Learning Reasoning Components

Karel Chvalovský[1], Jan Jakubův[1], Miroslav Olšák[2], and Josef Urban[1]

[1] Czech Technical University in Prague, Czechia
[2] University of Innsbruck, Austria

Introduction We describe two iterative algorithms that combine high-level proof state evaluation and strategic reasoning decisions with guided low-level saturation-style proof search. For each part, we learn the tasks using an efficient logic-aware graph neural network [14] (GNN) recently integrated [8] into the ENIGMA [9, 4] guidance system of E [16, 17]. The general motivation is to explore and develop more human-like reasoning architectures, i.e., systems combining various (Malarious [20]) AI components and learning/reasoning feedback loops, which are (preferably) also competitive with ATPs in resource-controlled large-theory settings.[1]

GNN-ENIGMA The novelty previously introduced by GNN-ENIGMA compared to other saturation provers is that the generated clauses are not ranked immediately and independently of other clauses. Instead, they are judged by the GNN in larger batches and with respect to a large number of already selected clauses – the *context*. The GNN estimates collectively the most useful subset of the context and new clauses (*queries*) by several rounds of message passing, which sees the connections between symbols, terms, literals, and clauses. The GNN is trained on many previous proof searches, estimating which clauses will work together best.

Leapfrogging Our first method implements the idea that the graph-based evaluation of a particular clause may significantly change as new clauses are produced and the context changes. It corresponds to the human-based mathematical exploration, in which initial actions can be done with relatively low confidence and following only uncertain hunches. After some amount of initial exploration is done, clearer patterns often appear, allowing re-evaluation of the approach, focusing on the most promising directions, and discarding of less useful ideas. In tableau-based provers such as leanCoP [15] with a compact notion of a *state*, such methods can be approximated in a reinforcement learning setting by the notion of *big steps* [12] in the Monte-Carlo tree search (MCTS), implementing the standard *explore/exploit* paradigm [7]. In the saturation setting, our proposed algorithm uses short standard saturation runs in the (low-level) exploration phase, after which the set of processed (selected) clauses is re-evaluated and a (high-level, strategic) decision on its most useful subset is made by the GNN. These two phases are iterated in a procedure that we call *leapfrogging*.

In more detail, given a problem consisting of a set of initial clauses $S = S_0$, a saturation-style search (E/ENIGMA) is run on S with an abstract time limit. We may use a fixed limit (e.g., 1000 nontrivial processed clauses) for all runs, or change (e.g. increase) the limits gradually. If the initial run results in a proof or saturation within the limit, the algorithm is finished. If not, we inspect the set of created clauses. We can inspect all generated clauses, or a smaller set, such as the set of all processed clauses. So far, we used the latter because it is typically much smaller and better suits our training methods. This (*large*) set is denoted as L_0. Then we apply a trained graph-based predictor to L_0, which selects a smaller *most promising* subset of L_0, denoted as S_1. We may or may not automatically include also the initial negated conjecture clauses or the whole initial set S_0 in S_1. S_1 is then used as an input to the next limited saturation run of E/ENIGMA. This process is iterated, producing gradually sets S_i and L_i.

[1]Examples of such fair AI-style settings are the global resource limits used in the MPTP Challenge [13] and CASC LTB [18]. Similarly for large ITP benchmarks, e.g., Mizar [11], Flyspeck [10], HOL4 [3], and Isabelle [2].

Table 1: Four leapfrogging runs with different GNN-ENIGMAs

GNN-strategy	original-60s-run	leapfrogging (300-500-60s)	union	added-by-lfrg
G_1	2711	2218	3370	659
G_2	2516	2426	3393	877
G_3	2655	2463	3512	857
G_4	2477	2268	3276	799

Learning Reasoning Components Mathematical problems often have well-separated reasoning and computational components. Examples include numerical calculations, computing derivatives and integrals, performing Boolean algebra in various settings, sequences of standard rewriting and normalization operations in various algebraic theories, etc. Such components of the larger problem can be often solved mostly in isolation from the other components, and only their results are then used together to connect them and solve the larger problem. Human-designed problem solving architectures addressing such decomposition include, e.g., SMT systems, systems such as MetiTarski [1], and a tactic-based learning-guided proof search in systems such as TacticToe [6]. In all these systems, the component procedures or tactics are however *human-designed* and (often painstakingly) human-implemented, with a lot of care both for the components and for the algorithms that merge their results. This seems hard to scale to the large number of complex algorithms and heuristics used in research-level mathematics.

