Expressivity of Transformers: Logic, Circuits, and Formal Languages

Day 5: Conclusion

David Chiang (Univ. of Notre Dame, USA)
Jon Rawski (MIT/San Jose State Univ., USA)
Lena Strobl (Umeå University, Sweden)
Andy Yang (Univ. of Notre Dame, USA)
2 August 2024

Introduction

Join our family

Online seminars on Formal Languages and Neural Networks (FLaNN).

Learn more about FLaNN on
`https://flann.super.site/`

Invite link
`https://discord.gg/zjradK75`

The link will expire in 7 days. Please feel free to share it with anyone you think might be interested. I chose not to post it on the ESSLLI channel to ensure it's shared with those who have a genuine interest rather than just curiosity about the link destination.

Today's Goals

- Recap the course
- Open issues in expressivity
- General expressivity discussion/questions
- Discuss learnability vs expressivity

Course Review

Analyzing transformers

How to evaluate a 'Language model'?

Empirically.

- train a model on corpus data, evaluate trained model on NLP task benchmarks
- probe a trained model using advanced correlational techniques

Theoretically.

- ??

How to evaluate a 'Language model'?

Empirically.

- train a model on corpus data, evaluate trained model on NLP task benchmarks
- probe a trained model using advanced correlational techniques

Theoretically.

- Logical satisfaction
- Circuit families
- formal language recognition

Analyzing transformers

How to evaluate a 'Language model'?

Empirically.

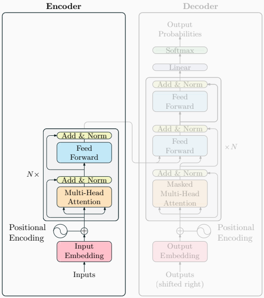
- train a model on corpus data, evaluate trained model on NLP task benchmarks
- probe a trained model using advanced correlational techniques

Theoretically.

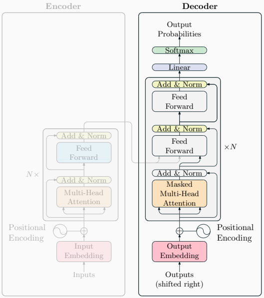
- Logical satisfaction
- Circuit families
- Formal language recognition

What are the advantages of each of these three?

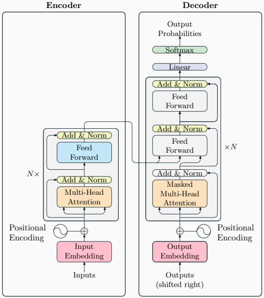
Decisions we made



Encoder-only



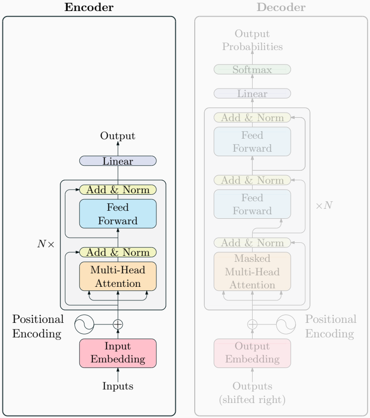
Decoder-only



Encoder-Decoder

Decisions we made

Definition of recognition
To use it as a language recognizer, we add an output layer that converts it to a probability.

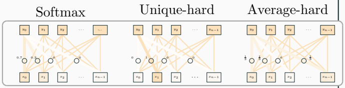


Decisions we made

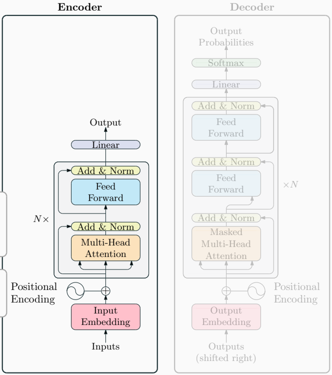
Residuals
- pre-norm
- post-norm

Layer norm
- include
- omit

Precision
- infinite
- $O(\log n)$
- $O(1)$



$$i, \frac{i}{n}, \frac{1}{i}, \frac{1}{2i}$$



Overview

Lower bound	Source	PE	Attention	Notes
$\ni$ MAJORITY	Pérez et al. 2019	none	average-hard	
$\ni$ SHUFFLE-DYCK-k	Bhattamishra et al. 2020a	none	softmax, future mask	
$\supseteq$ SSCMs	Bhattamishra et al. 2020a	none	softmax, future mask	
$\ni$ DYCK-k	Yao et al. 2021	i/n, i/n^3, n	softmax & leftmost-hard	
$\supseteq$ P	Pérez et al. 2021	i, $1/i$, $1/i^2$	average-hard	poly(n) steps
$\ni$ PARITY	Chiang and Cholak 2022	i/n, $(-1)^i$	softmax	
$\supseteq$ FOC[MOD; +]	Chiang et al. 2023	sinusoidal	softmax	
$\supseteq$ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
$\supseteq$ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	

