Report from Dagstuhl Seminar 25282

Theory of Neural Language Models

Pablo Barcelo[*1], David Chiang[*2], George Cybenko[*3], Lena Strobl[*4], and Andy Yang[†5]

1 **PUC – Santiago de Chile, CL.** pbarcelo@ing.puc.cl
2 **University of Notre Dame, US.** dchiang@nd.edu
3 **Dartmouth College Hanover, US.** george.cybenko@dartmouth.edu
4 **University of Umeå, SE.** lena.strobl@gmail.com
5 **University of Notre Dame, US.** ayang4@nd.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 25282 "Theory of Neural Language Models". The seminar aimed to bring researchers together to lay a foundation for continued work on the theory of neural language models, focusing on questions including: How do transformers, RNNs, other NLMs, and their variants, compare with one another in expressivity and trainability? How do the successes and failures of NLMs predicted by theoretical models manifest in practice? What modifications, or what wholly new architectures, are suggested by the theory?

Seminar July 06–11, 2025 – https://www.dagstuhl.de/25282
2012 ACM Subject Classification Computing methodologies → Artificial intelligence; Theory of computation → Formal languages and automata theory; Theory of computation → Logic; Theory of computation → Circuit complexity; Theory of computation → Communication complexity
Keywords and phrases Dagstuhl Seminar, Neural Networks, Language Models, Automata, Logic, Model Theory, Circuit Complexity
Digital Object Identifier 10.4230/DagRep.15.7.22

1 Executive Summary

Pablo Barcelo (PUC – Santiago de Chile, CL)
David Chiang (University of Notre Dame, US)
George Cybenko (Dartmouth College Hanover, US)
Lena Strobl (University of Umeå, SE)

Artificial intelligence (AI) has gone through multiple "summers" and "winters," with the current summer based on large neural models that generate text, images, and other content. ChatGPT and other neural language models (NLMs), which model sequences of tokens, have taken center stage, not only in natural language processing, but across a wide range of applications. This interest spans the academic, corporate, government, investor, and consumer sectors. However, whereas experimental research and product development are surging ahead, more theoretical research aimed at foundational questions about neural networks is lagging behind. Clear thinking about what NLMs can and can't do is more needed than ever.

[*] Editor / Organizer
[†] Editorial Assistant / Collector

Theory of Neural Language Models, *Dagstuhl Reports*, Vol. 15, Issue 7, pp. 22–52
Editors: Pablo Barcelo, David Chiang, George Cybenko, Lena Strobl, and Andy Yang
Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The old guard of NLMs, recurrent neural networks (RNNs), have been studied theoretically for decades, in relation to finite automata and Turing machines. At present, the dominant NLMs are based on transformers, whose computational power is a new and rapidly growing area of research. Transformers have been related to a wide variety of formal models from computability and complexity theory, like counter automata, Turing machines, Boolean circuits, and first-order logic. However, a unified and comprehensive theory of the abilities and limitations of transformers is not yet in sight.

Such a theory would ideally answer questions like:

- How do transformers, RNNs, other NLMs, and their variants, compare with one another in expressivity and trainability?
- How do the successes and failures of NLMs predicted by theoretical models manifest in practice?
- What modifications, or what wholly new architectures, are suggested by the theory?

There is a small but growing community of researchers that investigates such questions about NLMs. This Dagstuhl Seminar aimed to bring this community together to lay a foundation for continued work in this area, identifying central open problems, and fostering new collaborations. To achieve these goals, the seminar left ample room for informal discussions on topics suggested by the participants themselves, along the lines of an Open Space Technology meeting.

2 Table of Contents

3 Tutorials

3.1 Transformers

David Chiang (University of Notre Dame, US)

On Day 1, I gave an introduction to transformers, going over the basic definition briefly. (Andy Yang later gave a more detailed introduction for participants coming from outside AI.) Then I gave an overview of results proving exact equivalence between variants of transformers and various complexity classes, and focused on three themes that I thought would provoke discussion:

- The equivalence results naturally fall into two groups: on the one hand, equivalences between transformers and $\mathsf{FO}[<]$-uniform complexity classes; and on the other hand, equivalences between transformers *with intermediate steps* and DLOGTIME-uniform ($= \mathsf{FO}[+, \times]$-uniform) complexity classes. (Thanks to Michaël Cadilhac for this formulation.) The use of intermediate steps, then, is a particularly important variation.
- Another important variation is in *uniformity* or aspects of the network depending on n, the input length. I distinguished between "parameter-uniformity," where the parameters do not depend on n but other aspects might, and setups where the parameters may also depend on n.
- A very large number of variations can be seen as workarounds for a result by [1]: If a transformer uses softmax attention, has bounded position embeddings, and has only Lipschitz-continuous position-wise functions, then a change in a single input symbol results in an $O(1/n)$ change in the output, where n is the input length. If, furthermore, a minimum gap is required between different outputs (e.g., accept vs. reject), then a transformer can only solve trivial problems. So, many constructions use tricks to work around one of the above assumptions.

This overview was based in part on the survey by [2].

References

1 Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://aclanthology.org/2020.tacl-1.11.

2 Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024. doi: 10.1162/tacl_a_00663.

3.2 Theoretical Background

Howard Straubing (Boston College, US)

I discussed classes of formal languages – some classes of regular languages, classes defined by different kinds of logical formulas (variants of first order and temporal logic), and circuit complexity classes – that have been studied for many years by theoretical computer scientists, and that are receiving renewed attention because of their connection to the expressive power of transformers.

Regular languages:

$$\begin{aligned}\mathsf{FO}[<] &= \text{star free} = \mathsf{LTL} \text{ (linear temporal logic)}\\ &\subsetneq \mathsf{FO}[<, \mathsf{MOD}] \text{ (modular predicates)}\\ &\subsetneq \mathsf{FO{+}MOD}[<] \text{ (modular quantifiers)}\\ &\subsetneq \text{regular languages.}\end{aligned}$$

Circuit complexity classes:

$$\begin{aligned}\mathsf{AC}^0 &= \mathsf{FO}[\mathsf{Any}]\\ &\subsetneq \mathsf{ACC} = \mathsf{FO{+}MOD}[\mathsf{Any}]\\ &\subsetneq^{?} \mathsf{TC}^0 = \mathsf{Maj}[\mathsf{Any}] \text{ (majority quantifiers)}\\ &\subsetneq^{?} \mathsf{NC}^1\end{aligned}$$

where $\subsetneq^{?}$ means that inclusion is known, but strictness is not known.