We instead want to *learn* such *targeted components*, expressed as sets of clauses that perform targeted reasoning and computation within the saturation framework.[2] We also want to learn the merging of the results of the components automatically. This is ambitious, but there seems to be growing evidence that such targeted components are being learned in many iterations of GNN-guided proving followed by retraining of the GNNs in our recent large iterative evaluation over Mizar.[3] In these experiments we have significantly extended our previously published results [8],[4] eventually automatically proving 73.5% (more than 40k) of the Mizar theorems. In particular, there are many examples on the project's Github page showing that the GNN is learning to solve more and more involved computations in problems involving differentiation, integration, boolean algebra, algebraic rewriting, etc. Our proposed *Split and Merge* algorithm is therefore to (i) use the GNN to learn to identify interacting reasoning components, (ii) use graph-based and clustering-based algorithms to split the set of clauses into components based on the GNN predictions, (iii) run saturation on the components independently, (iv) possibly merge the most important parts of the components, and (v) iterate. In more detail, we use a modified version of our GNN to predict the graph of future clause interactions in (i), experiment with several (soft) clustering methods for (ii), and again use the GNN to implement (iv).

Experiments The first leapfrogging experiment uses increasing limits on the set of processed clauses (300 and 500) with the final run limited by CPU time (60s). This is done over 28k hard Mizar problems with four differently parameterized GNNs (G_1, ..., G_4). The methods indeed achieve high complementarity to the original GNN strategies (Table 1), likely thanks to the different context in which the GNNs see the clauses in the subsequent runs. In the first Split and Merge experiment, we get 111 newly solved problems in the Split (component) phase out of 3000 hard problems unsolved in the initial standard saturation run (both using a limit of 1000 processed clauses). Then the Merge phase yields (with the same limit) additional 66 problems.[5] Many of the new proofs indeed show frequent computational patterns (see Appendix A).

[2]Note that this is also *fuzzier* (and possibly easier) than related splitting by *crisp* (neural) conjecturing [5, 19].
[3]https://github.com/ai4reason/ATP_Proofs
[4]The publication of this large evaluation is in preparation.
[5]The full experimental details will be given in the talk.

References

[1] Behzad Akbarpour and Lawrence C. Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *J. Autom. Reasoning*, 44(3):175–205, 2010.

[2] Jasmin Christian Blanchette, David Greenaway, Cezary Kaliszyk, Daniel Kühlwein, and Josef Urban. A learning-based fact selector for isabelle/hol. *J. Autom. Reason.*, 57(3):219–244, 2016.

[3] Chad E. Brown, Thibault Gauthier, Cezary Kaliszyk, Geoff Sutcliffe, and Josef Urban. GRUNGE: A grand unified ATP challenge. 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 123–141. Springer, 2019.

[4] 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.

[5] Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Initial experiments with statistical conjecturing over large formal corpora. In Andrea Kohlhase, Paul Libbrecht, Bruce R. Miller, Adam Naumowicz, Walther Neuper, Pedro Quaresma, Frank Wm. Tompa, and Martin Suda, editors, *Joint Proceedings of the FM4M, MathUI, and ThEdu Workshops, Doctoral Program, and Work in Progress at the Conference on Intelligent Computer Mathematics 2016 co-located with the 9th Conference on Intelligent Computer Mathematics (CICM 2016), Bialystok, Poland, July 25-29, 2016*, volume 1785 of *CEUR Workshop Proceedings*, pages 219–228. CEUR-WS.org, 2016.

[6] Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. TacticToe: Learning to reason with HOL4 tactics. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 125–143. EasyChair, 2017.

[7] John C Gittins. Bandit processes and dynamic allocation indices. *J. the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[8] 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.

[9] 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.

[10] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *J. Autom. Reasoning*, 53(2):173–213, 2014.

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

[12] Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8836–8847, 2018.

[13] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Machine learner for automated reasoning 0.4 and 0.5. In Stephan Schulz, Leonardo de Moura, and Boris Konev, editors, *4th Workshop on Practical Aspects of Automated Reasoning, PAAR@IJCAR 2014, Vienna, Austria, 2014*, volume 31 of *EPiC Series in Computing*, pages 60–66. EasyChair, 2014.

