Neural Conjecturing for Saturation Theorem Provers

Jan Jakubuv[1], Miroslav Olšák[2], Martin Suda[1], and Josef Urban[3]

[1]Czech Technical University in Prague [2]Charles University in Prague [3]AI4REASON and University of Gothenburg

Abstract

We study neural generation of useful intermediate lemmas for saturation-style automated theorem provers as defined in [1]. Given a first-order problem in clausal normal form, the system[1] represents the problem as a heterogeneous graph, encodes it with a graph neural network, and generates candidate lemmas as clauses using pointer-based autoregressive decoders over the symbols occurring in the input problem. The generated clauses are not premises selected from a library: they are new conjectures, evaluated by E as cuts. On 3,161 Mizar40 problems with 122,356 extracted and evaluated proof cuts, the best configurations generate conjectures of which about 61% are provable intermediate lemmas and about 5–6% yield genuine proof-search speedups; the largest observed reduction is 77×. We have extensively used OpenAI Codex for coding the neural architectures.

Cut evaluation. For a CNF refutational problem P and an arbitrary clause (conjecture/lemma/cut) C, we run two refutational ATP tasks (i.e., both search for contradiction). P_1 tries to refute the original problem by extending it with C to $P \cup \{C\}$, testing whether C helps. P_2 tries to show that C follows (easily) from P by refuting $P \cup \{\neg C\}$, thus testing whether C can be (easily) derived as a valid lemma. If both tasks succeed, with proof-search lengths L_1 and L_2, and the original proof length is L, the cut ratio is

$$r = (L_1 + L_2)/L.$$

The conjecture C is *useful* when $r < 1$. This criterion is deliberately ATP-facing: a logically true lemma is not enough, because the cost of proving the lemma may exceed its search saving. This problem setting has been introduced in [1] together with a dataset of useful and harmful cuts derived from the runs of E prover [10].

Representation. Motivated by our previous anonymous/invariant architectures [8, 9], each CNF problem is represented as a typed graph with clause, literal, symbol, term, and variable nodes, and directed edges for relations such as clause membership, predicate occurrence, function application, and argument position. All occurrences of the same predicate or function share one symbol node. In the fully anonymous mode, a symbol initially receives only four structural features: predicate/function/constant indicators and normalized arity. Thus the model must infer mathematical role from usage, co-occurrence, and variable flow rather than from names. A hybrid mode adds learnable embeddings for roughly 3,097 frequent Mizar library symbols while keeping Skolem symbols anonymous. This preserves structural transfer while supplying useful semantic priors for common mathematical operators.

Architectures. We compared five decoders sharing the same heterogeneous GraphSAGE-style encoder.[2] *A: Transformer* is an autoregressive Transformer with causal self-attention, cross-attention over graph embeddings, pointer attention over symbol nodes, and KV-cached

[1] https://github.com/JUrban/E_conj_dev1/tree/master

[2] See also https://github.com/JUrban/E_conj_dev1/blob/master/report3.md#7-decoder-architectures.

inference. *B: Graph Growing* builds literals iteratively while conditioning on previously generated literal embeddings. *C: Conditional VAE* augments the Transformer with a latent variable to encourage diverse cuts. *D: Selective State-Space* (SSM) is a Mamba-style recurrent decoder with gated graph fusion; it is more exploratory and gives the best prover usefulness despite worse validation loss. *E: Subgraph Completion* predicts several literal slots in parallel. Architectures A, C, and D were competitive; B and E did not scale well.

Constrained decoding. The most important engineering step is arity-constrained decoding. Unconstrained decoders often generate malformed clauses, for example applying a binary predicate to three arguments. During generation we maintain a stack of pending arity obligations. After selecting a symbol of arity n, exactly n arguments must be generated before closing the term; nested function applications push further obligations. Invalid actions are masked, with no retraining. This raises syntactic validity from about 2% to 85–88%, and increases useful cut discovery by more than an order of magnitude.

Experimental observations. Named symbol embeddings reduce validation loss by 12–38% across the competitive decoder families, with the largest gain for the Transformer and the smallest for the SSM. The best validation loss is obtained by the VAE+named model (0.289), while the best test-set useful rate is obtained by SSM+named (5.8%). This mismatch is scientifically important: teacher-forced likelihood measures imitation of extracted proof cuts, but prover usefulness measures generative value under search. Conservative models produce easily provable lemmas; more adventurous models fail more often but occasionally expose decisive proof decompositions. Final selection therefore has to use prover evaluation, not validation loss alone.