Upper bound	Source	Precision	Attention	Notes
$\not\ni$ PARITY, DYCK-1	Hahn 2020	$\mathbb{R}$	leftmost-hard	
$\not\ni$ PARITY, DYCK-2	Hahn 2020	$\mathbb{R}$	softmax, future mask	$\varepsilon_N > 0$, vanishing KL
$\subseteq$ AC^0	Hao et al. 2022	$\mathbb{Q}$	leftmost-hard	
$\subseteq$ TC^0	Merrill et al. 2022	$\mathbb{F}$	average-hard	
$\subseteq$ FOC[MOD; +]	Chiang et al. 2023	$O(1)$	softmax	
$\subseteq$ L-uniform TC^0	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
$\subseteq$ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
$\subseteq$ L-uniform TC^0	Strobl 2023	$\mathbb{F}$	average-hard	

Equivalent	Source	PE	Attention	Notes
= RE	Pérez et al. 2021	i, $1/i$, $1/i^2$	average-hard	unbounded steps
= FO	Angluin et al. 2023	none	rightmost-hard, strict future mask	
= FO[MOD]	Angluin et al. 2023	sinusoidal	rightmost-hard, strict future mask	
= FO[Mon]	Angluin et al. 2023	arbitrary	rightmost-hard, strict future mask	
= P	Merrill & Sabharwal 2024	none	average-hard, future mask	poly(n) steps

Open Issues in Expressivity

Comparing the Variants

- Can average-hard attention transformers simulate unique-hard?
- Can softmax-attention transformers simulate unique-hard?
- Can softmax-attention transformers simulate average-hard? Or the other way around?
- Is there a transformer variant that is trainable, yet still easier to analyze than softmax?
- What happens if we vary the feed-forward layer? For example, using GeLU activations allow us to compute position-wise multiplication (approximately).

Precision

- Are $O(1)$-precision transformers equivalent to $\mathsf{FO} = \mathsf{LTL}$?
- Does the $\mathsf{K}_t[\#, +]$ lower bound apply to $O(\log n)$ precision transformers?
- At infinite precision, is it possible to find an upper bound?
- Do we need to consider infinite precision? That is, is there a difference between $O(\log n)$ and infinite precision? Or even $poly(n)$ precision?

Tighting Bounds

- At $O(\log n)$ precision, every operation except summation is in $\mathsf{FO}[+, \times]$. Is there a tighter upper bound than $\mathsf{FOC}[+, \times]$?
- The $\mathsf{K}_t[\#, +]$ lower bound only considers uniform attention. Is there a tighter lower bound than this?
- Can we exactly characterize softmax-attention transformers in terms of logic or circuits? What about average-hard attention?

Your Questions on Expressivity!

Expressivity vs Learnability and Trainability

A common question

What is the relationship between *learnability* and *expressivity*?

Does one affect the other?

can you study expressivity without learnability, or vice versa?

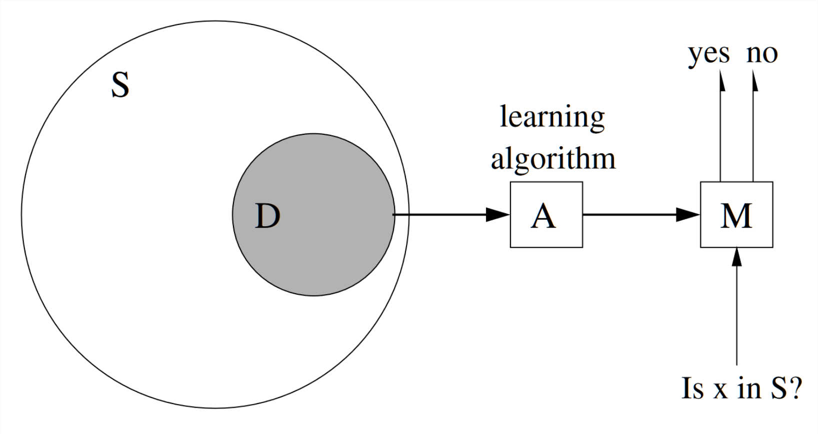
why or why not?