[14] Miroslav Olšák, Cezary Kaliszyk, and Josef Urban. Property invariant embedding for automated reasoning. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 1395–1402. IOS Press, 2020.

[15] Jens Otten and Wolfgang Bibel. leanCoP: lean connection-based theorem proving. *J. Symb. Comput.*, 36(1-2):139–161, 2003.

[16] Stephan Schulz. E - A Brainiac Theorem Prover. *AI Commun.*, 15(2-3):111–126, 2002.

[17] 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.

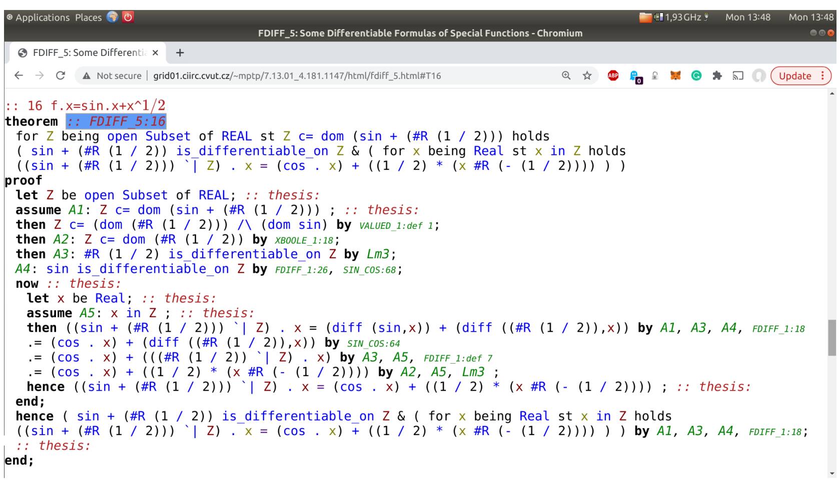
[18] Geoff Sutcliffe. The 4th IJCAR automated theorem proving system competition - CASC-J4. *AI Commun.*, 22(1):59–72, 2009.

[19] Josef Urban and Jan Jakubuv. First neural conjecturing datasets and experiments. In Christoph Benzmüller and Bruce R. Miller, editors, *Intelligent Computer Mathematics - 13th International Conference, CICM 2020, Bertinoro, Italy, July 26-31, 2020, Proceedings*, volume 12236 of *Lecture Notes in Computer Science*, pages 315–323. Springer, 2020.

[20] Josef Urban, Geoff Sutcliffe, Petr Pudlák, and Jiří Vyskočil. MaLARea SG1 - Machine Learner for Automated Reasoning with Semantic Guidance. In *IJCAR*, pages 441–456, 2008.

Appendix A Interesting Frequent Computational Patterns

Here we show three of the computationally looking Mizar proofs found automatically only by the Split and Merge algorithm for theorems `T16_FDIFF_5`,[6] `T48_NEWTON`,[7] and `T10_MATRIX_4`.[8]




```
:: 16 f.x=sin.x+x^1/2
theorem :: FDIFF_5:16
 for Z being open Subset of REAL st Z c= dom (sin + (#R (1 / 2))) holds
 ( sin + (#R (1 / 2)) is_differentiable_on Z & ( for x being Real st x in Z holds
 ((sin + (#R (1 / 2))) `| Z) . x = (cos . x) + ((1 / 2) * (x #R (- (1 / 2)))) ) )
proof
 let Z be open Subset of REAL; :: thesis:
 assume A1: Z c= dom (sin + (#R (1 / 2))) ; :: thesis:
 then Z c= (dom (#R (1 / 2))) /\ (dom sin) by VALUED_1:def 1;
 then A2: Z c= dom (#R (1 / 2)) by XBOOLE_1:18;
 then A3: #R (1 / 2) is_differentiable_on Z by Lm3;
 A4: sin is_differentiable_on Z by FDIFF_1:26, SIN_COS:68;
 now :: thesis:
  let x be Real; :: thesis:
  assume A5: x in Z ; :: thesis:
  then ((sin + (#R (1 / 2))) `| Z) . x = (diff (sin,x)) + (diff ((#R (1 / 2)),x)) by A1, A3, A4, FDIFF_1:18
  .= (cos . x) + (diff ((#R (1 / 2)),x)) by SIN_COS:64
  .= (cos . x) + (((#R (1 / 2)) `| Z) . x) by A3, A5, FDIFF_1:def 7
  .= (cos . x) + ((1 / 2) * (x #R (- (1 / 2)))) by A2, A5, Lm3 ;
  hence ((sin + (#R (1 / 2))) `| Z) . x = (cos . x) + ((1 / 2) * (x #R (- (1 / 2)))) ; :: thesis:
 end;
 hence ( sin + (#R (1 / 2)) is_differentiable_on Z & ( for x being Real st x in Z holds
 ((sin + (#R (1 / 2))) `| Z) . x = (cos . x) + ((1 / 2) * (x #R (- (1 / 2)))) ) ) by A1, A3, A4, FDIFF_1:18;
 :: thesis:
end;
```