Generated mathematical ideas. See the more detailed analysis[3] for many examples and their LLM-generated explanation. The strongest example is `l48_tops_1`, where several models independently generated variants of the set-difference fact

$$x \in A \setminus B \Rightarrow x \in A,$$

for example the clause `r2_hidden(X1,X2) | ~r2_hidden(X1,k6_subset_1(X2,X3))`. These conjectures reduce a proof of length 3889 by up to 77×. On `l21_wsierp_1`, the system generated cancellation lemmas equivalent to $(x \cdot y)/y = x$, with speedups above 11×. Other useful conjectures include BCI-algebra identities and graph-theoretic simplification lemmas. These examples suggest that the model is not merely producing well-typed text, but learning reusable structural proof patterns visible to saturation search.

Conclusion. The results support two claims. First, structural graph representations alone carry enough information to conjecture many valid lemmas: the anonymous model attains about 98% symbol precision, rarely hallucinating symbols absent from the problem. Second, symbol semantics and architectural diversity matter for usefulness. Different decoders find partially non-overlapping useful cuts, suggesting ensemble conjecturing. Future work should train from prover feedback, generate multi-clause cuts, combine conjecturing with ENIGMA-style[4, 11] guidance, use fast learning-guided ATPs such as ENIGMA and Deepire for large dataset generation and RL/feedback loops, and replace teacher-forced likelihood by ATP-aware objectives.

[3] https://github.com/JUrban/E_conj_dev1/blob/master/examples.md

References

[1] Z. Goertzel and J. Urban. Usefulness of Lemmas via Graph Neural Networks. In *4th Conference on Artificial Intelligence and Theorem Proving (AITP)*, 2019.

[2] A. Gu and T. Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752, 2023.

[3] W. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In *NeurIPS*, 2017.

[4] J. Jakubův and J. Urban. ENIGMA: Efficient Learning-Based Inference Guiding Machine. In *CICM*, pp. 292–302, 2017.

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

[6] L. Kovács and A. Voronkov. First-Order Theorem Proving and Vampire. In *CAV*, 2013.

[7] S. Loos, G. Irving, C. Szegedy, and C. Kaliszyk. Deep Network Guided Proof Search. In *LPAR*, 2017.

[8] M. Olsák, C. Kaliszyk, and J. Urban. Property Invariant Embedding for Automated Reasoning. In *ECAI*, pages 1395–1402, 2020.

[9] J. Piepenbrock, J. Urban, K. Korovin, M. Olsák, T. Heskes, and M. Janota. Invariant neural architecture for learning term synthesis in instantiation proving. *Journal of Symbolic Computation*, 128:102375, 2025.

[10] S. Schulz. E – A Brainiac Theorem Prover. *AI Communications*, 15(2–3):111–126, 2002.

[11] M. Suda. Efficient Neural Clause-Selection Reinforcement. In *CADE*, pages 403–422, 2025.

[12] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer Networks. In *NeurIPS*, 2015.

A Full experimental tables

The following tables show the detailed quantitative comparisons.

Table 1: Dataset statistics.

Metric	**Value**
Total problems	3,161
Total evaluated cuts	122,356
Good cuts ($r < 1.0$)	44,895 (36.7%)
Excellent cuts ($r < 0.1$)	1,509 (1.2%)
Unique Mizar symbols	5,105
Unique Skolem patterns	305
Median graph nodes per problem	395

Table 2: Action types for clause generation. PRED and ARG_FUNC use pointer attention over the encoder's symbol embeddings.

Action	**Argument**	**Description**
NEW_LIT_POS	—	Start positive literal
NEW_LIT_NEG	—	Start negative literal
PRED	symbol pointer	Select predicate
ARG_VAR	variable slot	Argument is a variable
ARG_FUNC	symbol pointer	Argument is a function application
END_ARGS	—	Close current arguments
END_CLAUSE	—	Generation complete

Table 3: Decoder architecture comparison. Val loss at 20 epochs (5,000 samples) and 100 epochs (full data, ∼44K samples). Training speed on A5000 GPU.

Decoder	**Params**	**Val@20**	**Val@100**	**Train (min)**	**Dropout**
A: Transformer	3.0M	1.71	0.474	48	0.1
D: SSM+dropout	2.6M	1.43	0.473	165	0.1
C: VAE	3.2M	1.75	0.403	∼50	0.1
B: Graph Growing	2.4M	3.02	—	—	0
E: Subgraph Compl.	2.7M	4.90	—	—	0

Table 4: Impact of named symbol embeddings on validation loss at 100 epochs. Named embeddings add ∼396K parameters for the embedding table plus ∼16K for the expanded input projection.

