From Explicit Algorithm Extraction to Sparse Search: Machine Learning for Constructive Mathematics

Coco Xiaoyu Huang

Temple University

VaNTAGe 2026
Tuesday 31st March 2026

Overview

Many recent talks focus on using AI to formalize mathematics and prove theorems:

- Competition level: IMO, Putnam Chen et al. 2025; Ren et al. 2025
- Research level: Abouzaid et al. 2026; Bryan et al. 2026; Feng 2026

In this talk, I will instead focus on several ways machine learning can contribute to constructive mathematics:

1. finding explicit relationships between mathematical structures,
2. finding sparse examples related to open conjectures,
3. and, as a byproduct, revealing interesting observations that may later be formulated as conjectures.

Frobenius Trace at p / Euler Factors

For an elliptic curve over the rationals, $E/\mathbb{Q}$,

$$y^2 = x^3 + ax + b, \qquad a, b \in \mathbb{Q},$$

one can associate an L-function $L(E, s)$, defined by local Euler factors:

$$L(E, s) = \prod_p L_p(E, s).$$

For all but finitely many primes p, the curve E has good reduction, and

$$L_p(E, s) = \left(1 - a_p p^{-s} + p^{1-2s}\right)^{-1},$$

where the Frobenius trace is

$$a_p(E) = p + 1 - \#E(\mathbb{F}_p).$$

Learning Euler Factors of Elliptic Curves

This project was carried out during the *Mathematics and Machine Learning* program at Harvard CMSA: Babei et al. 2025

Joint work with Angelica Babei, François Charton, Edgar Costa, Kyu-Hwan Lee, David Lowry-Duda, Ashvni Narayanan, and Alexey Pozdnyakov.

1. Goal: given some a_p-values of an elliptic curve, can a machine learning model predict other a_p-values?
2. Model: neural networks and transformers, implemented using `int2int`, a sequence-to-sequence transformer for short integer sequences.
3. Data: `ECQ6`. In most experiments, 2 million curves for training and 10,000 curves for testing, with no overlap. We keep only one curve from each isogeny class.

Results – predict a_{97}

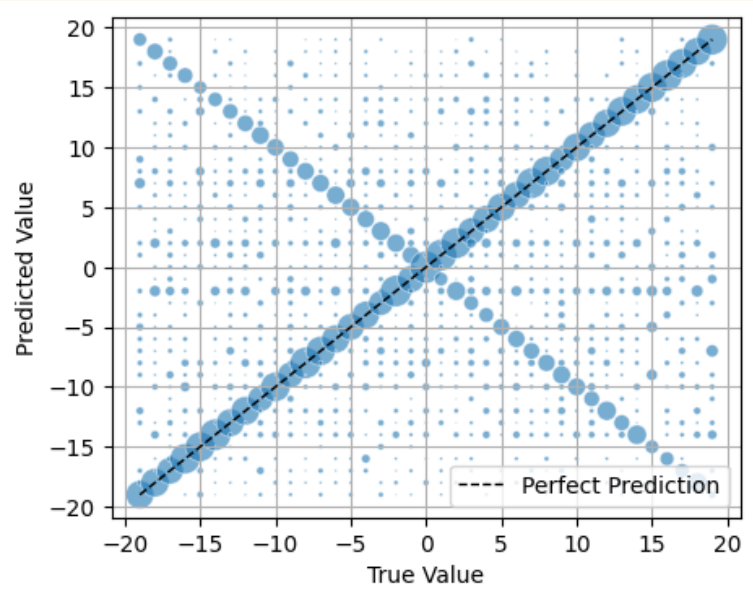
Here is one of the experiments we ran.

1. Task: given $\{a_q\}_{q\neq 97,\ q<97}$, predict a_{97}. The target range is

$$-19 \leq a_{97} \leq 19.$$

Result: the prediction accuracy is about 0.5. The baseline accuracy from random guessing is about 0.025.

The main difficulty is that the model struggles to distinguish the sign of a_{97}.

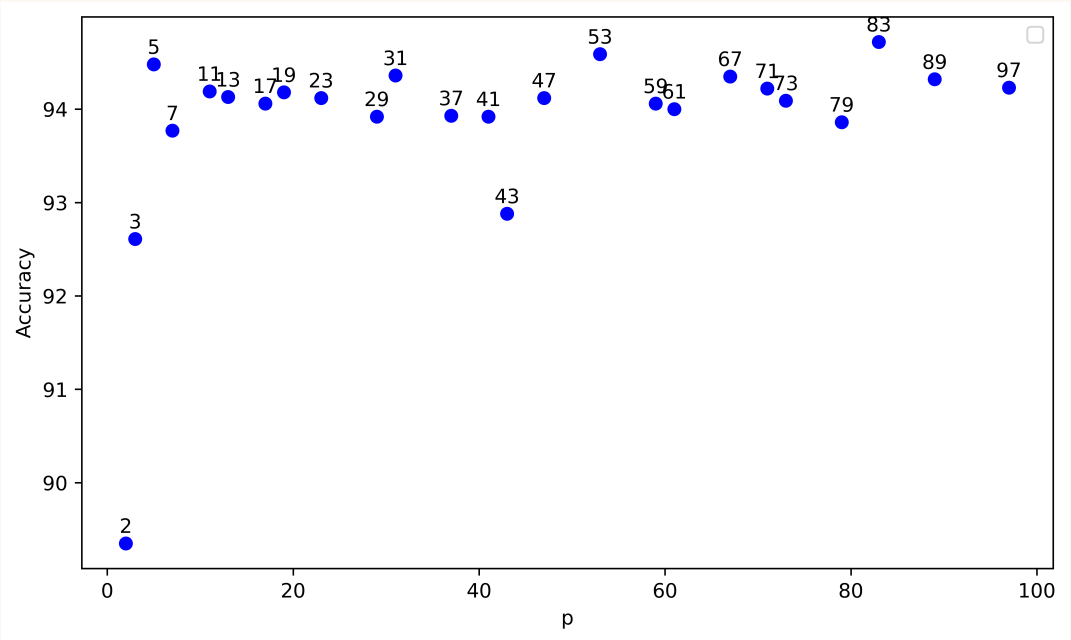


Results – predict $a_p \pmod 2$

2. Task: given $\{a_q\}_{q \neq p,\ q<100}$, predict

$$a_p \pmod 2.$$

Result: except for $p = 2, 3, 43$, the model achieves accuracy close to 0.94.