What are the ingredients of learning?

What are the ingredients of learning?

- a learner
- a thing to learn (a target)
- data
- hypotheses

Learnability Presupposes Expressivity [Rawski and Heinz, 2019]



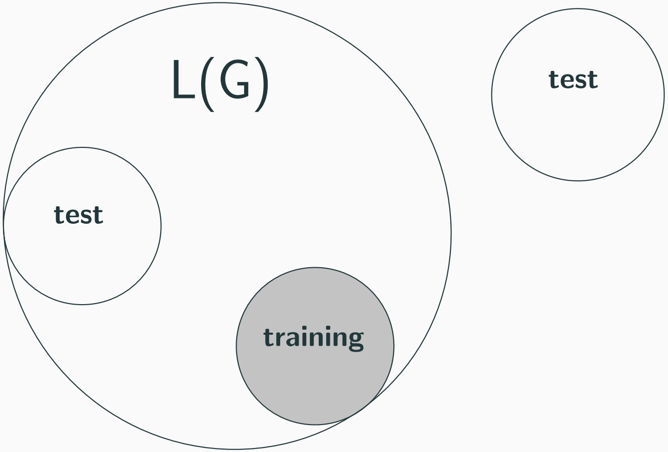
Learnability Presupposes Expressivity [Rawski and Heinz, 2019]

We also care about

> "circumstances under which these hypotheses stabilize to an accurate representation of the environment from which the evidence is drawn" [Osherson et al., 1986]

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

Artificial Grammar Learning Experiments



Data is drawn from a target/Intended set **I**

The subject is given a training/familiarization set **F**

The subject is then given a testing/Discrimination set **D**

Issues in AGL experiments [De Santo and Rawski, 2022]

Design

- Identifying relevant classes of patterns
- Finding minimal pairs of stringsets
- Finding sets of stimuli that distinguish those stringsets

Interpretation

- Identifying the class of patterns subject has generalized to
- Inferring the properties of the mechanism involved
  - ideally properties common to all mechanisms capable of identifying that class of patterns (like non-counting)

A simple learning experiment

Let's imagine a very simplified learning setup:

Our intended set I will be the context-free language A^nB^n

The dataset F will contain one string, i.e. $\{AABB\}$

What possible language(s) might a subject (human or machine) infer?

Obviously, one is our desired target A^nB^n.

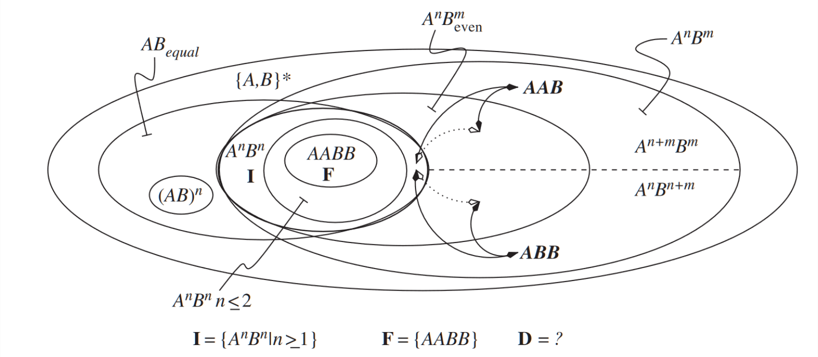
Any others?

Some possible generalizations

- memorization: $\{\text{AABB}\}$
- All of the A's precede all of the Bs: $A^n B^m$
- Same but they must be even length: $A^n B^m_{even}$
- At Least as many A's as B's: $A^n B^{n+m}$ or vice versa
- The number of A's equals the number of Bs: $|w|_a = |w|_b$
- The number of A's is finitely bounded: $A^n B^n, n \leq 2$
- Any combination of A's and B's: $\{A, B\}^*$
- Any even length combination of A's and B's: $\{A, B\}^*_{even}$

How is the learner supposed to decide between these?

Struggles with Generalization [Jäger and Rogers, 2012]



What should be in D?