I also discussed the uniform versions of the circuit complexity classes. Questions of inclusion among these classes are addressed by algebraic means, especially via the syntactic monoids of regular languages.

3.3 Methods and Architectures to Increase Expressivity

William Merrill (New York University, US)

We have seen how transformers (when used as classifiers or next-token predictors) cannot express inherently sequential computation outside the class TC^0. In contrast, many applications of LLMs, e.g., state tracking or reasoning about math or code, conceivably require sequential processing. This tutorial covered two ways to extend transformers' expressive power so they can solve inherently sequential computational problems:

1. Extending transformers with **chain of thought** allows them to simulate Turing machine steps, enabling computation outside TC^0 with enough chain-of-thought steps. However, this sacrifices parallelism in order to gain sequential expressive power.
2. Moving away from transformers, **linear RNNs** (a.k.a. state-space models) can gain some expressive power outside TC^0 while maintaining moderate parallelism on current hardware.

3.3.1 Chain of thought (CoT)

We view a transformer as a function $T : \Sigma^* \to \Sigma$ by taking the argmax logits. Without CoT, we solve a decision problem with T by simply computing $T(w)$ on input w and interpreting specific tokens as yes/no. With CoT, we allow T to autoregressively generate t "thinking" tokens before generating a final token $t + 1$ as yes/no output.

So, the full context generated by CoT looks like:

$$\underbrace{x_1 \dots x_n}_{\text{Input tokens}} \quad \underbrace{z_1 \dots z_t}_{\text{CoT tokens}} \quad \underbrace{y}_{\text{Output}}$$

▶ **Definition 1.** For any function $t(n)$, let $\mathsf{CoT}[t]$ be the class of problems expressible by a transformer that is allowed $O(t(n))$ steps of CoT on inputs of length n.

For example, $\mathsf{CoT}[\log n]$ represents *logarithmic CoT* and $\mathsf{CoT}[n]$ represents *linear CoT*. We will also refer to $\cup_{c=0}^{\infty}\mathsf{CoT}[n^c]$ as *polynomial CoT*.

3.3.1.1 Simulating Turing machines with CoT

Before seeing how transformers can use CoT to simulate Turing machines, we briefly review multitape Turing machines:

- Finite state, read-only input tape, and several work tapes
- At step i, each tape τ has contents Γ_i^τ and head position h_i^τ
- The Turing machine is specified a transition function δ that reads the current symbol on each tape head as well as some finite state and returns a new state and new output symbols and move directions on each tape. That is, for states $q, q' \in Q$, tape symbols $\gamma^\tau, \gamma'^\tau \in \Gamma$, and move directions $m^\gamma \in \{-1, 0, 1\}$,

$$\delta : q, \gamma^0, \gamma^1, \ldots \gamma^k \mapsto q', \gamma'^0, \gamma'^1, \ldots \gamma'^k, m^0, m^1, \ldots, m^k$$

- Having computed δ, we update the finite state, tape contents, and head position:

$$\begin{aligned} q_{i+1} &= q'_i \\ \Gamma^\tau_{i+1}[h^\tau_i] &= \gamma'^\tau_i \\ h^\tau_{i+1} &= h^\tau_i + m^\tau_i \end{aligned}$$

▶ **Theorem 2** ([1, 2])**.** *CoT transformers can simulate multitape Turing machines. That is, for any $t(n)$, $\mathsf{TIME}[t] \subseteq \mathsf{CoT}[t]$.*

Proof sketch. The idea will be to maintain an immutable representation of the Turing machine tape distributed across CoT tokens. Each token will store the symbols written to the tape at the previous step, as well as a pointer to the current head position. With this information, future CoT tokens can always attend back to reconstruct the last symbol written to a particular position.

We proceed by induction, showing that CoT token i can encode q_i, γ_i^τ (the token written by step $i-1$), and m_i (the move after step $i-1$).

1. **Compute Head Pointer:** At each token i, we will recover h_i^τ, the head position on tape τ at step i. This can be done by counting the number of right and left moves on each tape, i.e., as $\sum_{j=1}^{i} m_j$.
2. **Retrieve Tape Symbol:** Once we have h_i^τ, we recover the current symbol under head τ via an "induction head" [3] construction: we will find the last token where tape τ was at h_i^τ and retrieve the token outputted immediately after that. To do this, the head will hard attend from step i to the last step j such that $h_i^\tau = h_{j-1}^\tau$ and retrieve the CoT token δ_j.
   In more detail, one way to implement this head with saturated attention is as follows. Adapting [2], let $\phi(x)$ be the projection of $\langle x, 1\rangle$ onto the unit sphere and let $f(i)$ be small and monotonically decreasing.
   - **Query i:** $\langle \phi(h_i^\tau), -1\rangle$
   - **Key j:** $\langle \phi(h_{j-1}^\tau, 1), \phi(f(i))\rangle$
   - **Value j:** γ_j^τ, i.e., the token written to h_{j-1}^τ

This head returns γ_j^τ from the largest j such that $h_i^\tau = h_{j-1}^\tau$, i.e., the current symbol on tape τ at position h_i^τ.

3. **Compute Transition Function:** Once we have computed the symbol γ_i^τ on each tape, we can use a feedforward network to compute the following transition function via a finite lookup table:

   $$\delta(q_i, \gamma_i^0, \gamma_i^1, \ldots, \gamma_i^k).$$

   We then output a single CoT token that encodes the symbols that should be written and directions that should be moved – this means the vocab size increases with the number of state and tapes but is fixed w.r.t. n.

We conclude that we can simulate a Turing machine for t steps using t CoT tokens. ◀

Assuming $t \leq \text{poly}(n)$, this construction only requires $O(\log n)$ precision, since the growing tape is distributed across different tokens. This is different than the classical RNN Turing completeness construction for RNNs [4], which requires the precision to grow linearly with runtime.

3.3.1.2 Understanding how much CoT is required for more expressive power

We can consider the expressive power of transformers with different amounts of CoT. Immediately, we see that polynomial CoT allows transformers to solve any problem solvable in polynomial time:

▶ **Corollary 3.** $\bigcup_{c=0}^{\infty} \mathsf{CoT}[n^c] = \mathsf{P}$.

