



# Neural Conjecturing for Saturation Theorem Provers

Karel Chvalovský<sup>1</sup>, Martin Suda<sup>1</sup>, and Josef Urban<sup>2,3</sup>

<sup>1</sup> Czech Technical University in Prague

<sup>2</sup> AI4REASON

<sup>3</sup> University of Gothenburg / Chalmers University

## Abstract

We present a system for generating intermediate lemmas, used as cut formulas, that restructure first-order proof search in saturation-based theorem provers. Given a CNF problem, a heterogeneous graph neural network encodes clauses, literals, terms, variables and symbols. An autoregressive decoder then samples candidate lemmas by pointing to symbols in the input and by obeying arity constraints during decoding. Candidate lemmas are accepted when an ATP proves both sides of the cut, i.e. the problem extended with the lemma and with its negation. Together with adjustable criteria for proof-search speedup this gives a supervised learning signal and a learning-proving feedback loop.

We train and evaluate the methods on ATP datasets built from the Mizar Mathematical Library with both E and Vampire. On the E test set, the best single configuration helps 96 of 187 baseline-proved problems, while an ensemble helps 127 (67.9%), with speedups up to 196×. We also show that a cut generator trained on the E data transfers to Vampire: when the cuts are delivered as AVATAR *claims*, Vampire proves 501 hard Mizar problems unsolved by Vampire or E in the Mizar60 evaluation, and speeds up 1567 further problems. In combination with trained Deepire and Vampire’s CASC mode, the learning/conjecturing/portfolio methods discussed here prove 1228 Mizar problems unsolved by Vampire or E in the Mizar60 evaluation.<sup>1</sup>

## 1 Introduction

Conjecturing—the creation of useful intermediate statements—is a core mathematical ability that complements proof search. Humans routinely decompose hard problems into lemmas, and the ability to propose good lemmas is often what separates a feasible proof from an intractable search. Automated conjecturing has been studied since Lenat’s AM [21], with systems such as Fajtlowicz’s Graffiti [5] and Colton’s HR [3] exploring theory formation via heuristics or controlled enumeration. More recently, Urban and Jakubův [31] proposed using neural sequence models to generate conjectures over large formal libraries. This has been followed by many neural conjecturing systems done in various ITP contexts.

In the context of saturation-based refutational theorem proving, conjecturing takes a specific form: the classical cut rule. A formula  $L$  splits a hard problem  $P$ , which already contains the

---

<sup>1</sup>The development of the neural architectures and the paper writing has used Claude Code and Codex in multiple human-supervised sessions. The experiments and evaluations were conducted by humans.

negated goal, into two potentially easier problems,  $P \cup \{L\}$  (“use the lemma”) and  $P \cup \{\neg L\}$  (“prove the lemma from  $P$ ”). If both are unsatisfiable, then  $P$  is unsatisfiable. Modern provers such as E [26] and Vampire [20] combine many sophisticated search heuristics, yet a wrong trajectory may lead to millions of inferences before finding a short refutation. A well-chosen cut can bypass such bottlenecks entirely. The formula  $L$  need not be a user-interesting statement; it only has to expose a useful case split.

This paper presents a system for automated cut discovery. We train a neural generator on proof lemmas that have been measured to shorten proof search, then use the ATP to evaluate the generated candidates. The approach is attractive for theorem proving because the learned part is entirely speculative: a bad lemma may waste time, but cannot make the prover unsound. The generated lemmas could also be used as proof-search hints [32].

In more detail, our pipeline is as follows:

1. Solve many related library problems and extract proof lemmas;
2. Evaluate each lemma as a cut and keep only speedups;
3. Train a graph-to-clause neural generator on the useful cuts;
4. Sample diverse lemmas for new problems; and
5. Use them to split the ATP problems, or add them as AVATAR *claim* formulas.