We would like to decide between $A^n B^n$ and $A^n B^m$.

What kinds of test items would allow this?

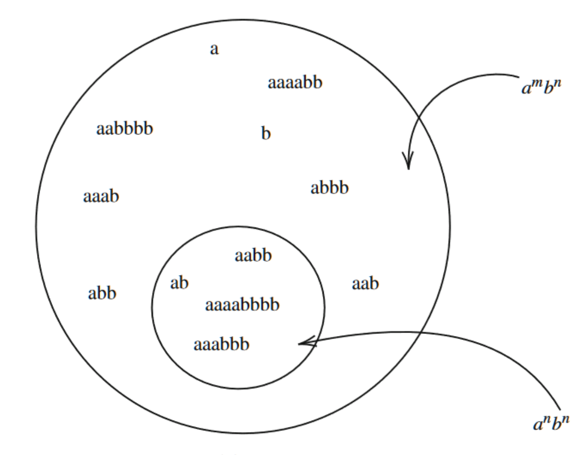
Recall, the participant has seen AABB

What should be in D?

We would like to decide between A^nB^n and A^nB^m.

What kinds of test items would allow this?

Recall, the participant has seen AABB



Testing ordered vs unordered languages

A common learning setup is to contrast 'ordered' vs 'unordered' languages

For example, Assume an alphabet of symbols $\Sigma = \{a, b, c, d, e\}$.

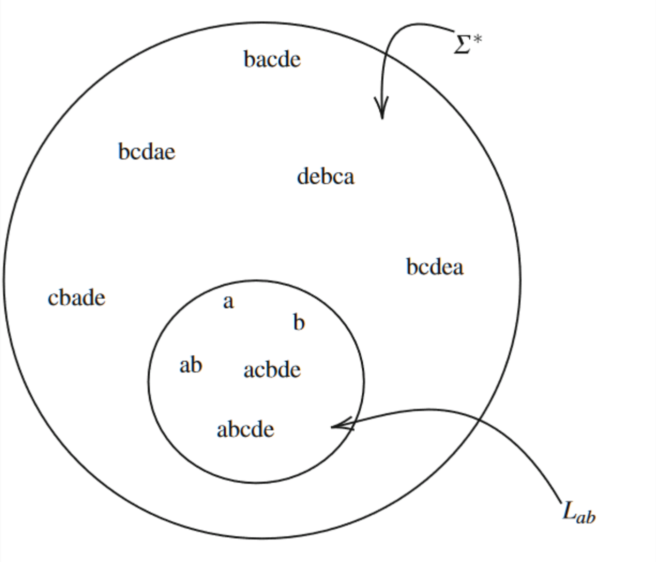
The language L_{ab}, only allows b after a (but not vice-versa)

L_{ab} would contain strings like $ab, acb, abcde$.

A 'free order' language contains sequences like ba, $bcdea$, $bacd$

Testing ordered vs unordered languages

This ‘free-order’ language is Σ^*, so it contains L_{ab} as well



Experimental Solution: Symmetric Differences in test sets

Remember: the subject has gotten AABB

Let's say **D** contains the test item AAB. Which hypotheses does this rule out?

Does this test the star-free boundary? or finite-state/regular?

Experimental Solution: Symmetric Differences in test sets

Remember: the subject has gotten AABB

Let's say we give them a test item AAB. Which hypotheses would accept/reject rule out?

- These are not in A^nB^n (CF) but are in the set A^nB^m (regular),
- Subjects that generalize to A^nB^n will find AAB surprising; those that generalized to A^nB^m will not.
- They are also not in $A^nB^n_{even}$, which is regular, but are in $A^{n+m}B^m$, which is CF.
- Thus these test/discrimination stimuli do not test the finite state boundary.
- what about $AAAB$?

An Example from RNNs

- Weiss et al. [2022] study how well Recurrent Neural Networks (RNNs) learn to recognize acceptable email addresses.
- The language of valid email addresses is a regular language, easily expressed with a DFA.
- One example from their paper: They trained an RNN to 100% accuracy on a 40,000 sample training set and a 2,000 sample test set.
- They refined a method to extract, from the learned RNN, a DFA approximation of it.
- Comparing the original and extracted DFA, they could find possible counterexamples.
- They find the RNN actually makes very stupid errors!

An Example from RNNs

Table 4. Counterexamples generated during extraction from an LSTM email network with 100% train and test accuracy. Examples of the network deviating from its target language are shown in bold.