Model	**Anonymous**	**+Named**	**Reduction**
A: Transformer (100 ep)	0.474	0.293	**38.2%**
C: VAE (100 ep)	0.403	0.289	**28.3%**
D: SSM+dropout (100 ep)	0.473	0.413	**12.7%**

Table 5: Effect of arity-constrained decoding on generation quality. Full-scale evaluation on 2,985 problems, 30 conjectures per problem.

	Syntactic Validity	**Both Proved**	**Useful Speedups**	**Best Speedup**
Without arity constraints	∼2%	∼1.4%	∼31	14.7×
With arity constraints	**85–88%**	**∼61%**	**1,655**	**77×**

Table 6: E prover evaluation on 279 held-out test problems (2,790 conjectures per model). All models use arity-constrained decoding with 10 conjectures per problem. E 2.6 with `-auto` and 10s timeout.

Model	**Both Proved**	**Both %**	**Useful**	**Useful %**
D+named	1,655	59.4%	**161**	**5.8%**
A+named	1,679	60.0%	159	5.7%
C+named	1,707	60.8%	151	5.4%
A3 (150ep anon)	1,708	60.8%	145	5.2%
C anon	1,742	62.1%	144	5.1%
A2 (100ep anon)	1,744	62.1%	142	5.1%

Table 7: Full E prover evaluation on all 2,985 provable problems (29,850 conjectures per model, 10 per problem).

Model	**Both Proved**	**Both %**	**Useful**	**Useful %**	**Best Ratio**
D+named	18,093	61.0%	**1,655**	**5.6%**	0.015
A+named	18,380	61.7%	1,638	5.5%	**0.013**
C+named	18,380	61.7%	1,638	5.5%	**0.013**
D anon	17,921	60.1%	1,506	5.1%	0.014
C anon	∼18,100	∼60.6%	1,408	4.7%	—
A3 (150ep)	∼18,200	∼61%	∼1,350	∼4.5%	—
A2 (100ep)	18,223	61.1%	1,346	4.5%	0.015

Table 8: Top proof speedups from generated conjectures across all models.

Problem	**Model**	L	**Speedup**	**Generated Lemma**
l48_tops_1	A+named	3889	77×	$x{\in}A \vee x{\notin}A{\setminus}B$
l48_tops_1	C+named	3889	77×	Set difference variant
l48_tops_1	D+named	3889	67×	Set difference variant
l48_tops_1	A2	3889	67×	Set difference variant
l48_tops_1	D anon	3889	71×	Set difference variant
l48_tops_1	various	3889	25–33×	Additional set variants
l21_wsierp_1	various	2455	11.4×	$(x{\cdot}y)/y = x$ for ordinals
t64_bcialg_1	various	2805	11.2×	BCI-algebra property
l47_jgraph_2	various	3314	7.4×	Graph-theoretic identity

Table 9: Complete validation loss results across all configurations.

Model	**Mode**	**Val Loss**	**Notes**
C (VAE)	+named, 100ep	**0.289**	Best overall
A (Transformer)	+named, 100ep	0.293	
A (Transformer)	anonymous, 150ep	0.403	Extended training
C (VAE)	anonymous, 100ep	0.403	
D (SSM+dropout)	+named, 100ep	0.413	Overfitting (train=0.31)
D (SSM+dropout)	anonymous, 100ep	0.473	
A (Transformer)	anonymous, 100ep	0.474	
D (SSM, no dropout)	anonymous, 100ep	0.527	Severe overfitting
B (Graph Growing)	anonymous, 20ep	3.02	Failed to scale
E (Subgraph Compl.)	anonymous, 20ep	4.90	Failed to scale

Table 10: Syntactic validity with arity-constrained decoding (per-problem mode, `max_steps=80`). Remaining invalidity is almost entirely truncation.

Model	**Validity Rate**
C anonymous	87.7%
A+named	86.2%
C+named	85.0%
D+named	83.1%
D+dropout anonymous	80.8%
Any model, no constraints	∼2%

B Graphs

The following figures include the CNF graph representation, training curves, useful-speedup comparison, speedup-ratio histogram, full training curves, and the arity-constraint state machine.