Figure 1: Differentiation – T16_FDIFF_5

[6] http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/fdiff_5.html#T16
[7] http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/newton.html#T48
[8] http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/matrix_4.html#T10

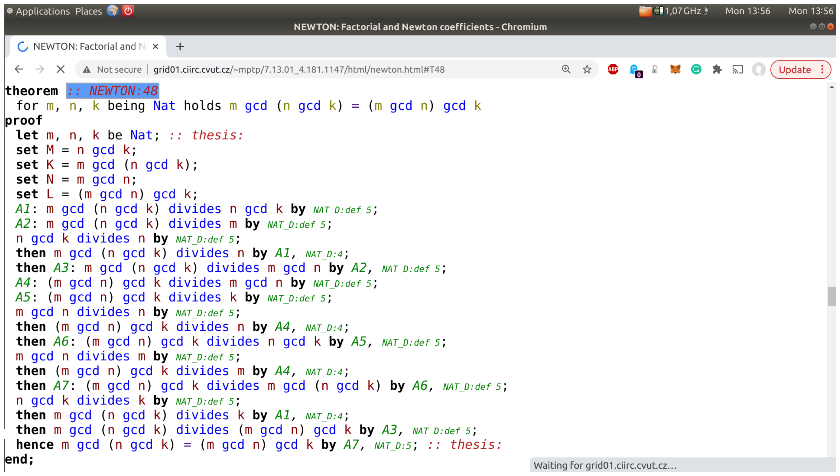


Figure 2: Associativity of gcd by many rewrites – T48_NEWTON

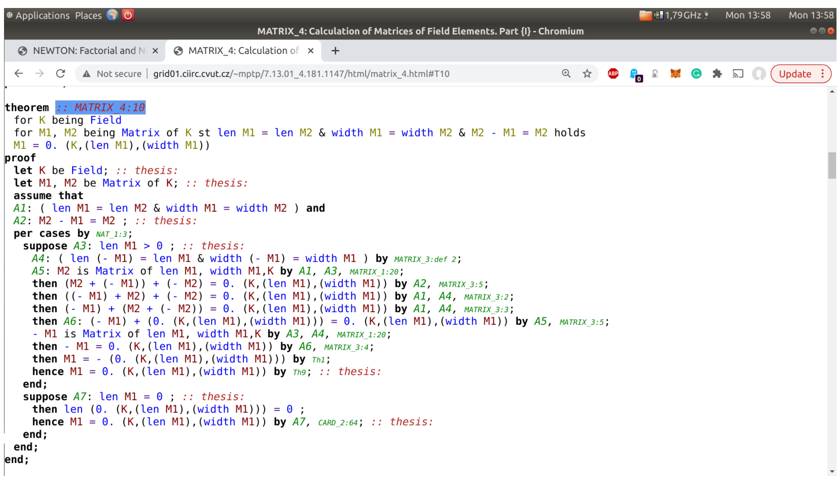


Figure 3: Matrix manipulation – T10_MATRIX_4

Acknowledgments

Supported by the ERC Consolidator grant *AI4REASON* no. 649043 (JJ, JU), by the Czech project AI & Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund (KC, JU), by the ERC Starting grant *SMART* no. 714034 (JJ, MO), and by the Czech MEYS under the ERC CZ project *POSTMAN* no. LL1902 (JJ).