions. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 336–353. Springer, 2019. ISBN 978-3-030-24257-2. doi: 10.1007/978-3-030-24258-9_24. URL https://doi.org/10.1007/978-3-030-24258-9_24.

[15] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[16] G. S. Tseitin. *On the Complexity of Derivation in Propositional Calculus*, pages 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. ISBN 978-3-642-81955-1. doi: 10.1007/978-3-642-81955-1_28. URL https://doi.org/10.1007/978-3-642-81955-1_28.

[17] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2786–2796, 2017. URL http://papers.nips.cc/paper/6871-premise-selection-for-theorem-proving-by-deep-graph-embedding.

[18] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks, 2019.