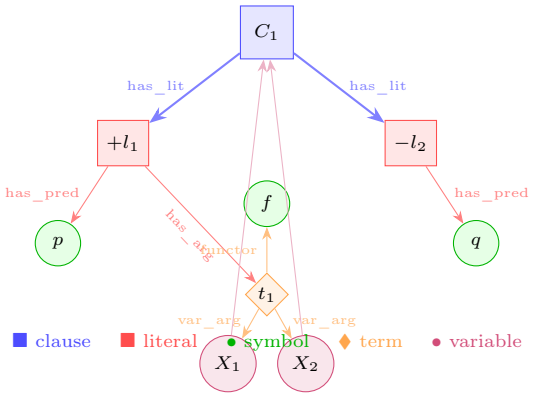

Figure 1: Heterogeneous graph representation of a CNF clause $C_1 = p(f(X_1, X_2)) \vee \neg q(\ldots)$. Symbol nodes (green) are shared across all occurrences—the same symbol node is used wherever a predicate or function appears, enabling the GNN to learn its structural role.

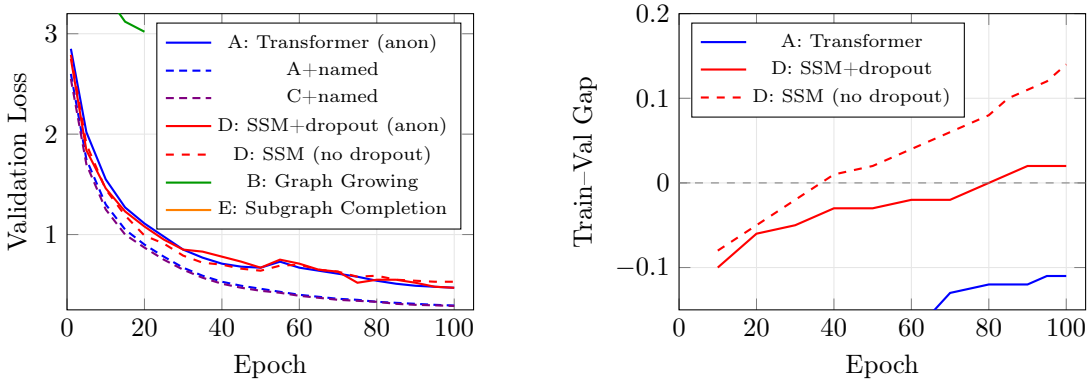

Figure 2: **Left:** Validation loss across training. Named embeddings (dashed blue and violet) reach ∼0.29, well below anonymous models (∼0.47). Plans B and E (shown for 20 epochs only) are significantly worse. **Right:** Train–val gap (positive = overfitting). D without dropout overfits progressively; adding dropout eliminates this. A's negative gap is an artifact of dropout being active during training but not evaluation.

C Extended discussion of generated conjectures

Set-theoretic decomposition. The best speedups occur for `148_tops_1`. Several generated clauses express variants of the elementary but search-critical principle that membership in a set

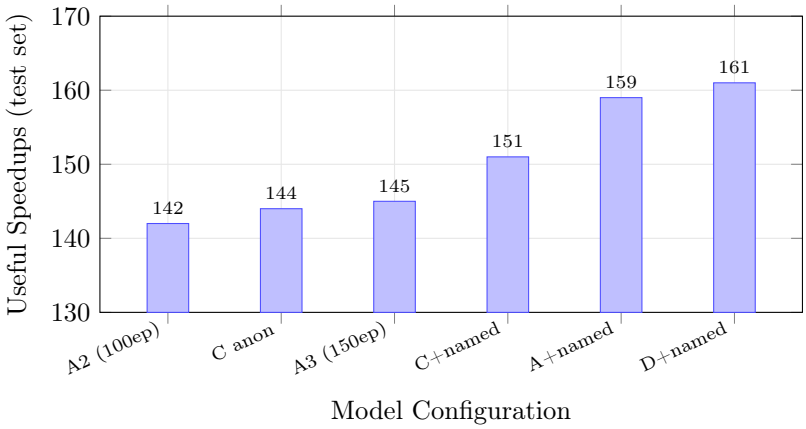

Figure 3: Test-set useful speedups across all six evaluated model configurations (279 problems, 2,790 conjectures each). Named embeddings (right three bars) consistently outperform anonymous models (left three bars). D+named achieves the highest useful count despite having the worst validation loss among named models.