But this is a lot of CoT. Do we still gain expressive power with shorter CoT lengths? Yes, with less CoT, we still can solve some problems likely outside TC^0:

▶ **Corollary 4** ([2, 5])**.** $\mathsf{CoT}[n]$ *contains* NC^1*-complete problems including regular language recognition and boolean formula evaluation.*

What about sublinear CoT? Interestingly, logarithmic CoT remains in TC^0:

▶ **Theorem 5** ([6])**.** $\mathsf{CoT}[\log n] \subseteq \mathsf{TC}^0$.

Proof sketch. We aim to simulate a transformer that uses $c \log n$ CoT steps with a TC^0 circuit family. To do this, we enumerate all possible CoT's using $O(n^c)$ constant gates. In parallel for each token of each possible CoT, we apply a TC^0 circuit (corresponding to the transformer) that checks whether the current token would have been produced by its left context. We select the unique CoT where each token matches the transformer's output. ◀

So more than logarithmic CoT is required to gain expressive power outside TC^0.

An interesting open question is proving lower bounds on CoT length against transformers. [7] make exciting progress on this assuming hard-attention transformers.

3.3.1.3 Alternatives to CoT: padding and looping

Padded transformers: augment the transformer with a CoT of blank tokens ($\square$) rather than model-generated tokens. Unlike standard CoT, this is parallelizable because we don't need to wait for the output of the first t tokens before processing token $t + 1$.

▶ **Theorem 6** ([8, 9, 10])**.** *Soft-attention transformers with polynomial padding tokens express exactly* TC^0.

Looped transformers: rather than autoregressively generating tokens, just repeat layers in the transformer. Also called "universal transformers".

▶ **Theorem 7** ([11, 12]). *Looped transformers can solve the* NC^1*-complete problem of regular language recognition.*

Moreover, transformers with polylogarithic looping and polynomial padding can express all of NC, showing that these methods can unlock substantial expressivity while preserving moderate parallelism (unlike CoT).

3.3.1.4 Takeaways about CoT and related methods

- CoT gains expressivity but sacrifices parallelism and requires many steps!
- Padding is parallelizable but doesn't gain expressivity outside TC^0
- Looping gains expressivity even with a small number of steps: relatively parallelizable
- Parallelism tradeoff: expressivity for sequential problems and parallelism are at odds

3.3.2 New architectures: linear RNNs

Rather than relying on CoT for greater expressive power, can we design new architectures that increase expressivity while maintaining parallelism? To get sequential expressivity, we could look back to RNNs:

▶ **Definition 8** (Nonlinear RNN)**.** For some nonlinearity σ, an RNN has the form

$$h_{i+1} = \sigma(Ah_i + Bx_i).$$

These RNNs can represent the NC^1-complete problem of recognizing regular languages [13]. But the nonlinearity means that it's not easy to parallelize RNNs.

A new line of work has therefore turned to linear RNNs (also called state-space models):

▶ **Definition 9** (Linear RNN)**.**

$$h_{i+1} = A_i h_i + Bx_i.$$

Relative to old RNNs, linear RNNs are parallelizable. Relative to transformers, they use less memory have the potential for greater expressivity on sequential problems.

S4 and Mamba are particular instantiations of linear RNNs (there are many others). Linear attention is also related:

▶ **Definition 10** (Linear attention)**.**

$$h_i^\top = \sum_{j=1}^{i} q_i^\top k_j \cdot v_j^\top.$$

Linear attention can be written as the following linear RNN:

$$\begin{aligned} S_{i+1} &= S_i + k_j v_j^\top \\ h_i^\top &= q_i^\top S_i. \end{aligned}$$

3.3.2.1 Parallelizing linear RNNs

Leveraging linearity, we can rewrite a linear RNN in its "convolutional form":

$$h_{i+1} = A_i h_i + Bx_i$$
$$h_i = \sum_{j=1}^{i} A_i A_{i-1} \dots A_{j+1} B x_j.$$

This can be computed efficiently as a prefix sum in the monoid generated by $\langle A_i, Bx_i \rangle$ where $\oplus$ is defined[1]

$$\langle A_1, b_1 \rangle \oplus \langle A_2, b_2 \rangle = \langle A_2 A_1, A_2 b_1 + b_2 \rangle.$$

Using efficient algorithms for prefix sums [14], we can compute this efficiently.

3.3.2.2 Expressivity benefits of linear RNNs

Does the recurrent nature of linear RNNs enable them to represent NC^1-complete problems like regular language recognition? For several popular early instantiations of linear RNNs, the answer is no, but, in general, it is possible.

▶ **Theorem 11** ([15]). *If $A_i = A$ (e.g., in S4), then a linear RNN is in TC^0.*

Proof. Computing the linear RNN reduces to

$$h_i = \sum_{j=1}^{i} A^{i-j} B x_j.$$

Iterated addition and matrix powering are in TC^0, so this can be computed in TC^0. ◀

▶ **Theorem 12** (Informal; [15]). *If A_i is diagonal and parameterized reasonably (e.g., in Mamba), then a linear RNN is in TC^0.*

Proof sketch. Since A_i's are diagonal, computing the product $A_i A_{i-1} \dots A_{j+1}$ reduces to iterated multiplication of scalars in parallel, which is commutative and in TC^0. ◀

However, it's possible to make linear RNNs expressive enough for regular language recognition by allowing A_i to be more complex:

▶ **Theorem 13** ([15]). *Let A_i be a learned embedding of token σ_i (call this IDS4). For any regular language, there exists an IDS4 linear RNN that recognizes it.*

Proof sketch. We can directly encode FSA transitions with such a linear RNN. We assume a beginning of sequence symbol \$. We set B\$ to return a representation of the initial state (a one-hot diagonal matrix). We use A_i to represent the transition monoid element δ_i associated with σ_i. The state is thus

$$h_i = (\delta_i \circ \delta_{i-1} \circ \dots \circ \delta_1)(q_0).$$

We can then use a final linear layer to detect whether h_i represents an accepting state. ◀

[1] Nice presentation here: `https://openreview.net/pdf?id=RDbuSCWhad`

▶ **Corollary 14.** *IDS4 can solve an* NC^1*-complete problem.*

Downside: letting A_i be an arbitrary matrix requires $O(d^2)$ memory instead of $O(d)$ for a diagonal matrix.

Recent work has come up with new A_i parameterizations that use less (i.e., linear) memory but still can recognize regular languages. Some of these linear RNNs have been scaled up recently with encouraging signs!

