Parental Guidance in E *

Zarathustra Amadeus Goertzel, Jan Jakubův, and Josef Urban

Czech Technical University in Prague, Prague

Parents and Clause Selection in E State-of-the-art automated theorem provers (ATP), such as E [13,14], Prover9 [10], and Vampire [11], are based on the saturation loop paradigm and the *given clause algorithm* [12]. The input problem in first-order logic is translated into a refutationally equivalent set of clauses. The ATP's search for a contradiction maintains two sets of clauses: *processed* (initially empty) and *unprocessed* (initially the input clauses). At each step, one unprocessed clause is selected (*given*), and all of the possible inferences with all the processed clauses are generated (typically using resolution, paramodulation, etc.), extending the unprocessed clause set. The selected clause is then moved to the processed clause set. An important invariant is that all mutual inferences among the processed clauses have been computed at each step.

The selection of the "right" given clause is known to be an important choice-point vital to the success of the proof search. E's strategies consist of *clause evaluation functions* that weigh and prioritize clauses for selection based on their symbols and properties. The ENIGMA systems [2–7] apply various machine learning methods to learn how to select effective given clauses from corpora of previous successful proof searches. Given clause selection does not give the ENIGMA system complete control over the inferred clauses because all inferences between the given clause and clauses in the processed clause set are computed. One reason this can be important is that the ENIGMA systems tend to perform best when run in cooperation with a strong E strategy where each selects half the clauses.

This talk discusses the implementation and experimentation of an ENIGMA system that can "judge children by their parents" to filter out unnecessary inferences between the given clause and processed clauses[1] [2]. It is hoped that pruning the children of irresponsible parents can improve E's performance by allowing clause selection ENIGMA models and E strategies greater focus.

Implementation There have been many versions of ENIGMA, and the latest is ENIGMA Anonymous [4], which uses as the underlying machine learning method either Graph Neural Networks or Gradient Boosted Decision Trees (GBDTs, implemented by LightGBM here) [1,9]. For the GBDTs, clauses are represented by fixed-length numerical vectors based on clause syntax trees and names are anonymized by replacing symbol names with their arities. The goal clauses and theory clauses (which include axioms) are merged to create the goal and theory fixed-length vectors, which represent the clause's context. The three are then concatenated to create the *feature vector*.

*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] The code can be found at https://github.com/zariuq/eprover/tree/parentalguidance_frozen.

[2] And a pre-print of the full paper can be found at https://arxiv.org/abs/2107.06750

Parental guidance can be generally defined as clause evaluation based on (the features of) the parents of the clause (and possibly also on the clause itself). Two methods are evaluated:

1. $\mathcal{P}_{\mathsf{fuse}}$ merges the feature vectors of the parent clauses into one vector, typically by simply adding the feature counts.

2. $\mathcal{P}_{\mathsf{cat}}$ concatenates the feature vectors of the parent clauses to preserve their information in full.

The resulting parent feature vector is concatenated with the goal and theory vectors to create the feature vector for parental guidance.

A GBDT based filter is inserted into E's given clause algorithm so that clauses generated by parents whose scores are below a chosen threshold do not get evaluated by E's clause selection heuristics. Because not all clauses are compatible to mate together, the clause's parents are only sent to the GBDT for evaluation after the clause has been generated and before simplifications are performed. This leverages E's efficient indexing. Because they have two parents, only clauses generated by paramodulation (which implements resolution in E) are evaluated by the parental guidance model. Filtered clauses are stored in the *freezer* set so that E can restore them if the unprocessed clause set becomes empty, which avoids impairing the completeness of the proof search.

Training The experiments are performed[3] on a large benchmark of 57 880 problems[4] originating from the Mizar Mathematical Library (MML) [8] exported to first-order logic by MPTP [15]. The data are split into 3 subsets[5]: (1) 52k problems for *training*, (2) 2896 problems for *development*, and (3) 2896 problems for final evaluation (*holdout*).

First, the *baseline* in this work, called $\mathcal{D}_{\mathsf{large}}$, is a clause selection ENIGMA Anonymous model that is trained over a dataset consisting of at most 3 proofs from ca. 36k problems in the *training* set. The model consists of 150 decision trees of depth 40 with 2048 leaves and is the model that performed best in some prior experiments. The training data for $\mathcal{D}_{\mathsf{large}}$ consists of clauses processed during a proof search: clauses appearing in the proof are labeled *positive* and other clauses are *negative*. When run on the *training* set, $\mathcal{D}_{\mathsf{large}}$ proves 28 495 problems with 30 seconds per problem.