Embeddings in a Transformer

The transformer architecture is the standard backbone of modern large language models.

In our setting, the processing pipeline is roughly:

Numerical a_p

$\downarrow$

discrete token

$\downarrow$

embedding lookup

$\downarrow$

$\mathbb{R}^{256}$ representation

$\downarrow$

transformer layers and prediction

An Example

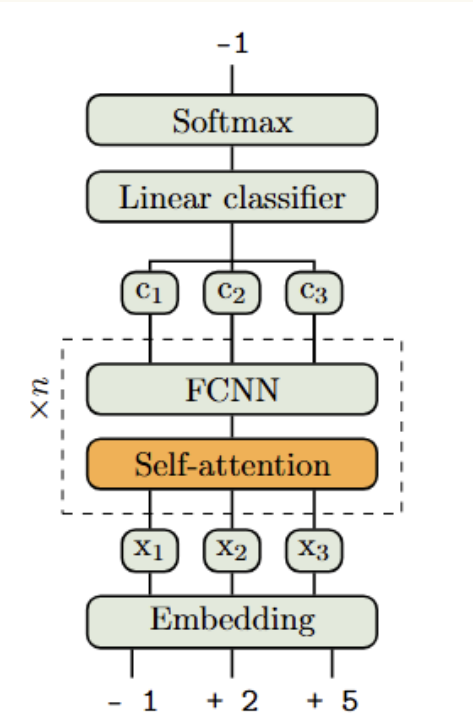
The elliptic curve of conductor 11 has 24 values a_p with $p < 97$ to predict a_{97}:

$$-2, -1, 1, \cdots, -6, 15$$

These become a sequence of tokens:

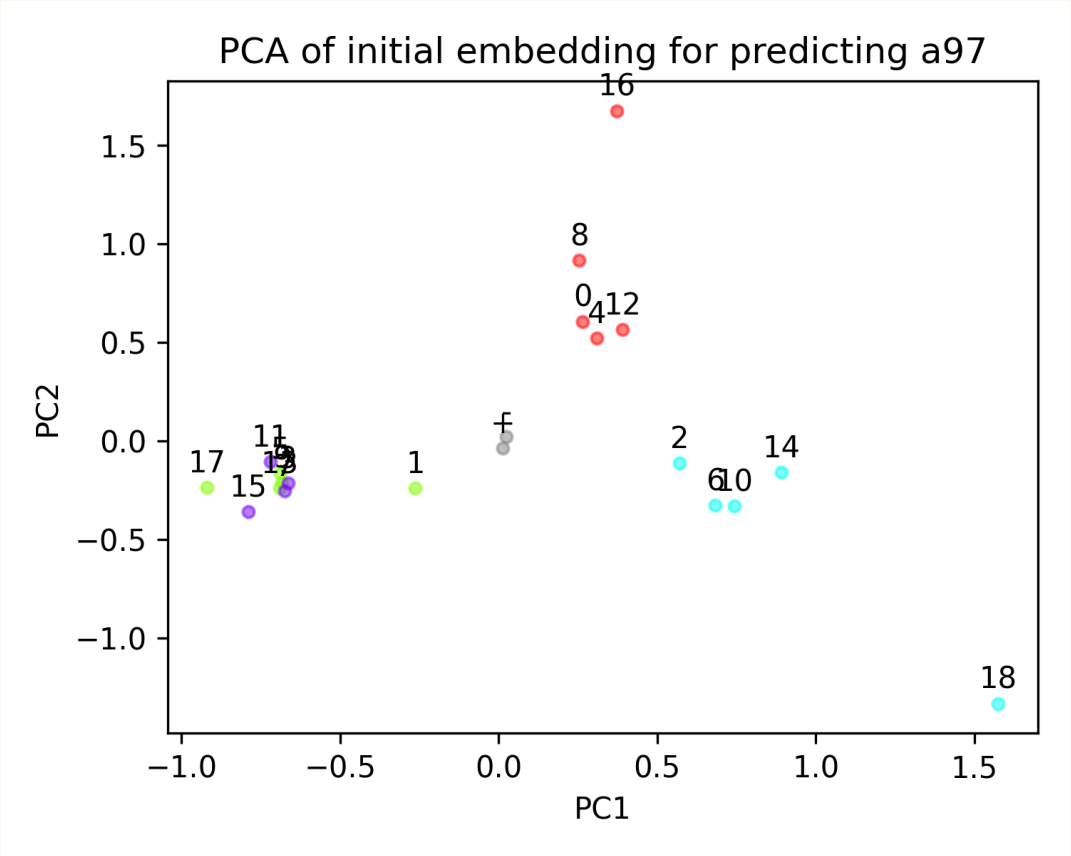
$$-, 2, -, 1, +, 1, \cdots, -, 6, +, 15$$

These tokens become a matrix in $\mathbb{R}^{48\times 256}$ for further processing.



Interpretability Summary

1. The model appears to use the input values $a_q \bmod 2$ to help predict $a_{97} \bmod 2$. This may explain part of its confusion about the sign.



A clear separation by congruence classes modulo 4 appears in the PCA of the initial embeddings.

Interpretability Summary

2. In the second task, where the input is $\{a_q\}$ and the target is $a_p \pmod 2$, the model again appears to convert the input a_q-values into their classes modulo 2, and then use that information to predict $a_p \pmod 2$.