- DeltaNet++ [16]: an extension to DeltaNet linear RNN similar to IDS4 unlocks greater expressivity and improves math/code reasoning at the 7B scale
- RWKV-7 [17]: 7B linear RNN that can express regular language recognition and achieves strong LM performance
- PaTH attention [18]: an extension to transformers inspired by DeltaNet extension that increases expressivity, synthetic evaluations, length generalization, and language modeling performance

3.3.2.3 Expressivity drawbacks of linear RNNs

Major drawback of linear RNNs relative to transformers: they cannot copy or retrieve arbitrary tokens from the past due to their bounded memory [19]

- As an implication, they cannot solve associative recall [20] or form induction heads [3], which have been shown to be important for language modeling
- The Turing machine simulation construction in Theorem 2 utilizes induction heads, so linear RNNs can't implement it

Potential solutions:

- Mix linear RNN layers with at least one transformer layer for retrieval
- Augment linear RNNs with some other retrieval mechanism

References

1. Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021. URL `http://jmlr.org/papers/v22/20-302.html`.
2. William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=NjNGlPh8Wh`.
3. Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022. URL `https://arxiv.org/abs/2209.11895`. arXiv:2209.11895.
4. H.T. Siegelmann and E.D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995. doi: `10.1006/jcss.1995.1013`. URL `https://www.sciencedirect.com/science/article/pii/S0022000085710136`.
5. Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind Chain of Thought: A theoretical perspective. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, pages 70757–70798, 2023. URL `https://papers.nips.cc/paper_files/paper/2023/hash/dfc310e81992d2e4cedc09ac47eff13e-Abstract-Conference.html`.

6 Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3EWTEy9MTM.

7 Alireza Amiri, Xinting Huang, Mark Rofin, and Michael Hahn. Lower bounds for chain-of-thought reasoning in hard-attention transformers. In *Proceedings of ICML*, 2025. URL https://arxiv.org/abs/2502.02393.

8 Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NikbrdtYvG.

9 William Merrill and Ashish Sabharwal. Exact expressive power of transformers with padding, 2025a. URL https://arxiv.org/abs/2505.18948. arXiv:2505.18948.

10 Charles London and Varun Kanade. Pause tokens strictly increase the expressivity of constant-depth transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.21024.

11 Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

12 William Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers, 2025b. URL https://arxiv.org/abs/2503.03961. arXiv:2503.03961.

13 William Merrill. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, August 2019. doi: 10.18653/v1/W19-3901. URL https://aclanthology.org/W19-3901/.

14 Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

15 William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QZgo9JZpLq.

16 Riccardo Grazzi, Julien Siems, Jörg K.H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear RNNs through negative eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

17 Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. RWKV-7 "Goose" with expressive dynamic state evolution, 2025. URL https://arxiv.org/abs/2503.14456. arXiv:2503.14456.

18 Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. PaTH attention: Position encoding via accumulating Householder transformations, 2025. URL https://arxiv.org/abs/2505.16381.

19 Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *Proceedings of the 41st International Conference on Machine Learning*, pages 21502–21521, 2024. URL https://proceedings.mlr.press/v235/jelassi24a.html.

20 Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 1763–1840, 2024. URL https://proceedings.mlr.press/v235/arora24a.html.

3.4 Industry Applications and Multi-step Reasoning

Clayton Sanford (Google – New York, US)

This tutorial discussed connections (existing and potential) between applied LLM research and foundational theory of Transformers, with a particular focus on multi step reasoning, and the communication lens.

I first provided a list of guiding questions for empirically-influenced theory. These included questions related to *reasoning capabilities* (What are the "correct" reasoning primitives to learn and how can they be evaluated? How are these capabilities impacted by a shift from single-pass transformer inference to multiple prompts); *scaling laws* (How can theoretical results be framed as functions of compute budget, rather than context length? Can theoretical results incorporate power law assumptions?); and *model trade-offs* (What guidance can be provided for models operating under different resource constraints, e.g., on-device models whose train compute budget is less constrained than its parameter count, models that require regular training updates).

I then introduced multi-step reasoning (instantiated by the k-hop induction heads task) and provided a series of communication models that exploit capacity constraints in different modeling regime. *Two-party communication bounds* establish the hardness of expressing tasks like induction heads with single-layer transformers. The *blackboard communication protocol* is a multiparty communication model that establishes the limitations of subquadratic models (e.g. linear attention, kernel attention) to solve multi-step tasks. The *CONGEST* framework establishes the limitations of GNNs. The *Massively Parallel Computation (MPC)* model has a bidirectional relationship with deep transformers and provide provides (conditional) lower bounds on multi-step reasoning.

I concluded with two research vignettes.

- An explanation of a *star graph path detection* problem with hard observed learnability.
- A theoretical result that establishes separations for *mixture of experts (MoE)* models.

4 Working Groups

4.1 The Big Picture

Noah A. Smith (University of Washington – Seattle, US), George Cybenko (Dartmouth College Hanover, US), Ashish Sabharwal (Allen Institute for AI – Seattle, US), Gail Weiss (EPFL – Lausanne, CH)

This working group focused on the challenge of guiding theoretical researchers toward questions of relevance to those building modern language model-based systems. The group met for one day (Monday) before dispersing, and identified a few potential areas where more attention might lead to useful results. Some of these align well with longstanding ways of thinking about how theory can serve practice:

- Proofs of impossibility; identify capabilities that are unachievable, so that applied researchers do not waste time building systems to achieve them.
- Efficiency of learning: shedding light on design choices that are predicted to learn a capability most efficiently.

Other directions are more focused on specific practices that are being applied widely, or on desiderata that are being sought widely, in the language modeling community today:

- Theory of distillation (where a model is trained to simulate another model's behavior).
- Theory for interpretability, e.g., mapping a learned transformer into rules.
- Theory for capabilities of language models, e.g., developing abstractions for language model computation that are grounded in theory rather than analogy to human cognition, or theories that explain patterns of task interaction and interference during learning.
- Theory for evaluation that helps move evaluation practices toward better interpretability of model behaviors.
- Theory for training models (e.g., for scaling laws).
- Theory of large collections of data as they relate to learning and capabilities.

A general observation was that, in most scientific fields, "theory" refers to an evolving collection of propositions that explain phenomena, which is used to generate hypotheses that can be tested, and whose results then lead to updates to the theory. The consensus was that this loop is not yet well established; large-scale experimental work is driven more by the goal of improving benchmark performance than by advancing a theory of language models.