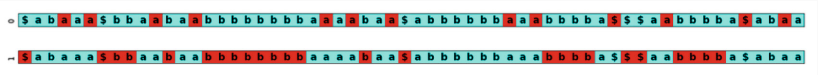
Counter-example	Time (s)	Network Classification	Target Classification
0@m.com	provided	$\surd$	$\surd$
@@y.net	2.93	$\times$	$\times$
25.net	1.60	$\surd$	$\times$
5x.nem	2.34	$\surd$	$\times$
0ch.nom	8.01	$\times$	$\times$
9s.not	3.29	$\times$	$\times$
2hs.net	3.56	$\surd$	$\times$
@cp.net	4.43	$\times$	$\times$

An Example from RNNs

- They note such cases are "annoyingly frequent: for many RNN-acceptors with 100% train and test accuracy on large test sets, our method was able to find many simple misclassified examples."
- They state this reveals the "brittleness in generalization" of trained RNNs,
- they suggest that evidence based on test-set performance "should be interpreted with extreme caution."

Another example [Oliva and Lago-Fernández, 2019]

- RNN with only 2 neurons in its hidden state trained on "Even-A" language.
- Input: stream of strings separated by $ symbol
- Neuron 0: all even As, and $ symbol after a rejected string
- Neuron 1: all B's following an even number of A's, and $ after an accepted string.



Expressivity of Recurrent neural Networks

Theorem ([Rabusseau et al., 2019])

Weighted FSA are expressively equivalent to second-order linear RNNs (linear 2-RNNs) for computing functions over sequences of discrete symbols

Theorem ([Merrill et al., 2020])

'saturated' RNNs accept exactly the regular languages

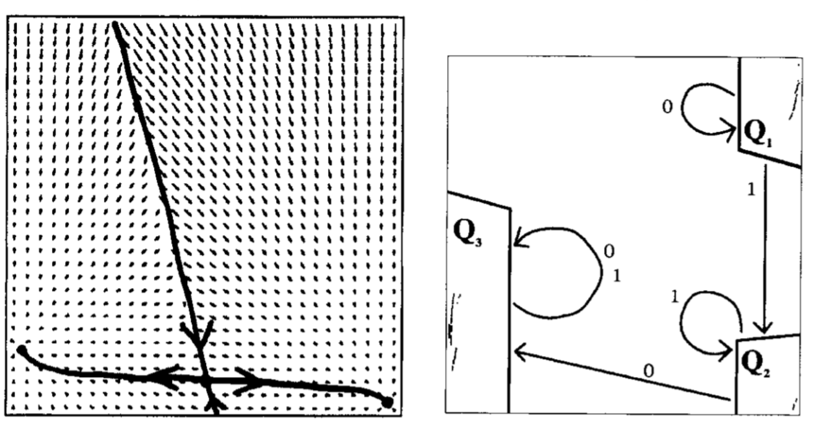
Theorem ([Casey, 1996])

A finite-dimensional RNN can robustly perform only finite-state computations.

Expressivity of Recurrent neural Network

Theorem

An RNN with finite-state behavior necessarily partitions its state space into disjoint regions that correspond to the states of the minimal FSA



Learnability vs Trainability

What does it mean for a transformer to be 'learnable'?

It is more helpful to distinguish learnability from trainability.

A transformer can be trainable or not, for some variety of functions

A formal language can be learnable or not, for some variety of model

The difference is mostly between learning vs optimization

Are transformers trainable?

For some varieties, clearly yes.

What about the varieties we considered?

We explicitly designed our models to probe expressivity, without caring about their trainability.

Are UHATs trainable?

How would one make them trainable?

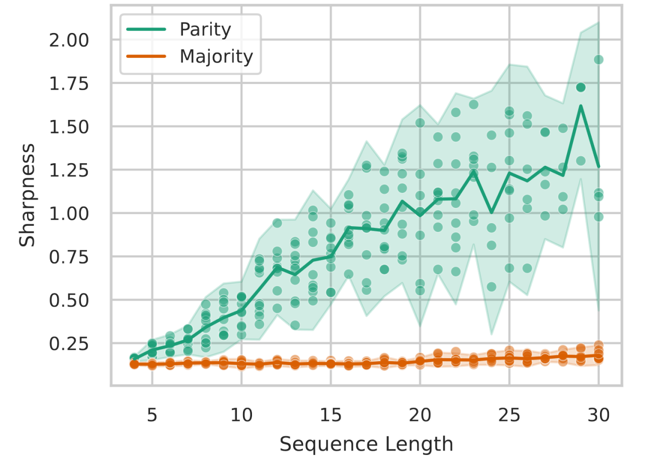
What about AHATs? SMATs?