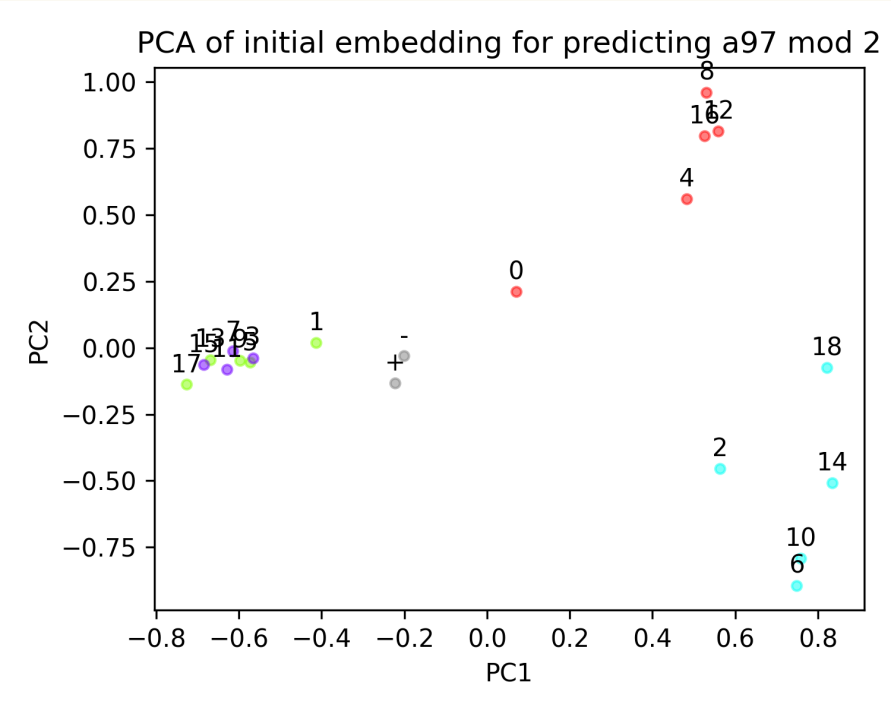


Figure: A very similar PCA separation appears when predicting $a_{97} \pmod 2$ and when predicting a_{97}.

Interpretability Summary

3. There are early signs that the model is using not only information modulo 2 and 4, but also other local information, such as modulo 3 and 6.

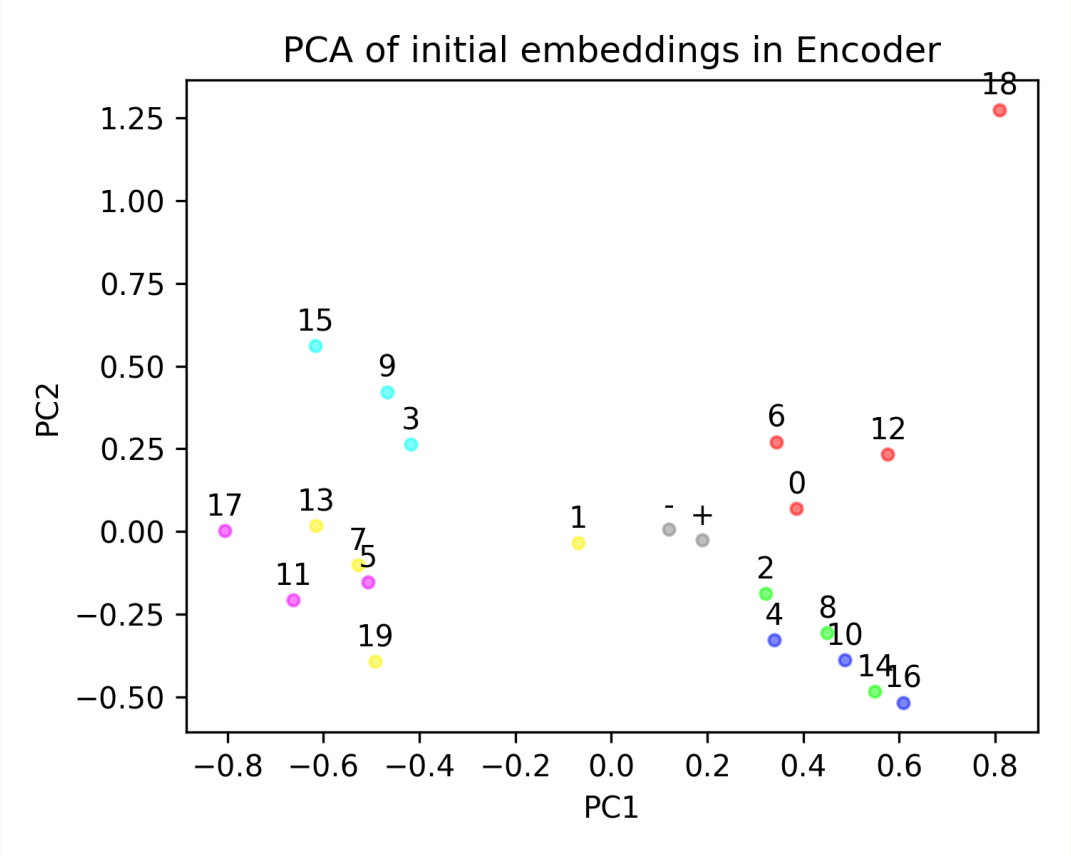


Figure: Separation according to congruence classes modulo 6 and modulo 2 when predicting a_2, where $-2 \leq a_2 \leq 2$.

Arithmetic Structure in Data

After discussing these results with fellow number theorists, one plausible explanation is that the model is exploiting arithmetic structure in the data through memorized quadratic twists:

- For an elliptic curve $E : y^2 = x^3 + ax + b$, its quadratic twist E^d is given by

$$E^d : dy^2 = x^3 + ax + b,$$

and for any prime p of good reduction for both E and E^d, one has

$$a_p(E^d) = \pm a_p(E).$$

- Many curves in ECQ6 are in fact quadratic twists of one another.

This could explain the model's sign ambiguity, as well as its apparent dependence on information modulo 2.

❓ While this is interesting, can a transformer genuinely learn an algorithm rather than rely on memorization?

🤦 Let us look at an example suggesting that the answer is yes.

Dyck Paths

Dyck paths of semilength n are monotone lattice paths in an $n \times n$ grid that stay above the diagonal.

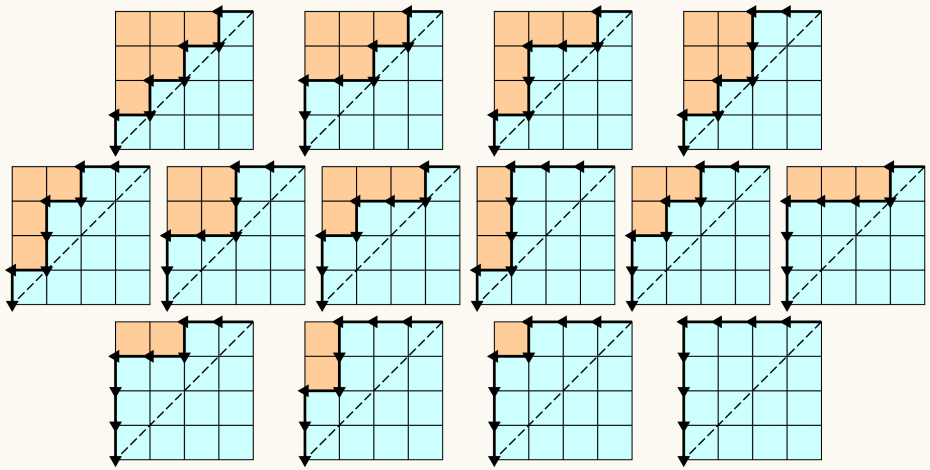


Figure: All Dyck paths for $n = 4$.