4.2 Learnability

Brian DuSell (ETH Zürich, CH), Gail Weiss (EPFL – Lausanne, CH), Michael Benedikt (University of Oxford, GB), George Cybenko (Dartmouth College Hanover, US), Robert Frank (Yale University, US), Anthony W. Lin (RPTU Kaiserslautern-Landau, DE), Paul S. Lintilhac (Dartmouth College Hanover, US), Jon Rawski (San José State University, US), Guillaume Rabusseau (University of Montreal, CA), Clayton Sanford (Google – New York, US), Noah A. Smith (University of Washington – Seattle, US), Lena Strobl (University of Umeå, SE), Andy Yang (University of Notre Dame, US)

This working group discussed theoretical and experimental approaches to understanding the learning dynamics of transformer networks. The group met from Monday through Friday, following an arc through theoretical to practically grounded discussions with varying participants.

Much theoretical work on neural networks has focused on their expressivity, i.e., the set of functions for which a parameterized network exists that implements them. Comparatively little work has formally characterized the *learnability* of functions by neural networks, i.e., the functions for which network network training algorithms have a reasonable probability of producing a network that implements them.

The discussion began with a recognition of the difficulty of capturing "learnability," especially as it applies to modern settings, e.g., the training of transformer neural networks with an adaptive, momentum-based optimizer (e.g., Adam). Rather than the more traditional setting of finding algorithms which could successfully learn a given task, we asked whether we could characterize the tasks learnable with a given algorithm. Simultaneously, we wondered whether it would be possible to characterize these tasks without getting completely entangled in the details of these complicated algorithms.

We discussed existing theoretical frameworks with which we could approach this problem, including in particular *sharpness* in the loss landscape of neural networks (the extent to which a minor perturbation in parameters will change the loss). We also discussed the expression of Boolean functions as linear combinations of sparse parities, properties of the class of *sensitive* languages (languages for which changing a single or small number of bits in an input is sufficient to change its classification) and *regular* languages (languages recognisable with a deterministic finite state automaton). We also discussed closure properties of learnability, for example the learnability of composed simple tasks.

On Thursday and Friday, we moved to exploring a case study involving two languages which individually are easy to learn, but very difficult when mixed together – despite the mixture permitting an apparently simple solution. The discussion was fruitful, raising multiple hypotheses and potential approaches to understanding the source of this added difficulty.

4.2.1 Discussed Problems

This group discussed problems related to loss landscapes (sharpness in parameter space), brittleness of transformer solutions, generalization, and interference. We detail our main discussions below.

4.2.1.1 Loss landscape of transformers computing regular languages

One broad question that was discussed is: how do different parameterizations of a DFA-simulating transformer affect the loss landscape around the solution?

More concretely, given a DFA, we can construct a transformer simulating it (up to some length n) using several techniques, for example:

- naive approach with n layers: each layer implements one transition of the DFA
- recursive scan approach with $\log n$ layers
- shortcut solution (if the DFA is solvable) with $\mathcal{O}(1)$ layers
- (if possible,) write a program recognizing the language in (some variant) of RASP and translate it into a transformer
- train a transformer on traces of the DFA

The first three approaches are described by [1].

Taking for granted that sharpness around the solution in parameter space is a good proxy for generalization ability / trainability, it is interesting to ask whether different parameterizations lead to different sharpness. Another dimension to consider is the structure of the DFA: how does its size, sparsity, number of components in the Krohn–Rhodes decomposition, etc., affect the sharpness of the different solutions. We hypothesized that in the case where a transformer permits multiple solutions to the same problem, the learning algorithm would prefer a solution with lower sharpness. However, we also considered that, due to the high complexity of the loss landscape, it may be possible that a specific data mix, task, or initialization could define a path into an otherwise sharp solution. We considered that sharpness may not tell the full story of learnability.

4.2.1.2 Interfering tasks when learning transformers

Gail shared an observation on two simple languages that are individually easy to learn, but together are much more difficult. These languages were the *copy* and *pairs* tasks, consisting of sequences of the sort ww for $w \in \Sigma^*$ (*copy*) and $w \in \Sigma_2^*$ for $\Sigma_2 = \{\sigma\sigma \mid \sigma \in \Sigma\}$ (*pairs*). We found that the union of these tasks (with a disambiguating prefix token) takes a long time

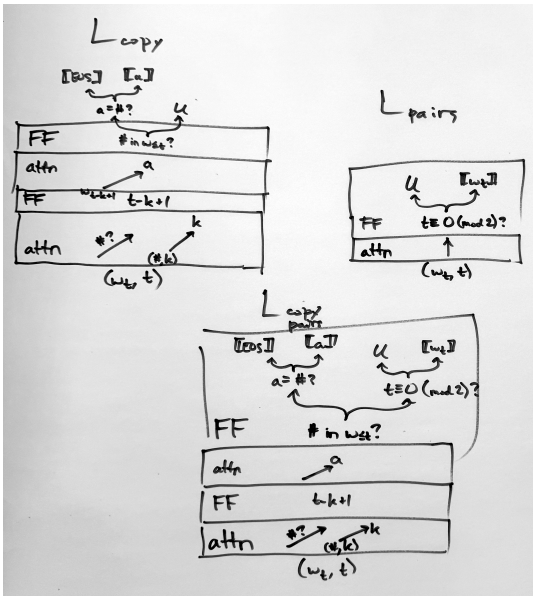

Figure 1 Simple transformers for copy (top left), pairs (top right), and copy-pairs (bottom). Sketch by Brian DuSell.

to learn jointly, much longer than the sum of their individual learning times. In particular, we observed that the pairs task is learned very quickly, while the copy task experiences significant delay (∼8x). However, when there is a warmup period where we learn only copy and then learn both, learning happens much faster, closer to what we would expect. We found that a composition of these languages called *copied pairs*, containing sequences of the sort $aabbcc :: aabbcc$, was also much slower to learn than its component tasks.

We found that both of these tasks are very easy to express in a transformer, and even described a simple solution for the copied pairs task in a simple 2 layer transformer. The trained transformers had 4 layers and 4 heads each, and generally sufficient representation space for the task, as evidenced by their ability to model it given enough time, as well as their ability to reach a solution quickly under helpful interventions. Hence the challenge must stem from some combination of the data presentation, learning algorithm, and model biases, rather than a limitation in expressive power. An interesting observation arose, in that once pairs was learned in these combined tasks (which would happen early on), it remained at near-zero loss for the duration of training, suggesting that from that point the learning algorithm was only able to explore the loss landscape where the pairs loss was low, restricting the exploration.