The code is developed online at our public repository.<sup>2</sup>

Our main contributions are as follows. We develop, train and evaluate with E several symbol-independent graph-to-clause architectures for first-order cut formula generation. We then train Deepire on the relevant Mizar problems and prove without conjecturing (i.e., no lemmas) 620 hard problems previously unsolved by any saturation-based ATP in the Mizar60 evaluation [15]. Finally, we generate lemmas in two iterations of training and evaluation and show that when added as AVATAR claims they help Vampire solve hundreds of additional hard Mizar problems.

## 2 Learning Cuts from Proofs

We use the framework for measuring lemma usefulness as ATP proof shortening introduced by Goertzel and Urban [8]. In particular, they created a dataset of 3161 Mizar CNF problems with cut ratios which we start with here.<sup>3</sup> For a problem  $P$  and candidate lemma  $L$ , let  $C_0$  be the baseline cost of proving  $P$ , and let  $C_1, C_2$  be the costs of proving  $P \cup \{L\}$  and  $P \cup \{\neg L\}$ . We define the cut ratio

$$r = \frac{C_1 + C_2}{C_0}.$$

Lemma  $L$  is useful for  $P$  when both subproofs succeed and  $r < 1$ . The cost is measured as processed clauses for E and as burned CPU instructions for Vampire. This is the key supervision signal: useful cuts are not labeled by humans, but by measuring ATP search reduction.

**AVATAR claims:** Running two separate proofs is expensive. Vampire’s AVATAR architecture [33] can instead introduce  $L$  through the TPTP role claim [4]. Its SAT layer creates complementary assumptions corresponding to  $L$  and  $\neg L$ , interleaves the two branches, and shares clauses derived without the claim. A proof is complete when the use of  $L$  and the proof

<sup>2</sup>[https://github.com/JUrban/E\\_conj\\_dev1/tree/audit-fixes6](https://github.com/JUrban/E_conj_dev1/tree/audit-fixes6)

<sup>3</sup>The original dataset and code: [https://github.com/JUrban/E\\_conj](https://github.com/JUrban/E_conj)

Table 1: Data and evaluation corpora. Useful means both cut branches prove and  $r < 1$ . Following [8], the E count also includes 154 examples with  $r = 1$ .

| Corpus                 | Problems | Solved | Evaluated pairs | Useful cuts |
|------------------------|----------|--------|-----------------|-------------|
| E minimized CNF        | 3161     | 1888   | 122 356         | 44 895      |
| Vampire full-axiom CNF | 57 880   | 34 182 | 2 470 292       | 196 199     |

Table 2: Quality of the 196 199 useful Vampire cut examples.

| $r \leq$ | Examples | Problems | % examples | % problems |
|----------|----------|----------|------------|------------|
| 0.10     | 3190     | 164      | 1.6        | 4.6        |
| 0.20     | 15 165   | 553      | 7.7        | 15.4       |
| 0.50     | 66 461   | 1645     | 33.9       | 45.8       |
| 0.80     | 135 380  | 2783     | 69.0       | 77.5       |
| 0.95     | 179 059  | 3371     | 91.3       | 93.9       |
| 1.00     | 196 199  | 3591     | 100.0      | 100.0      |

of  $L$  have both been discharged or when the original problem has been proved without using either. This lets us test generated lemmas speculatively in a single Vampire run, and also permits several generated claims to be added to the same search.

**Datasets:** The E development data uses 3161 post-proof-minimized Mizar CNF problems exported in TPTP style [29]: E solves 1888 of them and produces 122 356 evaluated (problem, lemma) pairs. The larger Vampire data uses the Deepire 2.0 system [28], which trains GNN-based clause selection inside Vampire by an iterative train/eval loop. A non-AVATAR strategy was first obtained by strategy-space hill climbing on a subset of the Mizar40 corpus (3000 problems, instruction limit 3000 Mi) and then used with Deepire to train clause guidance on all 57 880 Mizar40 CNF problems (instruction limit 16 000 Mi, 18 iterations, solving 40.5 %  $\rightarrow$  57.7 %). We use this model to extract proof lemmas. For the AVATAR-based evaluation of generated lemmas, a separate Deepire model was trained with AVATAR enabled, first on a random half of Mizar40 (20 iterations, 42.1 %  $\rightarrow$  62.0 %), and then on the premise-selected Mizar60 [15] corpus (22 iterations, 60.1 %  $\rightarrow$  70.6 % on 27 916 problems).

