Usefulness of Lemmas via Graph Neural Networks *

Zarathustra Goertzel and Josef Urban

Czech Technical University, Prague,

Introduction: Lemmas in Math and ATPs

In mathematics a *lemma* is generally used to denote an intermediate result. A mathematician or student proves some lemmas and then uses these lemmas in the proof of a target theorem. Sometimes lemmas are useful in multiple theorems (such as Schur's Lemma[1]), sometimes they are didactic or make for a clearer proof. Automated theorem provers (ATPs) such as E prover [10] struggle with finding long proofs. There are various (sometimes handmade) tools [9, 8, 3, 5] to heuristically identify 'interesting' or 'meaningful' steps in ATP proofs and sometimes to better present them to humans. We believe that improved automated determination of lemmas in an ATP proof or proof-search will help ATPs prove more difficult theorems.

The first part of this research proposal describes how to measure the degree to which an E proof step is a *useful* lemma. The second part describes two attempts to solve the classification task and the reasons why we believe graph neural networks could be suitable for proof analysis.

Measuring the Usefulness of Proof Lemmas

Setting: Formally, we view lemmas as logical *cuts*. Given axioms Γ and a conjecture C we call L a lemma if $\Gamma \vdash L$ and $\Gamma, L \vdash C$ hold, which necessitate $\Gamma \vdash C$. However in saturation-style refutational clausal ATPs like E, proofs are of the form $\Gamma, \neg C \vdash \bot$. Therefore we use the law of excluded middle instead: $\Gamma, (L \vee \neg L) \vdash C$ allows us to split the proof of C into two sub-problems: $\Gamma, L \vdash C$ and $\Gamma, \neg L \vdash C$. This formulation likely leads to faster lemma proofs as many proof clauses in E are derived from the negated conjecture $\neg C$.[2]

Proof shortening ratio: To measure whether the introduction of a lemma L facilitates an ATP proof of C given Γ, we define a *proof shortening ratio* of L wrt. Γ and C, or shortly $psr(L, \Gamma, C)$. The definition uses the length of an ATP proof search, denoted as $|\Gamma \vdash C|$ [3] for each (sub)problem: $psr(L, \Gamma, C) = \frac{|\Gamma, L \vdash C| + |\Gamma, \neg L \vdash C|}{|\Gamma \vdash C|}$. If $psr(L, \Gamma, C) < 1$, then L can be said to help E prove C from Γ faster: L is a good lemma.

Dataset: We have created a dataset consisting of 3161 CNF problems from the Mizar40 dataset [4] and their E proofs. The proofs were all found by a fixed single E strategy. For each such E proof P of $\Gamma \vdash C$ and each of P's proof clauses L_i^P, we introduce L_i^P as a lemma and run E on both subproblems $\Gamma, L_i^P \vdash C$ and $\Gamma, \neg L_i^P \vdash C$, recording $psr(L_i^P, \Gamma, C)$.

There are 230528 proof clauses in total, of which 98472 are axioms and negated conjectures. There are 44895 positive examples [4] and 87161 negative examples

*Supported by the ERC Consolidator grant no. 649043 AI4REASON and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/ 0000466 and the European Regional Development Fund.

[1] https://en.wikipedia.org/wiki/Schur's_lemma

[2] Another technical reason for this formulation is to avoid dealing with the skolem symbols from $\neg C$ in L.

[3] We use the number of given clause loops in the E proof.

[4] Good lemmas, such that $psr(L, \Gamma, C) < 1$ and 154 with $psr(L, \Gamma, C) = 1$ (we treat them as positives too).

Lemma Classification

Ideally one wants to generate hypothesized lemmas given Γ and C, or to scan an incomplete proof-search and identify lemmas to add to Γ. As a first step, we explore the problem of classifying existing E proof lemmas as good or bad. We use ENIGMA features [2] to represent the clauses in the dataset [5]. Because there are 219724 features, a Singular Value Decomposition is computed via the scikit-learn [7] library to reduce the dimensionality to 300. Moreover, only 15000 clauses are used so that the adjacency matrix can fit in GPU memory. As a baseline we use scikit-learn's support vector machine classifier *SVC*. To counteract the imbalanced dataset, positive examples are weighted at $2x$ negative examples.