This is a concrete example of what we identified as a major open research question: **what kinds of formal languages have the property of being learnable under different kinds of compositions?** Said differently, which kinds of formal languages interfere with each other during training, either constructively or destructively? As a baseline, we can see that learning the copy task and reverse task together does not experience as much destructive interference. This question is completely separate from expressivity. But given some of the results on using formal languages in a learning curriculum to benefit natural language learning, it seems important to better understand these kinds of interference, and how it is affected by the order of training.

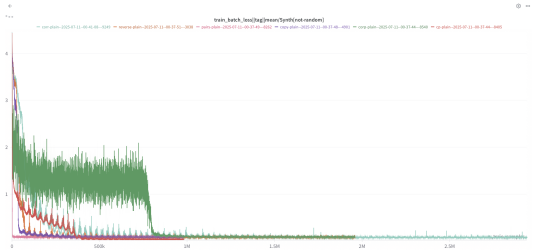

Figure 2 An image showing the loss over time (measured in number of training samples) while learning the combined copy-or-pairs task (green) as compared to just copy (purple) or just pairs (pink) tasks. The loss curve of the composed task, copied pairs, is shown in red. A brief exploration of reverse is also shown: reverse in orange, and copy-or-reverse in blue.

4.2.1.3 Other open problems & research questions

Other problems were discussed during the sessions of this group, including:

- Is the set of functions learnable by transformers closed under common operations (union, concatenation, etc.)?
- How sharpness can affect or be affected by decisions in training, in particular with how it can be involved in the training objective, and how it relates to batch size or other decisions.
- How architectural changes in a transformer can affect the learnability of different tasks, for example the use of rotary versus learned positional embeddings.

4.2.2 Possible Approaches

4.2.2.1 Loss landscape of transformers computing regular languages

For this question, two first steps were identified:

- Experiments. Collect a set of "interesting" automata (parity, flip-flop, bounded Dyck languages, etc.), translate each of them into a transformer using different constructions and estimate the sharpness of the loss landscape around the corresponding point in the parameter space. The sharpness can be estimated using a relatively naive Monte-Carlo approach (simply sample perturbed transfomers and compute the average loss difference over the train and test set). More efficient approaches could be investigated, such as random projections to estimate the trace of the Hessian.
- Theory. Deriving an analytical expression of the Hessian and its spectrum, which characterize the sharpness) is likely too difficult for even a small transformer construction with multiple layers. However, it would be interesting to formally analyze the sensitivity of the different building blocks used in different constructions. For example, in the shortcut solutions, each attention layer implements one of the simple automata from the Krohn–Rhodes decomposition while the MLP computes a form of parity function. Similarly, in the recursive scan solution, attention layers are used to map transition functions with one another while the MLP layers implement composition of these transitions. It should be possible to derive analytical results on the sensitivity of each of these building blocks to inform us on which one are more sensitive/brittle.

4.2.2.2 Case study on interference

Several interesting questions were raised to validate and explore the observation on copy-pairs interference, among them:

1. Training hyperparameters: to validate the observation, participants asked whether the interference holds even when when each task is hyperparameter-tuned? (i.e., could the delay be due simply to poorly adapted hyperparameters?)
2. Model Architecture: is the copy-pairs interference a special case of the architectures used in this observation, or does it hold for other sequence-processing models? In particular, does it hold both for transformers with learned and with rotary positional embeddings? And can a similar task pair be found for recurrent neural networks and their variants?
3. Task specifics: Was there something special about the composition of copy and pairs in the *copied pairsd*' task that encouraged the interference, which would not hold in compositions? For example, would pairing only on the left (sequences of the sort $aabbcc :: abc$) or right ($abc : aabbcc$) lead to similar levels of interference?
4. Model Generalisation: due to the nature of next-token based training, in which the model is only exposed to positive samples, a discussion arose around the exact rules the model is learning for its task. These can be explored via evaluating its response to out-of-domain inputs: for example, a well-trained *copied pairs* model, when presented with a malformed prefix such as $aabbcd ::$ (i.e., not adhering to left-pairing as expected), could reveal the model's tendency towards either pairs (generating *aabbcc*) or copy (generating *aabbcd*)?
5. Source of problem: the biggest question in the discussion was why this interference was happening at all, and what in the training was going wrong to delay copy learning. We raised several hypotheses, among them: degeneration of positional embeddings (as next-token predictions for the pairs task require only an understanding of the current token and current position's parity), or an initial strong devaluing of attention heads, due to their irrelevance for the pairs task? We considered comparing the gradients from different subtasks for these questions.
6. Characterisation: To gain a better understanding of the phenomenon, we raised the avenue of finding additional language pairs showing this interference, in particular with the hopes of finding enough to characterise and predict whether a pair of languages will or will not interfere in learning.

The group looks forward to exploring these questions in future work.

References

1 Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=De4FYqjFueZ`.

4.3 Interpretability

Michael Hahn (Universität des Saarlandes – Saarbrücken, DE), Anej Svete (ETH Zürich, CH), Joshua M. Ackerman (Dartmouth College – Hanover, US), Satwik Bhattamishra (University of Oxford, GB), Jiaoda Li (ETH Zürich, CH), Paul S. Lintilhac (Dartmouth College Hanover, US), Andy Yang (University of Notre Dame, US)

This working group focused on ways in which the theory of language models could support the development of interpretability methods of language models.

4.3.1 Discussed Problems

- Desiderata of an interpretation,
- How theoretical work can help interpretation efforts,
- Prospects for extracting logical descriptions from neural language models,
- Local vs. global interpretations,
- Formalization of what researchers usually refer to as "circuits for interpretation", and
- Compression of models for interpretability.

4.3.2 Possible Approaches

4.3.2.1 Desiderata of interpretability

A usable interpretation should satisfy multiple desiderata.