When Vampire uses AVATAR [33], the clauses it derives in general depend on splits (propositional variables representing clauses themselves) and are thus logically equivalent only to more complex formulas. That is why we decided, in order to keep the lemma extraction simple, to employ a non-AVATAR strategy there. The non-AVATAR model solves 34 182 of 57 880 full-axiom problems, from which 2.47 million proof-lemma pairs are extracted and evaluated; 196 199 are useful cuts and 3591 problems have at least one useful lemma.

We also evaluate on the premise-selected Mizar60 corpus of 55 832 problems, whose median CNF size is 34 clauses after clausification, giving a more realistic middle ground between post-proof minimized and full-axiom problems. Table 1 summarizes the main corpora. The useful cuts are highly skewed in quality: on the Vampire corpus (detailed in Table 2), 66 461 examples give at least a  $2\times$  speedup and 3190 give at least a  $10\times$  speedup. We keep all useful examples during training, but loss weighting makes the high-speedup tail more influential.

### 3 The Generator

We use a standard graph neural network (GNN) to prepare the input problem for the followup lemma generator. The GNN transforms a graph representation of the problem’s CNF into vectorial representations of the signature symbols on which then a neural generator builds.

**Problem graph:** The input CNF is represented as a heterogeneous graph with five node types: clause, literal, term, variable, and symbol. Sixteen directed edge types encode containment and syntax, including clause-to-literal, literal-to-argument, term-to-subterm and reverse edges. Four message passing layers are applied. Positional edges use edge-aware GINE convolutions [14]; the remaining edges use GraphSAGE-style aggregation [10]. The final symbol-node embeddings represent both the memory over which the decoder attends and the objects to which it may point.

**Why point to symbols:** Unlike a standard transformer (or more generally, a language model), the generator does not emit symbol names from a fixed vocabulary. Predicates and functions are selected by a pointer distribution over symbol nodes in the current problem. Thus the model can be applied to problems containing symbols not seen in training: it relies on arity, role, and occurrence structure rather than on names alone. This general *anonymous* setting has been introduced in ATP by [23] and used in various later systems such as [24]. Optional named Mizar embeddings are here added for recurring symbols, but unknown symbols still remain generatable through their graph embeddings.

**Decoder and syntax constraints:** The decoder generates a clause as actions: NEWLIT-POS, NEWLITNEG, PRED, ARGVAR, ARGFUNC, ENDARGS, and ENDCLAUSE. At each step it predicts an action, a symbol pointer when the action needs a predicate or function, and a variable slot when the action needs a variable. Role masks prevent predicates from being used as functions and conversely. An arity stack records how many arguments remain open; actions inconsistent with the stack are masked to  $-\infty$  before sampling. This simple state machine is responsible for achieving high syntactic validity in the experiments, compared to earlier simpler unconstrained generation.

**Training and variants:** Our loss is the sum of cross-entropies for action, symbol pointer and variable prediction, ignoring irrelevant outputs at each step. Samples are weighted by  $1/(1 + r)$  so that stronger speedups matter more, and per-problem normalization prevents a few lemma-rich theorems from dominating training. We experiment with training Transformer (Trans), conditional Variational Autoencoder (CVAE) and state-space decoder variants inspired by selective state-space models [9] (SSM), varying dimension, named-symbol embeddings and sampling temperature. Generation uses nucleus sampling [6, 13] with role masks and removes duplicate clauses before ATP evaluation.