They can be encoded as *Dyck words* of length $2n$:

$$\text{up} \mapsto 1, \qquad \text{right} \mapsto 0.$$

Example: the last path is

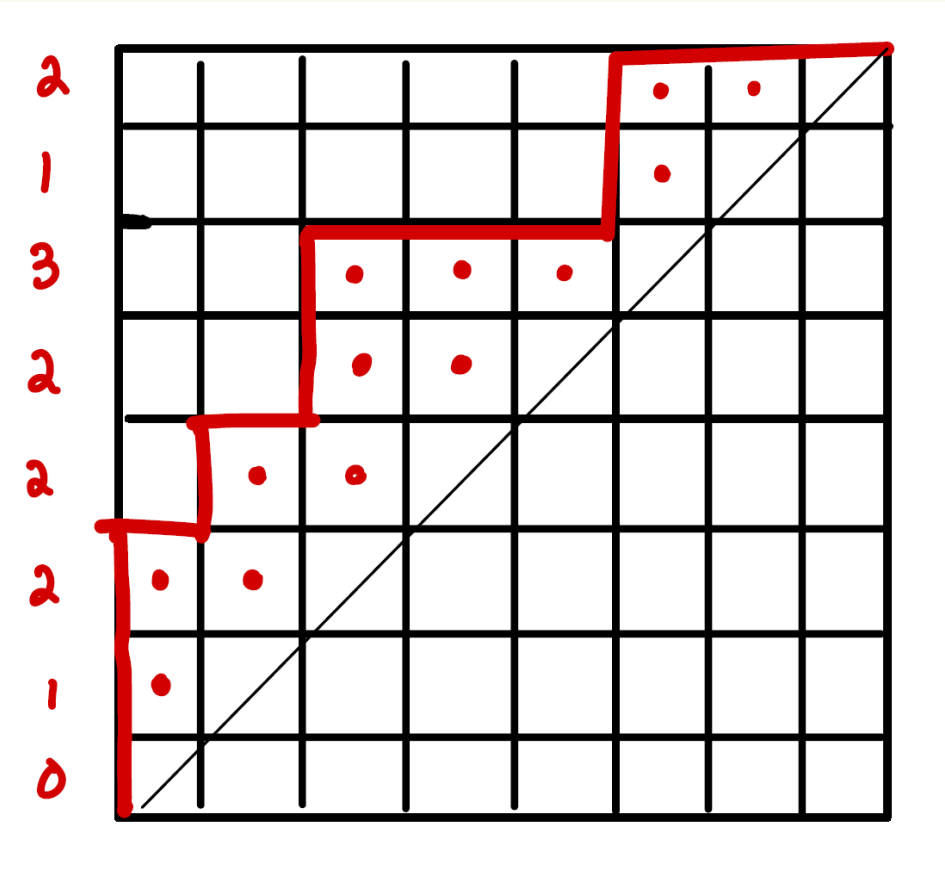
$$1, 1, 1, 1, 0, 0, 0, 0.$$

Associated with each Dyck path are three statistics: *area*, *bounce*, and *dinv*.

Area

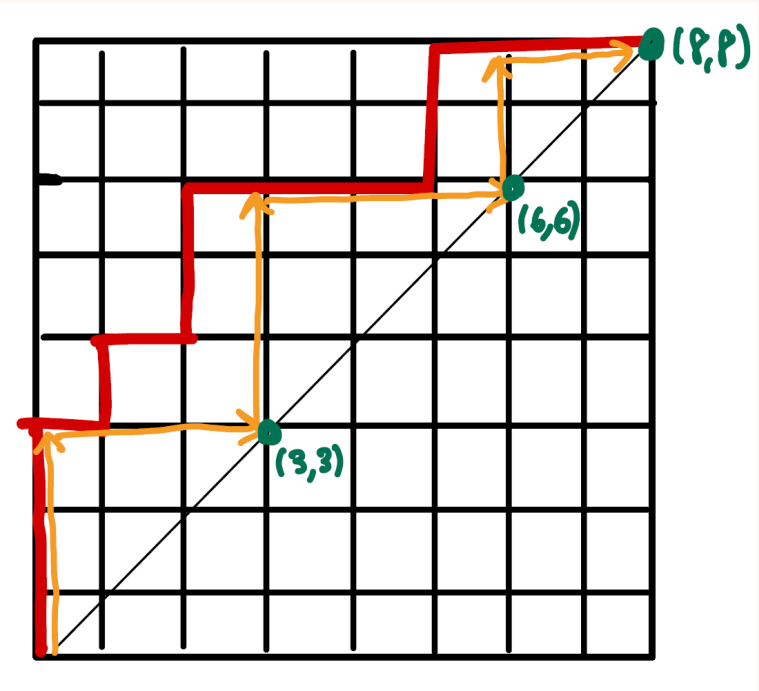
Consider the following Dyck path π. The area sequence $\{\alpha_i\}$ records, from bottom to top, the number of full boxes between the path and the diagonal. Then