Graph Attention Networks (GATs): Mathematics is full of graphs. Formulas can be represented as trees or as graphs [12]. Proofs are directed acyclic graphs. And mathematical theories consist of dependency graphs. We want to test the hypothesis that the graph structure of a proof will prove useful for identifying lemmas: what are L's parents and children, and via what inference rules are they derived? A proof-graph captures this structure. There are many recent developments in applying neural networks to graph data [6, 1]. Graph convolution operators transform a node's neighbors into the node's new features. The new idea in GATs[6] [11] is to use self-attention to learn how to weigh neighboring nodes depending on their feature vectors. We chose GATs for their expressive simplicity. The two primary parameters to tune are the number of attention heads at each layer and the number of hidden units, the size of the feature vector, at the next layer. In addition we added a dense tanh layer to the end of the network, which may help the network to do regression rather than classification. In the case of proofs, each graph attention layer expands the influence on lemmas' classification based on parents and descendants in the proof graph. A drawback is that the proof graph is represented just as an adjacency matrix. Inference steps defining edges are not multi-edges, and must be represented numerically. Formulas must be represented vectorially, and how to do this effectively is a burning research topic.

Results: Because the goal is to predict good lemmas, the results are analyzed in terms of positive examples only. Recall is the accuracy on good lemmas. Precision measures the degree to which only good lemmas are classified as good. F-score is defined by $2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$.

SVC achieves 0.45 precision and 0.64 recall with an F-score of 0.53 on the 10% test set. The best GAT setting used regression with 64 units in the tanh-layer, $[64, 32]$ for hidden-unit size and 3 attention heads per layer. This achieves 0.45 precision, 0.72 recall, and an F-score of 0.55. The total GAT accuracy is 55%, and total SVC accuracy is 74%. The precision means that 45% of predicted good lemmas are actually good.

Discussion and Future Directions The GATs so far do not perform much better than SVM. This could be due to the architecture, due to the problem difficulty, or due to the limitations of the ENIGMA features used in both experiments. Many lemmas have *psr* close to 1, which may make disambiguation hard.

The ML methods come close to achieving high recall with greater than 50% precision, which could be good enough for further ATP experiments. We also plan to experiment with similar datasets extracted directly from the human proofs in ITP libraries, where the lemma representation (e.g. in the declarative Jaskowski-style proof systems) may be quite different and less dependent on the particular ATPs and their strategies.

[5] Clauses are translated into fixed-length numeric vectors via (top-down-)oriented term-tree walks of length 3 as *features*. For example, a unit clause "$P(f(a,b))$" contains only features "(P,f,a)" and "(P,f,b)" (see [2, Sec. 3.2] for details).

[6] See http://petar-v.com/GAT/ for a better introduction.

References

[1] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, aglar Gülehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[2] Jan Jakubuv and Josef Urban. ENIGMA: efficient learning-based inference guiding machine. In Herman Geuvers, Matthew England, Osman Hasan, Florian Rabe, and Olaf Teschke, editors, *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, volume 10383 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 2017.

[3] Cezary Kaliszyk and Josef Urban. Learning-assisted theorem proving with millions of lemmas. *Journal of Symbolic Computation*, 69:109–128, 2015.

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

[5] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Lemmatization for stronger reasoning in large theories. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems - 10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21-24, 2015. Proceedings*, volume 9322 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2015.

[6] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *CoRR*, abs/1807.07984, 2018.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Petr Pudlák. Search for faster and shorter proofs using machine generated lemmas. In G. Sutcliffe, R. Schmidt, and S. Schulz, editors, *Proceedings of the FLoC'06 Workshop on Empirically Successful Computerized Reasoning, 3rd International Joint Conference on Automated Reasoning*, volume 192 of *CEUR Workshop Proceedings*, pages 34–52, 2006.

[9] Yury Puzis, Yi Gao, and Geoff Sutcliffe. Automated generation of interesting theorems. In Geoff Sutcliffe and Randy Goebel, editors, *FLAIRS*, pages 49–54. AAAI Press, 2006.

[10] Stephan Schulz. System description: E 1.8. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *LPAR*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[12] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2786–2796. Curran Associates, Inc., 2017.