**Engineering:** The implementation has two performance bottlenecks: graph construction and batching. For the Vampire data, a single problem may produce thousands of graph nodes, and the heterogeneous GNN applies a separate convolution per edge type. We therefore precompute graph samples, cache them in a compact tensor format, and batch by a total-node budget rather than by a fixed sample count. A batch of small problems may contain more than 100 samples, while a batch of large problems may contain only a few; the loss is normalized by token count,

so the objective is invariant to this packing. Persistent data-loader workers and mixed precision keep the GPUs<sup>4</sup> saturated enough for experiments at the 100K-sample scale.

## 4 Experiments

All reported successes are prover-verified. The generator proposes formulas, but a result is counted only when the corresponding ATP proof succeeds. For E, we test each generated lemma  $L$  by two independent 2-second runs for  $P \cup \{L\}$  and  $P \cup \{\neg L\}$ . For Vampire, we use AVATAR claims with the Deepire 2.0 AVATAR-trained clause guidance model and an instruction limit of 100 000 Mi plus a 30-second wall-clock limit.

We used the relatively short limits for E to quickly understand which architecture choices lead to good performance. The later Vampire runs were meant to fathom the limits of the approach with much more generous resource budgets.

**Protocol details:** The baseline for a problem is always measured without generated lemmas, using the same prover and resource limits as in the corresponding experiment. A generated lemma is discarded before proving if it is syntactically invalid, uses symbols with incorrect roles, exceeds the arity-state limits, or duplicates an earlier sample modulo printing. For solved baseline problems we report speedup ratios; for unsolved baseline problems we report newly proved problems. This separates the two main benefits: cuts can either make an existing proof cheaper, or move a problem from unsolved to solved within a fixed budget. All aggregate counts are by problem, not by lemma, so a problem with many useful lemmas is counted once.

### 4.1 E Prover Results

The E test split contains 279 problems, of which 187 are solved by the 10-second baseline. We train 15 configurations and generate 40–80 lemmas per problem using nucleus sampling. Table 3 shows selected results. The best single configuration helps 96 of the 187 baseline-solved problems (51.3%). The union of all configurations helps 127 problems (67.9%), showing that diversity across models and sampling regimes is essential. The best individual speedup is  $196\times$  on `l101_xreal_1`; other large speedups include  $50\times$  on `t87_qc_lang2` and  $38\times$  on `t35_xbool_1`.

The useful lemmas usually occur early. Across representative models, the median rank of the first useful conjecture is between 2 and 5 (Table 5); evaluating only the first five lemmas already recovers a large fraction of helped problems. Wider sampling has worse average rank but better total coverage, which suggests a practical two-phase strategy: run a cheap top- $k$  filter across several diverse models, and spend the remaining ATP budget on configurations whose first successes are complementary. Table 4 shows the effect: the best single model covers 96 problems, but five carefully chosen methods cover 123, very close to the 127-problem oracle over all 15 methods.

### 4.2 Cross-Prover Transfer to Vampire

The central experiment asks whether cut patterns learned from one prover are useful to another. We take an E-trained  $d = 256$  model with named symbol embeddings and generate at most 50 lemmas for each of 11 881 hard or previously unsolved Mizar60 problems, producing 517 435 lemmas with 98.9% syntactic validity. The top 42 lemmas are given to Vampire as AVATAR claims, one per run, so that each can be safely speculatively tried out by the prover.

<sup>4</sup>Most of the final experiments use the Leonardo HPC cluster with A100 GPUs.

Table 3: Selected E results on 187 baseline-proved test problems. All configurations use named Mizar symbol embeddings.  $d$ : hidden dimension. Sampling:  $T$  is temperature,  $k/p$  are top- $k$ /top- $p$  nucleus parameters [6, 13]. AvgR is the best cut ratio averaged over helped problems; lower is better.