$$\text{area}(\pi) = \sum \alpha_i.$$



Bounce

$$bounce(\pi) = \sum_{\text{bounces at } (j,j)} (n-j)$$

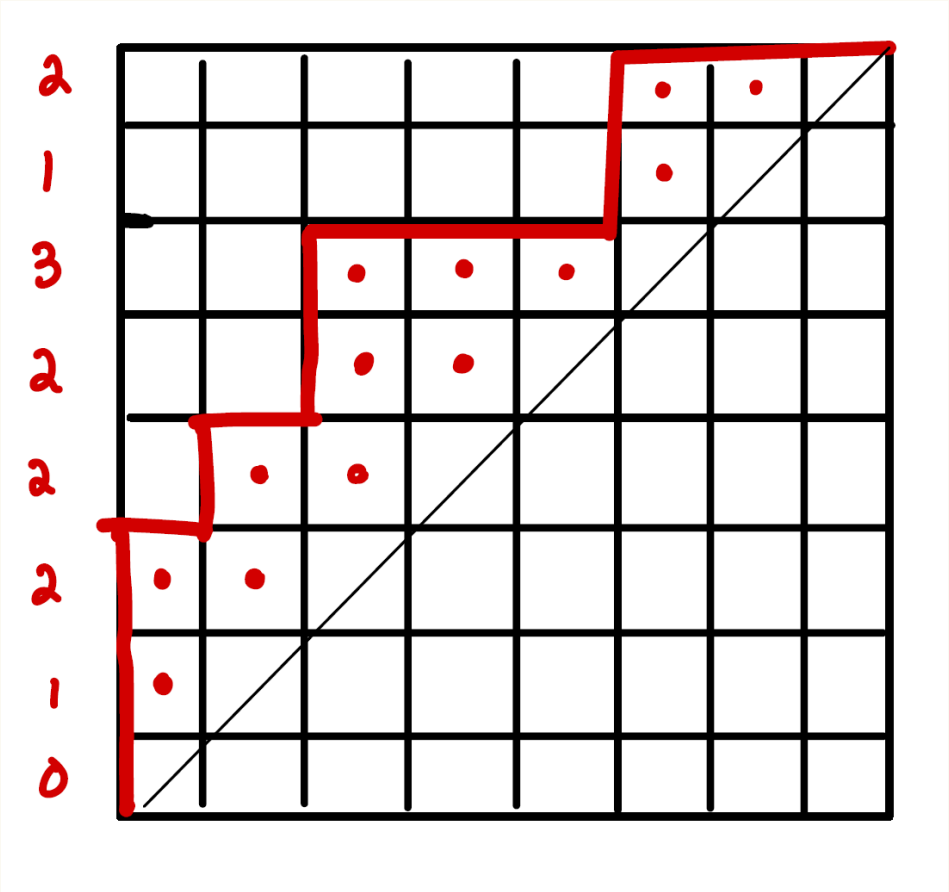


In this example,

$$bounce(\pi) = (8-3) + (8-6) + (8-8) = 7.$$

Dinv

$$dinv(\pi) = \#\{(i,j) : 1 \leq i < j \leq n,\ \alpha_i = \alpha_j \text{ or } \alpha_i = \alpha_j + 1\}$$



Zeta Map

From work in algebraic combinatorics (haiman2000, haglund2003), we know that there is a unique bijection, the zeta map ζ, on the set of Dyck paths that exchanges

$$(\text{area}, \text{bounce}) \longleftrightarrow (\text{dinv}, \text{area}).$$

In other words, if π is sent to $\zeta(\pi)$, then

$$(\text{area}(\pi), \text{bounce}(\pi)) = (\text{dinv}(\zeta(\pi)), \text{area}(\zeta(\pi))).$$

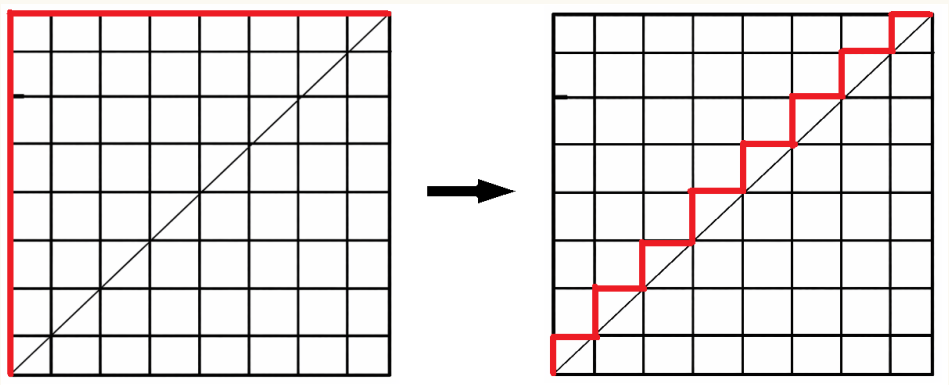


Figure: An example of a Dyck path and its image under the zeta map.

Learning the Zeta Map

In X. Huang, Jackson, and Lee 2025, we trained transformer models to learn the zeta map for Dyck words of a fixed semilength n.

Task

Given a Dyck word, predict its image under the zeta map. For example, for semilength 5,

$$1,0,1,1,0,1,0,0,1,0 \longmapsto 1,1,1,0,0,1,1,0,0,1.$$

For $n = 11$ to 15, we experimented with a variety of transformer architectures.

Main observation

Most models achieved near-perfect prediction accuracy.

Interpretability: The Minimal Dyck Transformer

Among the models we tested, the simplest encoder–decoder model, the *Minimal Dyck Transformer*, has just one layer and one attention head.

Observation

Despite its simplicity, it achieves 100% accuracy.

We then extracted the algorithm implemented by this model, which we called the *scaffolding algorithm*.

1. To the best of our knowledge, the scaffolding algorithm is the first algorithmic interpretation of the zeta map that does not directly rely on the area sequence.
2. The validity of the scaffolding algorithm is entirely mathematical. The machine-learning component serves only as a discovery aid and does not replace mathematical proof.

Transformer Cross-Attention

The key to the extraction is to examine the cross-attention in the transformer model.