Some Recent work on Transformer Trainability

- **Transformers Learn In-Context by Gradient Descent**
  [Von Oswald et al., 2023]
- **Transformers learn through gradual rank increase**
  [Abbe et al., 2024]
- **Linear attention is (maybe) all you need (to understand transformer optimization)**
  [Ahn et al., 2023]
- **One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention**
  [Mahankali et al., 2023]
- **Why are sensitive functions so hard for transformers?**
  [Hahn and Rofin, 2024]
- **Inductive Biases and Variable Creation in Self-Attention Mechanisms**
  [Edelman et al., 2022]

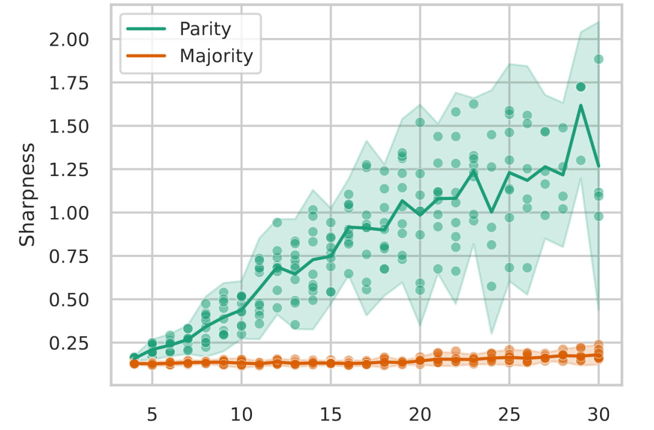
Hahn & Rofin 2024

- We noted that UHAT and SMAT can represent PARITY
- However, trainable transformers find a difficulty with PARITY (a sharp loss landscape)
- PARITY is sensitive: flipping any bit flips the string's parity



Hahn & Rofin 2024

- H&R prove that transformers whose output is sensitive to many parts of the input string inhabit isolated points in parameter space
- this leads to a low-sensitivity bias in generalization.
- this holds even with finite precision, hard attention, and Lipschitz-continuous layer norm.



The Moral of the Learning Story

- Learnability is subtly different than trainability
- Expressivity is a precursor to learnability
- Expressivity complements trainability

Final Words

Thanks for the Great Course from All of Us!

References i

Emmanuel Abbe, Samy Bengio, Enric Boix-Adsera, Etai Littwin, and Joshua Susskind. Transformers learn through gradual rank increase. *Advances in Neural Information Processing Systems*, 36, 2024.

Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization). *arXiv preprint arXiv:2310.01082*, 2023.

Mike Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural computation*, 8(6):1135–1178, 1996.

Aniello De Santo and Jonathan Rawski. Mathematical linguistics and cognitive complexity. In *Handbook of Cognitive Mathematics*, pages 1–38. Springer, 2022.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5793–5831. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/edelman22a.html`.

Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *arXiv preprint arXiv:2402.09963*, 2024.

Gerhard Jäger and James Rogers. Formal language theory: refining the chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970, 2012.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*, 2023.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. A formal hierarchy of RNN architectures. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 443–459, Online, July 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.43. URL `https://aclanthology.org/2020.acl-main.43`.

Christian Oliva and Luis F Lago-Fernández. On the interpretation of recurrent neural networks as finite state machines. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Theoretical Neural Computation: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part I 28*, pages 312–323. Springer, 2019.

References ii

Daniel N Osherson, Michael Stob, and Scott Weinstein. *Systems that learn: An introduction to learning theory for cognitive and computer scientists.* The MIT Press, 1986.

Guillaume Rabusseau, Tianyu Li, and Doina Precup. Connecting weighted automata and recurrent neural networks through spectral learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1630–1639. PMLR, 2019.

Jonathan Rawski and Jeffrey Heinz. No free lunch in linguistics or machine learning: Response to pater. *Language*, 95(1):e125–e135, 2019.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples (extended version). *Machine Learning*, 113(5):2877–2919, 2022.