| Decoder                        | $d$  | Sampling                        | Helped | %    | AvgR  |
|--------------------------------|------|---------------------------------|--------|------|-------|
| Transformer                    | 256  | $T=1.5, k=25, p=.98$ (wide)     | 96     | 51.3 | 0.558 |
| Transformer                    | 256  | $T=1.2, k=15, p=.95$ (diverse)  | 91     | 48.7 | 0.574 |
| Transformer                    | 256  | $T=1.2, k=15, p=.95$ (balanced) | 87     | 46.5 | 0.524 |
| Transformer                    | 512  | $T=1.2, k=15, p=.95$ (diverse)  | 80     | 42.8 | 0.536 |
| CVAE                           | 256  | $T=1.2, k=15, p=.95$            | 78     | 41.7 | 0.597 |
| SSM                            | 256  | $T=1.2, k=15, p=.95$            | 72     | 38.5 | 0.612 |
| Transformer                    | 1024 | $T=1.2, k=15, p=.95$ (diverse)  | 67     | 35.8 | 0.530 |
| Union of all 15 configurations |      |                                 | 127    | 67.9 | 0.530 |

Table 4: Complementarity on the E test set. “Saved” is total baseline processed clauses saved by the best cut for each helped problem. (A) and (B) are two different Trans-256 diverse configurations.

| $N$ | Best combination by coverage | Helped | Saved |
|-----|------------------------------|--------|-------|
| 1   | Trans-256, wide              | 96     | 197K  |
| 2   | + SSM-256                    | 106    | 218K  |
| 3   | + Trans-256, diverse (A)     | 113    | 286K  |
| 4   | + Trans-256, balanced        | 119    | 287K  |
| 5   | + Trans-256, diverse (B)     | 123    | 292K  |
| 15  | oracle union                 | 127    | 301K  |

Table 6 gives the results. Without generated lemmas, the AVATAR-guided Vampire baseline proves 3738 of the 11 881 problems (including 620 from the hard “minsub” set that no prior saturation-based ATP had solved). The generated lemmas prove 582 additional problems, a 15.6% relative increase. The effect is concentrated on the hardest problems: 442 of the 582 newly proved are from the minsub set, nearly doubling the number of hard problems solved ( $620 \rightarrow 1062$ ). Among already solved problems, 1567 obtain at least a 10% search reduction (mean ratio 0.608).

This transfer is nontrivial. The generator was trained on E proof lemmas from post-proof minimized problems, but it helps Vampire on premise-selected problems with different clauses, guidance and proof search. The most plausible explanation is that the model has learned structural templates for useful case splits rather than prover-specific syntactic artifacts. The pointer architecture is important here: a fixed-vocabulary generator would be much less robust to new Mizar symbols, new premise selections, and the many autogenerated Skolem symbols introduced during clausification.

**Closing the loop:** The successful Vampire runs yield 24 025 additional useful examples. We use them to test whether conjecture generation can become an iterative prove-learn feedback loop. Training from scratch on these examples overfits quickly. Initializing from the E checkpoint is much better, and the best validation loss is obtained by mixing the original E cuts with

Table 5: Effect of evaluating only the top- $k$  lemmas for three representative E models.

| $k$ | Problems helped |            |           | Avg. best ratio |            |           |
|-----|-----------------|------------|-----------|-----------------|------------|-----------|
|     | Trans-256 wide  | Trans-1024 | Trans-512 | Trans-256 wide  | Trans-1024 | Trans-512 |
| 1   | 20              | 31         | 35        | 0.378           | 0.433      | 0.490     |
| 5   | 54              | 53         | 63        | 0.439           | 0.478      | 0.536     |
| 10  | 70              | 60         | 75        | 0.503           | 0.498      | 0.551     |
| 20  | 81              | 65         | —         | 0.518           | 0.529      | —         |
| 50  | 90              | 67         | —         | 0.551           | 0.530      | —         |

Table 6: Cross-prover transfer: E-trained lemmas evaluated with Vampire on hard or unsolved Mizar60 problems.

|                                              |         |
|----------------------------------------------|---------|
| Problems evaluated                           | 11 881  |
| Generated lemmas                             | 517 435 |
| Syntactically valid lemmas                   | 98.9 %  |
| Baseline proved by Vampire, no lemmas        | 3738    |
| of which hard                                | 620     |
| Previously unproved, solved with lemmas      | 582     |
| of which hard                                | 442     |
| Total proved (baseline + lemmas)             | 4320    |
| total hard problems proved                   | 1062    |
| Problems with at least 10 % speedup          | 1567    |
| Mean ratio on already-solved helped problems | 0.608   |
| Median rank of first useful conjecture       | 3       |

the new Vampire cuts (59K combined examples). The resulting  $d=256$  model generates 235 089 second-iteration conjectures (at most 20 per problem) for the same 11 881 problems with 99.3 % syntactic validity.