1. It should capture the right level of *abstraction* – it should allow human-readable information while staying faithful to the computation performed by the model,
2. It should be *actionable* – it should allow the user to intervene on the human-interpretable computations performed by the model and change them to induce target behavior,
3. It should enhance the *scientific understanding* of the model's computation, biases, etc.,
4. It should provide insights into the usability and limitations of the model – it should inform the user when the model's outputs can be trusted and when not,
5. It should enhance the user's trust in the model – its interpretation should provide evidence that the model is faithful to the user's intent, and
6. It should aid model's auditability and ensure that the model satisfies requirements for deployment.

4.3.2.2 Extracting the logical descriptions of neural language models

Existing literature provides exact characterizations of transformers as counter-free and partially-ordered finite-state automata. This theoretically allows us to convert any transformer into an automaton and leverage tools available for the interpretation of those on transformers. For instance, expressing a multiple transformers as finite-state automata would allow one to ask about their equivalence, differences, and emptiness. It would also allow one to test whether the models are provably performing computations required by looking for patterns in the automata.

Nevertheless, issues arise when considering the size of the resulting representations – the multi-step conversions required by known constructions result in super-exponential growth of the automata with respect to parameters such as the precision and number of layers. This is in conflict with the desideratum of a manageable and human-readable representation and raises questions about the feasibility of actual implementations.

4.3.2.3 Formalization of a circuit for interpretability

The group discussed the notion of *circuits* prevalent in recent interpretability research on language models. Such circuits provide *local* interpretations specific to individual input prompts x. Faithfulness to the original language model is typically evaluated on prompts formally highly similar to the target prompt x, often following some template. The group then discussed how one could formalize such local interpretations in the case of transformer language models, and desiderata for overall meaningfulness for such local interpretations.

4.3.2.4 Model compression for interpretability

Motivated by the desideratum of interpreting models with smaller but faithful representations, we discussed the problem of model compression. Intuitively, by compressing a model into a faithful smaller and thus easier-to-interpret representation, one could make inferences about the original model.

We discussed multiple avenues for compression:

- Quantization of model weights,
- Removing neurons in the feedforward networks, i.e., reducing the model width,
- Removing heads, which is analogous to reducing the model width,
- Low rank representations of the matrices, and
- Reducing the number of layers.

The last point seems challenging as it requires a large intervention on the model. Nevertheless, some results on the depth requirements to express certain languages do suggest that sometimes the depth could be reduced.

To address the other compression methods, the group discussed in what way the compressed model should be equivalent of similar to the original one. A good starting point could be requiring the compressed model to represent an equivalent *recognizer* – the identical string-to-membership function. Concretely, in the standard transformer setting in which real-valued outputs are mapped to classification decisions, one can take advantage of the classification gap to change the model slightly without influencing the string classifications.

4.4 Uniformity

William Merrill (New York University, US), Laura Strieker (Leibniz Universität Hannover, DE), Satwik Bhattamishra (University of Oxford, GB), Michaël Cadilhac (DePaul University – Chicago, US), David Chiang (University of Notre Dame, US), Ashish Sabharwal (Allen Institute for AI – Seattle, US), Clayton Sanford (Google – New York, US), Howard Straubing (Boston College, US)

Much work on transformer expressivity has followed in the formal language theory tradition of defining a problem as expressible if there exists a fixed transformer (with respect to the input length n) that can recognize a language. However, some work allows the hyperparameters of the network (e.g. depth, width) or even the parameter values themselves to depend on n. To understand the differences between these regimes, this group sought to carefully define a notion of *uniformity* for transformers whose parameters can depend on the sequence length. We used this definition to instantiate several natural classes of uniform transformer families

generalizing the standard fully uniform transformers that do not depend on n in any way. We also considered potential separations between these classes (e.g., between fully and L-uniform classes).

4.4.1 Discussed Problems

We propose carefully defining and analyzing the uniformity of families of transformers, motivated by the following questions:

- How should we think about uniformity for transformers? What differences are there to circuit classes.
- Theories of expressivity with a bounded sequence length
- Better accounting for the role of depth and width in transformer expressivity
- Separations between transformers that are fully uniform and transformers where some parameters or hyperparameters can depend uniformly n

4.4.2 Possible Approaches

We can think about two ways that a transformer can potentially change as the sequence length n increases:

1. The parameters θ themselves can change. The shape could remain the same but values can adapt. Potentially, the number of parameters could even expand with n.
2. Given a fixed vector of parameters, aspects of inference (how we apply the model to data) could change. For example, we could increase the precision as $O(\log n)$ or pad the input with $O(\text{poly}(n))$ blank tokens. Crucially, in either case, we use the same parameters for all input lengths.

To account for both of these forms of change with n, we propose decomposing a transformer into its architectural schema and parameter vector. The architectural schema can specify that certain properties of inference evolve with n (e.g., precision, chain of thought, or padding). Uniformity of the transformer weights can then be defined similarly to circuits.

Concretely, we can define an architectural schema as follows:

▶ **Definition 1.** A *transformer architecture* $\mathcal{T}$ is a Tuple $(p, t, d, w, e, m, n, a, c)$ where

- p denotes the precision,
- t temperature,
- d depth,
- w width,
- e positional encoding,
- m masking,
- n layer norm,
- $a \in \{uha, aha, sma\}$ attention,
- c number of chain of thought steps (or padding tokens).

We call these the *hyperparameters* of a transformer.

The following example illustrates this definition. $\mathcal{T} = (O(1), ..., O(1), sma, 0)$ are fixed-precision, temperature, and width transformers with softmax attention and no chain of thought. None of the hyperparameters depend on n.

A transformer architecture $\mathcal{T}$ can be parameterized by a weight vector θ to obtain a language recognizer $\mathcal{T}_\theta : \Sigma^* \to \{0, 1\}$. We can then define uniform transformer classes in terms of these language recognizers and uniform families of parameters $\{\theta_N\}_{N=0}^{\infty}$:

▶ **Definition 2.** Let $\mathcal{T}$ be a transformer architecture. Then U-uniform-$\mathcal{T}$ is the set of languages L such that there exists a family of parameter vectors $\{\theta_N\}_{N=0}^{\infty}$ such that $\mathcal{T}_{\theta_N}$ recognizes $L^{\leq N} = \{w \mid |w| \leq N \wedge w \in L\}$ for all N and the function $N \mapsto \theta_N$ can be computed in the class U.

Note it is enforced that, as we change the parameters of a transformer family, its behaviour must remain the same as smaller inputs. This restriction is not possible with circuits, which take a fixed-size input. In contrast, transformers can read a variable-length string as input.