To train parental guidance models, the parents of all generated clauses from the $\mathcal{D}_{\mathsf{large}}$ run on the *trraining* set are used. Two methods of classifying the good pairs of parents are considered:

1. $\mathcal{P}^{\mathsf{proof}}$ classifies parents of only the proof clauses as *positive* and all other generated clauses as *negative*.

2. $\mathcal{P}^{\mathsf{given}}$ classifies parents of all processed (selected) clauses as *positive* and the unprocessed generated clauses as *negative*.

The reasoning behind (2) is that if a clause is selected by a well-trained strategy, then it probably should not be filtered: the aim is to remove only the worst of the children.

In the $\mathcal{P}^{\mathsf{proof}}$ data, the *pos-neg ratio*, the ration of positive to negative clauses, is 1:192. This is is experimentally reduced.

[3] On a server with 36 hyperthreading Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz cores and 755 GB of memory.

[4] http://grid01.ciirc.cvut.cz/~mptp/1147/MPTP2/problems_small_consist.tar.gz

[5] http://grid01.ciirc.cvut.cz/~mptp/Mizar_eval_final_split

Evaluation The parental guidance models are evaluated in combination with $\mathcal{D}_{\mathsf{large}}$ for clause selection with 30 seconds per problem. First a series of grid searches is done on a 300 problem subset of the development set and then the results are compared with the baseline $\mathcal{D}_{\mathsf{large}}$ on the full *development* and *holdout* sets.

The following parameters are tested:

1. the *pos-neg reduction ratio* of negative to positive clauses,
2. the threshold by which to filter clauses,
3. the positive data classification scheme ($\mathcal{P}^{\mathsf{proof}}$ vs $\mathcal{P}^{\mathsf{given}}$),
4. the LightGBM parameters (number of trees, maximum leaves per tree, and maximum tree depth)
5. the parental feature vector creation method ($\mathcal{P}_{\mathsf{fuse}}$ vs $\mathcal{P}_{\mathsf{cat}}$)

The final results can be seen in Table 1. Only considering proof clauses as positive examples ($\mathcal{P}^{\mathsf{proof}}$) outperforms considering all selected clauses ()$\mathcal{P}^{\mathsf{given}}$), which is probably because the data is cleaner and includes no confusing clauses. The low thresholds among the best models indicate that parental guidance works best when only the most obviously irresponsible parents are filtered. The cost of mistakenly filtering a necessary clause is high. Concatenating parent clause features ($\mathcal{P}_{\mathsf{cat}}$) seems far superier to merging them ($\mathcal{P}_{\mathsf{fuse}}$). The improvement of 11.7%, num163 more problems than the baseline, seems highly promising.

model	threshold	solved (D)	solved (H)
$\mathcal{D}_{\mathsf{large}}$	-	1397	1390
$\mathcal{P}^{\mathsf{given}}_{\mathsf{fuse}}+\mathcal{D}_{\mathsf{large}}$	0.05	1411 (+1.0%)	1417 (+1.9%)
$\mathcal{P}^{\mathsf{proof}}_{\mathsf{fuse}}+\mathcal{D}_{\mathsf{large}}$	0.1	1489 (+6.6%)	1486 (+6.9%)
$\mathcal{P}_{\mathsf{cat}}+\mathcal{D}_{\mathsf{large}}$	0.05	1571 (+12.4%)	1553 (+11.7%)

Table 1: Final 30s evaluation on development (D), and holdout (H) compared with $\mathcal{D}_{\mathsf{large}}$.

References

[1] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[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] Zarathustra Goertzel, Jan Jakubuv, and Josef Urban. Enigmawatch: Proofwatch meets ENIGMA. In Serenella Cerrito and Andrei Popescu, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings*, volume 11714 of *Lecture Notes in Computer Science*, pages 374–388. Springer, 2019.

[4] Jan Jakubuv, Karel Chvalovský, Miroslav Olsák, Bartosz Piotrowski, Martin Suda, and Josef Urban. ENIGMA anonymous: Symbol-independent inference guiding machine (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th*

International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II, volume 12167 of *Lecture Notes in Computer Science*, pages 448–463. Springer, 2020.

[5] 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.

[6] Jan Jakubuv and Josef Urban. Enhancing ENIGMA given clause guidance. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 2018.

[7] Jan Jakubuv and Josef Urban. Hammering Mizar by learning clause guidance. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*, volume 141 of *LIPIcs*, pages 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

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

[9] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pages 3146–3154, 2017.

[10] 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.

[11] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

[12] Ross A. Overbeek. A new class of automated theorem-proving algorithms. *J. ACM*, 21(2):191–200, April 1974.

[13] 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.

[14] Stephan Schulz, Simon Cruanes, and Petar Vukmirovic. Faster, higher, stronger: E 2.3. 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 495–507. Springer, 2019.

[15] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.