Vampire evaluation of the second iteration (Table 7) shows continued improvement. With 20 conjectures from a model trained on its own verified outputs, the second iteration adds 59 new hard problems beyond the first, bringing the cumulative total to **1121**—an 81 % increase over the baseline of 620. Figure 1 shows the number of hard problems solved per conjecture rank: even rank 20 still solves 356 hard problems, confirming the value of diverse sampling.

For comparison, Vampire’s CASC portfolio mode (30-second limit, multiple strategies with AVATAR) solves 3262 problems on the same benchmark, of which only 289 are hard—less than half the 620 solved by the single trained Deepire 2.0 strategy. Adding CASC to the conjecturing results is complementary: the full union reaches 4507 problems solved and **1228 hard problems**, roughly doubling the baseline.

## 5 Discussion and Threats to Validity

The experiments support three practical lessons. First, neural cut generation should be evaluated by downstream proof search, not by language-model likelihood. Many well-formed lemmas are irrelevant, and some low-probability samples create large speedups. Second, diversity among generated candidates matters. The best E result is obtained by combining several model configurations, and in the Vampire experiments useful claims occur at many positions in the sampled

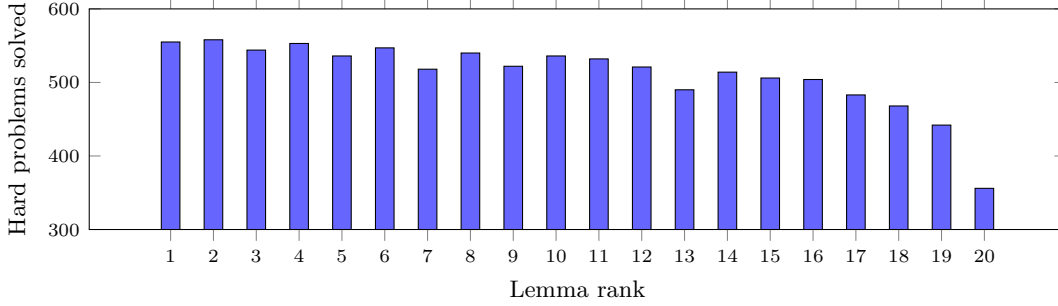


Figure 1: Hard (minsub) problems solved per conjecture rank in the second iteration. The baseline (no conjectures) solves 620. Each rank independently solves 356–558 hard problems; useful conjectures appear throughout, with a gradual decline for lower-ranked candidates.

Table 7: Cumulative results on 11 881 hard or unsolved Mizar60 problems.

| Configuration                    | Total       | Hard        |
|----------------------------------|-------------|-------------|
| Vampire CASC portfolio (30s)     | 3262        | 289         |
| Deepire AVATAR+neural (no conj.) | 3738        | 620         |
| + 1st iter. conj. (42, E-only)   | 4320        | 1062        |
| + 2nd iter. conj. (20, combined) | 4389        | 1121        |
| + CASC portfolio                 | <b>4507</b> | <b>1228</b> |

list. Third, speculative proving is a natural interface between learning and ATPs: Vampire claims let the prover benefit from a conjecture while simultaneously checking it.

The main threats are cost and selection bias. Cut evaluation is relatively expensive. Moreover, our datasets inevitably reflect the prover strategies used to generate and test the lemmas; different strategies might produce different useful cuts. The E data is small and post-proof minimized, while the full Vampire data is larger but harder to train on because of graph size. We mitigate these issues with per-problem weighting, size-aware batching and cross-prover evaluation, but final generality should be tested on more libraries and on fresh ATP strategies.