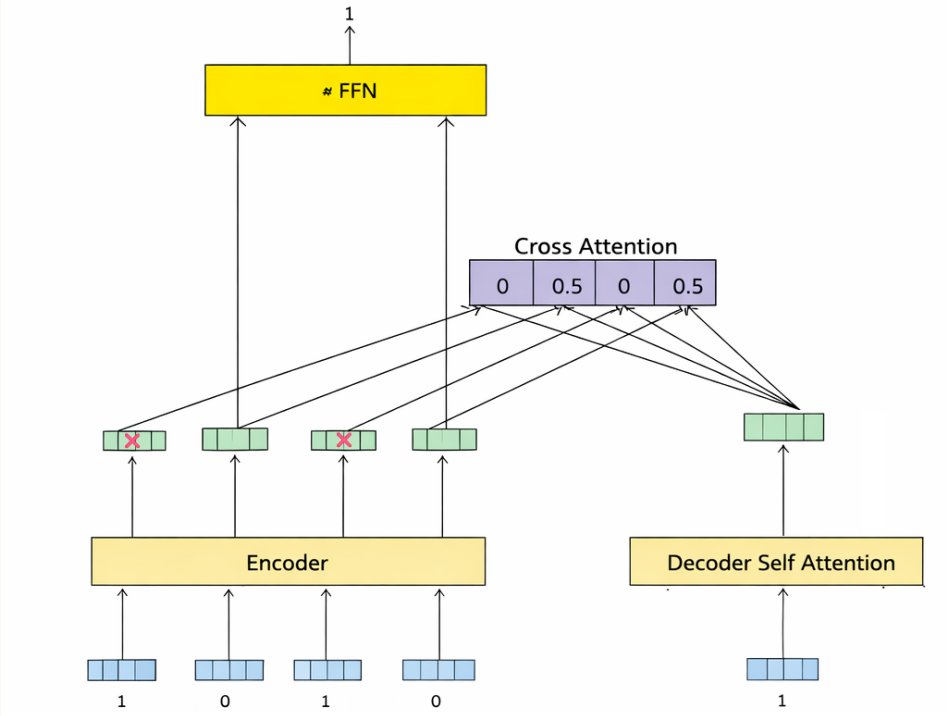
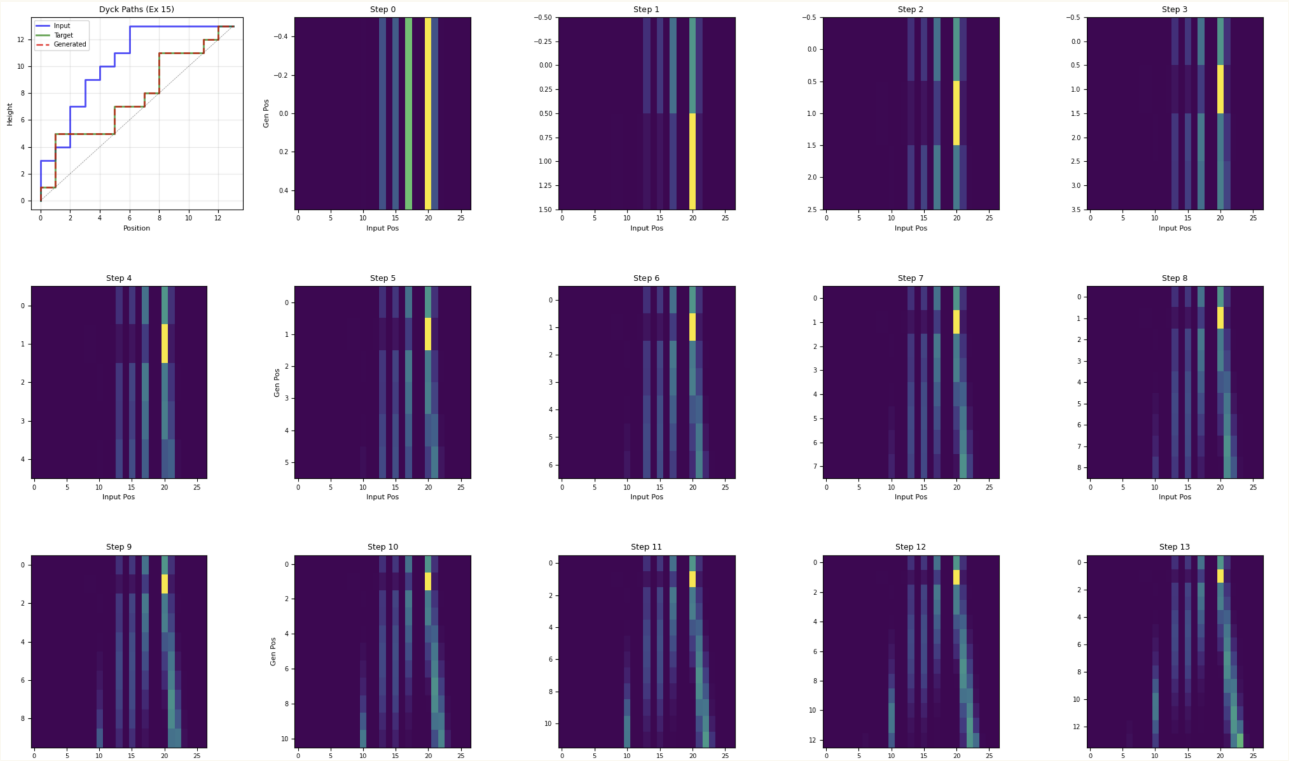


Figure: A rough sketch of our model with a focus on cross-attention.

Consistent Cross-Attention Patterns

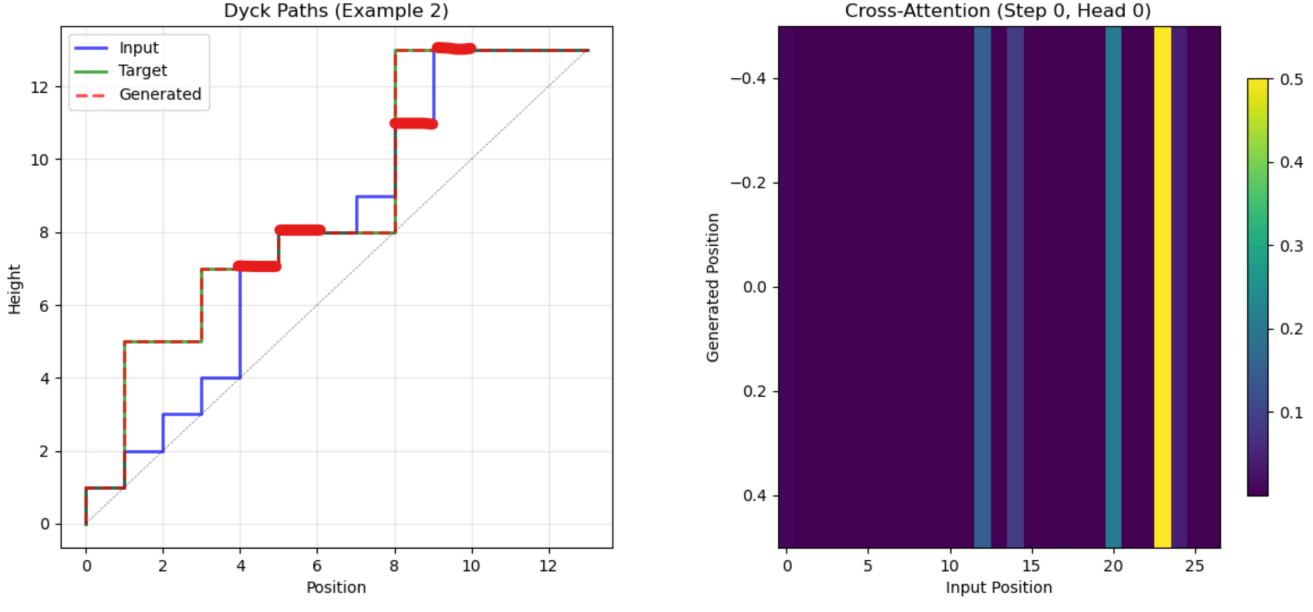
In our case, the biggest hint about what the model is doing comes from tracking the cross-attention matrix throughout output generation.