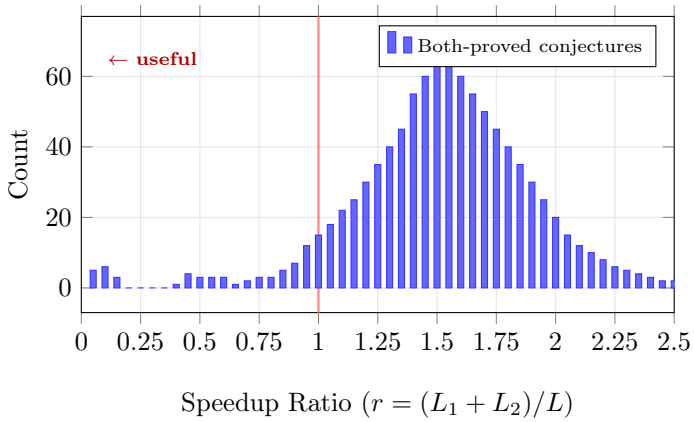

Figure 4: Distribution of speedup ratios for both-proved conjectures. The red line marks r=1.0: conjectures to the left provide actual speedup.

difference implies membership in the left-hand set. The clause

```
r2_hidden(X1,X2) | ~r2_hidden(X1,k6_subset_1(X2,X3))
```

corresponds to $x \in A \setminus B \Rightarrow x \in A$ and reaches ratio 0.013, i.e. a 77× speedup. A second family, roughly $x \in A \setminus B \vee x \in B \vee x \notin A$, exposes the complementary case split hidden in the set-difference definition. From the prover's perspective these are not deep human theorems, but they are excellent cuts: they replace blind saturation through definitions by a direct decomposition.

Arithmetic cancellation. For `l21_wsierp_1`, generated clauses include

```
$eq(k7_xcmplx_0(k3_xcmplx_0(X1,X2),X2),X1) | ~v7_ordinal1(X1).
```

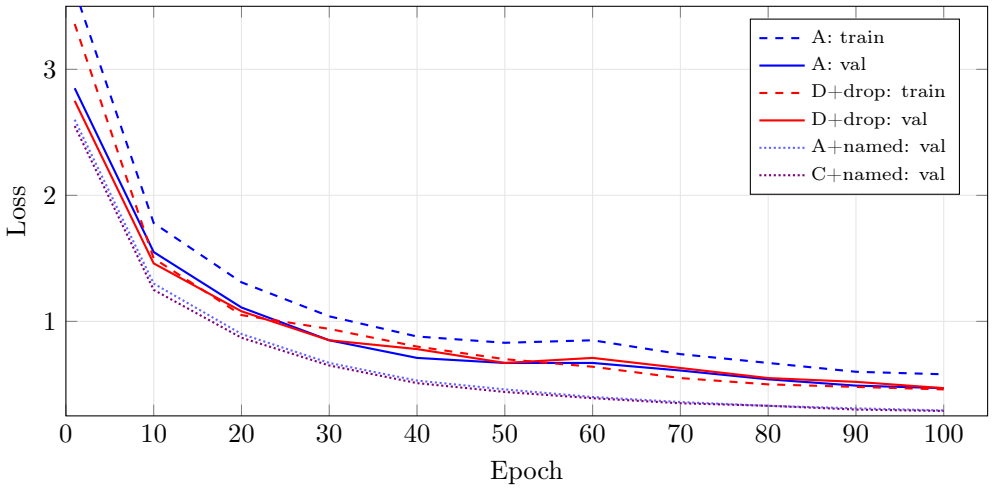

Figure 5: Full training curves for all competitive configurations. Anonymous models (A, D+dropout) converge to val≈0.47. Named models (A+named, C+named) converge to val≈0.29, a 38% reduction. A's train loss is higher than val due to dropout being active during training but not evaluation.

Informally this is a conditional form of $(x \cdot y)/y = x$. The generated lemma acts like algebraic preprocessing, shortening the search by more than 11×. A related variant normalizes $(6 \cdot x)/x$ back to 6 under the same kind of side condition.

Algebraic, graph-theoretic, and non-useful valid lemmas. The full report also records useful BCI-algebra properties for `t64_bcialg_1` and graph-theoretic identities for `l47_jgraph_2`. Conversely, on `l100_fomodel0`, all sampled conjectures were valid, but none improved proof search because the original problem was already easy. This is a useful negative example: validity, even at 100%, is not the same as ATP utility.