A second threat is budget accounting. A generated lemma can incur an impressive speedup once it is known, but a deployment must pay for all failed candidates as well. Our rank analysis partially addresses this by showing that useful cuts often appear early and that model ensembles can be scheduled greedily, but the current system still relies on ATP checking as the final ranker.

A learned utility model—similar to premise selectors—could help rank or filter candidates before ATP checking, but the final evaluation should still be based on downstream proof search.

## 6 Related Work

Cut introduction has a long proof-theoretic history beginning with Gentzen [7]; automated cut introduction for first-order logic with equality has been studied by Hetzl et al. [11]. Our work differs by generating cuts from a neural model and then checking them by ATP proof search. Machine learning for ATPs has most often focused on premise selection [30, 1, 19, 12] and proof or clause guidance [18, 16, 22, 17, 27, 25, 2]. Those methods choose what to search next;

our method changes the shape of the problem by adding auxiliary cut formulas. The closest antecedents are the cut-usefulness classifier of Goertzel and Urban [8] and the first neural conjecturing datasets of Urban and Jakubův [31]; the present system adds a symbol-pointer graph generator in the spirit of [24], arity-constrained decoding, large-scale Vampire cut data and cross-prover transfer.

## 7 Conclusion

We presented a neural system for generating cut formulas for saturation-based theorem provers. The generated formulas are speculative but soundly verified: a lemma counts only if the ATP proves both sides of the cut. On E, ensembles help 67.9% of baseline-proved test problems with speedups up to 196×. On Vampire, lemmas learned from E proofs help solving 442 additional hard Mizar problems (when run as claims under a trained neurally-guided AVATAR strategy), rising to 501 after a second iteration trained on combined E and Vampire cuts—nearly doubling the baseline of 620 hard problems solved by the trained neural Vampire strategy. Combined with Vampire’s CASC portfolio, the full union reaches 1228 hard problems. These results suggest that learned cut formulas capture reusable structure in mathematical proofs and that iterative conjecturing–proving–learning loops can progressively expand the reach of automated theorem provers.

## Acknowledgements

The work was supported by the Czech Science Foundation project 24-12759S, the Renaissance Philanthropy grant DEEPER, and the ERC grant NextReason (Grant Agreement No. 101200949).

## References

- [1] Jesse Alama, Tom Heskes, Daniel Kühlwein, Evgeni Tsivtsivadze, and Josef Urban. Premise selection for mathematics by corpus analysis and kernel methods. *J. Autom. Reason.*, 52(2):191–213, 2014.
- [2] Karel Chvalovský, Konstantin Korovin, Jelle Piepenbrock, and Josef Urban. Guiding an instantiation prover with graph neural networks. In Ruzica Piskac and Andrei Voronkov, editors, *LPAR 2023: Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Manizales, Colombia, 4-9th June 2023*, volume 94 of *EPiC Series in Computing*, pages 112–123. EasyChair, 2023.
- [3] Simon Colton. *Automated Theory Formation in Pure Mathematics*. Distinguished dissertations. Springer, 2002.
- [4] Sólrún Halla Einarsdóttir, Márton Hajdú, Moa Johansson, Nicholas Smallbone, and Martin Suda. Lemma discovery and strategies for automated induction. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part I*, volume 14739 of *Lecture Notes in Computer Science*, pages 214–232. Springer, 2024.
- [5] Siemion Fajtlowicz. On conjectures of graffiti. *Discret. Math.*, 72(1-3):113–118, 1988.
- [6] Angela Fan, Mike Lewis, and Yann N. Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 889–898. Association for Computational Linguistics, 2018.