A clear triangular pattern appears in many examples. There are also consistent gaps.

Let's Zoom In

Let's zoom in on the attention pattern during the generation of the first token.



1. The model attends to the right steps in the Dyck path.
2. In particular, it attends to the corners that are furthest from the diagonal.

The Scaffolding Algorithm

By following when attention appears and disappears, we extracted the scaffolding algorithm: agents moving along a scaffold to collect information.

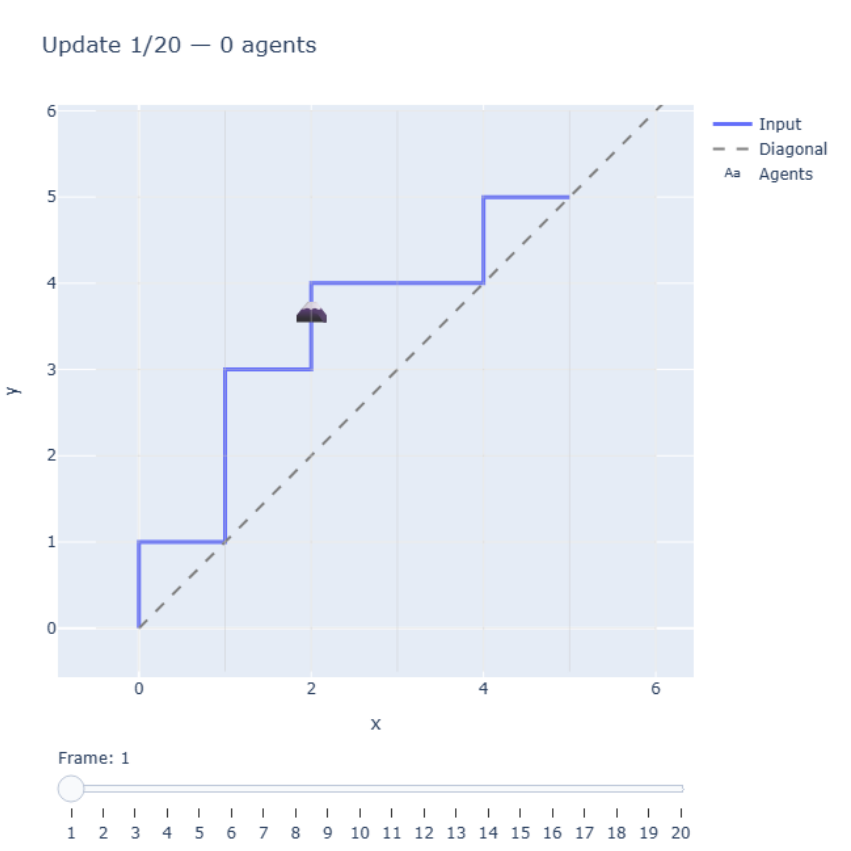


Figure: We illustrate the scaffolding algorithm using this example, where the Dyck word is $1, 0, 1, 1, 0, 1, 0, 0, 1, 0$.

Neural Networks and Algorithmic Tasks

Should we be surprised that a transformer can internally learn and implement a concrete algorithm?

1. *Are there other cases where neural networks have been found to implement recognizable algorithms?*

   Yes, across number theory, algebra, and stochastic processes.

   - Modular arithmetic and group operations Nanda et al. 2023; Chughtai, Chan, and Nanda 2023
   - In recent work L. Wang and X. Huang 2026, we find that a transformer trained for a state-space model naturally develops a two-stage computation:
     1. it first infers the latent volatility state from past observations,
     2. then maps that representation to the forecast variable.

   This mirrors the classical filtering–prediction structure.

Caution And Opportunity

2. *Do models reliably rediscover the known algorithms for these tasks?*

   No.

   - Patterns can be nearly randomized and hard to interpret (Dyck language case study Wen et al. 2023).
   - Variants of familiar algorithms may be discovered (the clock and pizza case study on modular arithmetic Zhong et al. 2023).
   - Claude appears to perform mental arithmetic in a complicated internal way, even when its explanation sounds familiar to us (Anthropic 2025).

Language Modeling and Number Theory

❓ Should we be cautious when applying transformer models to number-theoretic data?

Transformers were designed for sequence modeling and work especially well when the input naturally forms a sequence with meaningful contextual dependencies.

Dyck paths, for example, can naturally be viewed as a formal language (cf. Strobl et al. 2024).

Local Structure in Number-Theoretic Data

🧐 A distinctive feature of number theory is that the data can carry *local arithmetic meaning*.

For example, in the Euler factor learning task of Babei et al. 2025, the model may be given a sequence such as

$$-2, 1, 1, \cdots, -6, 15.$$

But a plain sequence representation does not explicitly indicate that

- the first 1 corresponds to information modulo 3,
- the second 1 corresponds to information modulo 5.

Transformers may still infer such context from surrounding tokens, especially at scale, but that does not mean the representation faithfully captures the underlying mathematical structure.

Local Structure

Here is an even more concrete example based on the Chinese remainder theorem:

$$x \equiv 1 \pmod{101}$$
$$x \equiv 23 \pmod{103}$$

A transformer struggles if the prediction task is simply

$$1, 23 \to 9293$$

or even

$$1, 101, 23, 103 \to 9293.$$

This is because the numbers 1 and 23 are in different modular contexts.

By contrast, the task becomes much easier if the model is designed so that the two residues pass through different embedding lookups.

✨ Maybe `poly2poly` is a natural thing to try?

Sparse Search

Apart from algorithmic discovery, another major goal of AI for constructive mathematics is to find sparse examples or counterexamples.

Three approaches are especially popular at the moment:

1. Reinforcement learning (RL) (Wagner 2021; Gukov et al. 2021; Shehper et al. 2024)
2. Evolutionary search with LLM-proposed candidates (Novikov et al. 2025)
3. Generative modeling combined with local search (Charton et al. 2024; Bérczi, Hashemi, and Klüver 2026)

I will briefly discuss a current direction on RL for constructive search at a high level.

RL for Constructive Mathematical Search

1. Goal: find rare mathematical objects with a desired property, such as high-rank curves, extremal configurations, or unusual algebraic structures.
2. A natural viewpoint is to model construction as a sequential decision process:
   - a *state* records the current partial object,
   - an *action* modifies or extends it,
   - a *reward* measures how promising the construction is.
3. This is appealing because, in sparse search problems, we usually do not know beforehand what a successful example should look like.

More broadly, any problem that can be formulated as a sequential decision process is a natural candidate for RL.

Andrews–Curtis Conjecture

Let us illustrate the workflow using the Andrews–Curtis conjecture, a long-standing conjecture in combinatorial group theory.

Andrews–Curtis conjecture $(n = 2)$

Any balanced presentation of the trivial group $\langle x, y \mid r_1, r_2 \rangle$ can be transformed into the trivial presentation $\langle x, y \mid x, y \rangle$ using a finite sequence of AC-moves.

The AC-moves are:

- (AC1) Replace a relator r_i by its inverse r_i^{-1}.
- (AC2) Replace a relator r_i by $r_i r_j$ for $i \neq j$.
- (AC3) Replace a relator r_i by $w r_i w^{-1}$ for any word w in the generators.

State, Action, Reward

Shehper et al. 2024 first applied RL to the Andrews–Curtis conjecture and obtained strong results on two major families of potential counterexamples:

1. various potential counterexamples in the Miller–Schupp series (1991), including three infinite subfamilies,
2. and length reducibility for all but two presentations in the Akbulut–Kirby series (1981).

What are the state, action, and reward in Shehper et al. 2024?

1. State: the relators r_1, r_2; for example, $y^{-1}x^{-1}yx^{-1}y^{-1}x, y^{-3}x^{-4}$
2. Action: AC-moves
3. Reward: a combination of relator-length reduction and a large terminal reward when the state becomes trivial

Improvements

The main challenge is usually not the RL formalism itself, but difficult design choices:

1. Action space: the moves must be rich enough to reach interesting objects within a finite horizon, but restricted enough to keep the search tractable.
2. Sparse reward: since the desired object is rare, a naive reward often provides almost no learning signal. The reward should correlate with genuine mathematical progress, not just with easy-to-game proxies.
3. Learning algorithm: the machine learning model the RL agent relies on should be appropriate for the structure of the search problem.

In follow-up work, we found that addressing these issues can substantially improve performance over the earlier baseline in Shehper et al. 2024.

❓ Can we build similar RL frameworks for constructive search problems in number theory?

Observations

As a byproduct, these methods may also reveal interesting mathematical observations.

1. Murmurations, as discussed in the previous talks.
2. If we write a minimal Weierstrass equation for an elliptic curve E as

$$E : y^2 + A_1xy + A_3y = x^3 + A_2x^2 + A_4x + A_6,$$

then we find that

$$\begin{aligned} A_1 &\equiv a_2(E) \bmod 2, \\ A_2 &= (a_3(E) + 1 - A_1) \bmod 3 - 1, \end{aligned}$$

regardless of whether 2 and 3 are primes of good reduction for E.

Thank You

Thank you for listening!
Please feel free to contact me anytime if you have any questions or would like to chat.

References I


Abouzaid, Mohammed et al. (2026). "First Proof". In: *arXiv preprint arXiv:2602.05192*.


Anthropic (2025). *Tracing the thoughts of a large language model*. Accessed: 2025-03-29. URL: `https://www.anthropic.com/research/tracing-thoughts-language-model`.


Babei, Angelica et al. (2025). "Learning Euler Factors of Elliptic Curves". In: *arXiv preprint arXiv:2502.10357*.


Bérczi, Gergely, Baran Hashemi, and Jonas Klüver (2026). "Flow-based Extremal Mathematical Structure Discovery". In: *arXiv preprint arXiv:2601.18005*.


Bryan, Jim et al. (2026). "The motivic class of the space of genus 0 maps to the flag variety". In: *arXiv preprint arXiv:2601.07222*.


Charton, François et al. (2024). "Patternboost: Constructions in mathematics with a little help from ai". In: *arXiv preprint arXiv:2411.00566*.


Chen, Luoxin et al. (2025). "Seed-prover: Deep and broad reasoning for automated theorem proving". In: *arXiv preprint arXiv:2507.23726*.

References II

- Chughtai, Bilal, Lawrence Chan, and Neel Nanda (2023). "A toy model of universality: Reverse engineering how networks learn group operations". In: *International Conference on Machine Learning*. PMLR, pp. 6243–6267.
- Feng, Tony (2026). "Eigenweights for arithmetic Hirzebruch Proportionality". In: *arXiv preprint arXiv:2601.23245*.
- Gukov, Sergei et al. (2021). "Learning to unknot". In: *Machine Learning: Science and Technology* 2.2, p. 025035.
- Huang, Xiaoyu, Blake Jackson, and Kyu-Hwan Lee (2025). "From Black Box to Bijection: Interpreting Machine Learning to Build a Zeta Map Algorithm". In: *arXiv preprint arXiv:2511.12421*.
- Nanda, Neel et al. (2023). "Progress measures for grokking via mechanistic interpretability". In: *arXiv preprint arXiv:2301.05217*.
- Novikov, Alexander et al. (2025). "AlphaEvolve: A coding agent for scientific and algorithmic discovery". In: *arXiv preprint arXiv:2506.13131*.

References III

- Ren, ZZ et al. (2025). "Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition". In: *arXiv preprint arXiv:2504.21801*.
- Shehper, Ali et al. (2024). "What makes math problems hard for reinforcement learning: a case study". In: *arXiv preprint arXiv:2408.15332*.
- Strobl, Lena et al. (2024). "What formal languages can transformers express? a survey". In: *Transactions of the Association for Computational Linguistics* 12, pp. 543–561.
- Wagner, Adam Zsolt (2021). "Constructions in combinatorics via neural networks". In: *arXiv preprint arXiv:2104.14516*.
- Wang, Lulu and Xiaoyu Huang (2026). "Do Transformers Recover Latent Volatility States? Evidence from Stochastic Volatility Models". Manuscript, submitted under double-blind review.
- Wen, Kaiyue et al. (2023). "Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars". In: *Advances in Neural Information Processing Systems* 36, pp. 38723–38766.

References IV

 Zhong, Ziqian et al. (2023). "The clock and the pizza: Two stories in mechanistic explanation of neural networks". In: *Advances in neural information processing systems* 36, pp. 27223–27250.