- [7] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(1):176–210, 1935.
- [8] Zarathustra Goertzel and Josef Urban. Usefulness of lemmas via graph neural networks. In *4th Conference on Artificial Intelligence and Theorem Proving (AITP 2019), Obergurgl, Austria, April 7–12, 2019, Extended Abstracts*, 2019. [http://aitp-conference.org/2019/abstract/AITP\\_2019\\_paper\\_32.pdf](http://aitp-conference.org/2019/abstract/AITP_2019_paper_32.pdf).
- [9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023.
- [10] William L. Hamilton, Zhitaoying, and Jure Leskovec. Inductive representation learning on large graphs. 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, December 4–9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.
- [11] Stefan Hetzl, Alexander Leitsch, Giselle Reis, Janos Tapolczai, and Daniel Weller. Introducing quantified cuts in logic with equality. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19–22, 2014. Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2014.
- [12] Edvard K. Holden and Konstantin Korovin. Graph sequence learning for premise selection. *J. Symb. Comput.*, 128:102376, 2025.
- [13] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net, 2020.
- [14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net, 2020.
- [15] Jan Jakubuv, Karel Chvalovský, Zarathustra Amadeus Goertzel, Cezary Kaliszyk, Mirek Olšák, Bartosz Piotrowski, Stephan Schulz, Martin Suda, and Josef Urban. MizAR 60 for Mizar 50. In Adam Naumowicz and René Thiemann, editors, *14th International Conference on Interactive Theorem Proving, ITP 2023, Białystok, Poland, July 31 - August 4, 2023*, volume 268 of *LIPICs*, pages 19:1–19:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
- [16] 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.
- [17] Jan Jakubuv and Josef Urban. Hammering Mizar by learning clause guidance (short paper). In John Harrison, John O’Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving, ITP 2019, Portland, OR, USA, September 9–12, 2019*, volume 141 of *LIPICs*, pages 34:1–34:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [18] Cezary Kaliszyk and Josef Urban. Femalecop: Fairly efficient machine learning connection prover. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24–28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 88–96. Springer, 2015.
- [19] Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reason.*, 55(3):245–256, 2015.
- [20] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Confer-*

- ence, *CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.
- [21] Douglas B. Lenat. Automated theory formation in mathematics. In Raj Reddy, editor, *Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, USA, August 22-25, 1977*, pages 833–842. William Kaufmann, 1977.
  - [22] Sarah M. Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. 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 85–105. EasyChair, 2017.
  - [23] 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.
  - [24] Jelle Piepenbrock, Josef Urban, Konstantin Korovin, Miroslav Olšák, Tom Heskes, and Mikoláš Janota. Invariant neural architecture for learning term synthesis in instantiation proving. *J. Symb. Comput.*, 128:102375, 2025.
  - [25] Michael Rawson and Giles Reger. lazycop: Lazy paramodulation meets neurally guided search. In Anupam Das and Sara Negri, editors, *Automated Reasoning with Analytic Tableaux and Related Methods - 30th International Conference, TABLEAUX 2021, Birmingham, UK, September 6-9, 2021, Proceedings*, volume 12842 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2021.
  - [26] Stephan Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126, 2002.
  - [27] Martin Suda. Vampire with a brain is a good ITP hammer. In Boris Konev and Giles Reger, editors, *Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, volume 12941 of *Lecture Notes in Computer Science*, pages 192–209. Springer, 2021.
  - [28] Martin Suda. Efficient neural clause-selection reinforcement. In Clark W. Barrett and Uwe Waldmann, editors, *Automated Deduction - CADE 30 - 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings*, volume 15943 of *Lecture Notes in Computer Science*, pages 403–422. Springer, 2025.
  - [29] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *J. Autom. Reason.*, 43(4):337–362, 2009.
  - [30] Josef Urban. MaLAREa: a metasytem for automated reasoning in large theories. In Geoff Sutcliffe, Josef Urban, and Stephan Schulz, editors, *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories, Bremen, Germany, 17th July 2007*, volume 257 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
  - [31] 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.
  - [32] Robert Veroff. Using hints to increase the effectiveness of an automated reasoning program: Case studies. *J. Autom. Reason.*, 16(3):223–239, 1996.
  - [33] Andrei Voronkov. AVATAR: the architecture for first-order theorem provers. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer, 2014.