Let $\mathcal{C}$ denote the set of constant functions $\mathbb{N} \to \Sigma^*$ and recall $\mathcal{T}_{fu}$. Then $\mathcal{C}$-uniform-$\mathcal{T}_{fu}$ is the expressive power of "fully uniform" transformers where the parameters cannot change at all with n. In contrast, $\mathcal{L}$-uniform-$\mathcal{T}_{fu}$ is the class where the number of parameters stays the same, but the values of θ can change uniformly in a way computable in log space. Finally, let $\mathcal{T}_w = (O(1), ..., O(1), O(\log n), sma, 0)$ denote transformers with logarithmic width. Then $\mathcal{L}$-uniform-$\mathcal{T}_w$ is the set of languages recognizable by transformers with logarithmic width, where the parameters can change with n in a way computable in log space. Thus, our single definition can be instantiated to make all of these notions of uniform transformer families precise.

4.4.3 Tentative Separations

A general theme is that, if a fully uniform transformer model $\mathcal{C}$-uniform $\mathcal{T}$ has $O(\log n)$ communication complexity, we expect that L-uniform $\mathcal{T}$ will not. This can allow us to show a separation between the $\mathcal{C}$-uniform and L-uniform $\mathcal{T}$.

- One layer: communication complexity lower bound for fully uniform version; increase temperature to increase expressive power for L-uniform model
- Multilayer: [3] gives analogous communication complexity lower bound; increase temperature to gain power for L-uniform model
- C-RASP: same idea of $O(\log n)$ communication complexity, but uniformity lets us expand that
- What about C-RASP transformers with dynamic temperature? Does the additional power of L-uniformity persist?
- With padding, L-uniform and fully uniform AHAT collapse, since we get both equal to TC^0.

4.4.4 Conclusions

We have made progress towards a general and rigorous notion of uniformity for families of transformers, which we hope will aid future work studying the expressivity of transformer families, allowing it to be more standardized and rigorous. An immediate next step is refining our understanding of separations between various types of transformers with different levels of uniformity, which could have some relevant for understanding fundamental limits on length generalization: inexpressibility of a language under a fully uniform model implies length generalization is not possible on that language. More generally, better understanding the expressivity of uniform transformers can help us better conceptualize expressivity with a maximum context length and the role of parameters like depth or width in transformer expressivity.

4.5 Depth

Satwik Bhattamishra (University of Oxford, GB), Clayton Sanford (Google – New York, US), Michael Hahn (Universität des Saarlandes – Saarbrücken, DE), Ashish Sabharwal (Allen Institute for AI – Seattle, US), Andy Yang (University of Notre Dame, US)

Understanding the limitations of small-width multi-layer Transformers has remained challenging. For sequences of length $\leq N$, recent works [1, 2] show that one-layer Transformers of width $o(N)$ cannot express certain functions, but analogous limitations for multi-layer models are still elusive. An interesting open direction is to identify functions that can be represented by three-layer Transformers of small width (e.g., $O(\log N)$) but not by two-layer Transformers (e.g., they require $\omega(\log N)$ width). Such separations are known for 1-layer vs. 2-layer Transformers [2]. Notably, [3] proved unconditional lower bounds and provided separations for multi-layer Transformers with *causal masking*; comparable lower bounds are not known for Transformers without causal masking.

4.5.1 Discussed Problems

- Our primary focus was on approaches to derive unconditional lower bounds for multi-layer Transformers.
- We explored potential strategies beyond the current communication complexity–based reductions.
- We compared and contrasted the logic-based and communication complexity-based results on transformers.
- We sought to identify concrete functions or tasks that appear easier for three-layer Transformers but not for two-layer Transformers as a starting point for such analysis.
- We discussed barriers that make proving lower bounds for two-layer Transformers harder than for one-layer models.

4.5.2 Open Problems and Possible Approaches

KW games. One approach we explored is adapting Karchmer–Wigderson (KW) games [4], a communication-complexity technique that has been fruitful for deriving depth lower bounds for monotone Boolean functions. Current approaches partition the input between two parties and show that the network output can be computed with a few bits of communication (depending on the network width), which then yields width lower bounds. In contrast, KW games organize the protocol so that communication proceeds *across depth*, and CC-reductions lead to depth lower bounds. When attempting to apply this idea to Transformers, we encountered barriers: for circuits, the approach relies (to some degree) on monotonicity (which does not hold for neural nets) and on bounded fan-in. The latter holds for hard-attention Transformers but not more general settings. Developing a nontrivial adaptation that leverages the KW-games paradigm [4] for Transformers remains an interesting open direction.

2 vs. 3 layer separation. As a starting point, we aimed to identify tasks that are intuitively difficult for two-layer Transformers but not for three layers. The goal is to isolate a concrete problem that can be analyzed to formalize these intuitions.

We develop the *compositional indexing* problem (Def. 1), based on the following idea. A Transformer layer consists of two blocks: the attention block, which gathers or processes information across the sequence, and the feedforward block, which processes information at a particular position. Prior work shows that certain tasks (e.g., Inner Product mod 2, IP2, and Disjointness) are hard for one-layer Transformers but not for two layers. We extend this by asking the model to compute IP2 not on the raw input $x \in \{0,1\}^N$ but on a subsequence specified by a list of indices $i_1, \ldots, i_N \in [N]$. The model receives a bit string $x_1, \ldots, x_{2N} \in \{0,1\}$ followed by indices $i_1, \ldots, i_N \in [2N]$, and must compute IP2 on the selected bits. The formal definition is as follows.

▶ **Definition 1** (Compositional indexing problem)**.** Fix a positive integer $N \in \mathbb{N}$ and write $[m] = \{1, 2, \ldots, m\}$.

Base function. Let $f\colon \{0,1\}^N \to \{0,1\}$ be any Boolean function with two-party communication complexity $\Omega(N)$. For concreteness, one may take f to be the IP mod 2 function or Equality. The IP mod 2 function $\mathrm{IP2} : \{0,1\}^{N/2} \times \{0,1\}^{N/2} \to \{0,1\}$ is

$$\mathrm{IP2}(\mathbf{x}, \mathbf{y}) \;=\; \langle \mathbf{x}, \mathbf{y} \rangle \bmod 2 \;=\; \big(h(x_1, y_1) + h(x_2, y_2) + \cdots + h(x_{N/2}, y_{N/2})\big) \bmod 2,$$

where $h(x_i, y_i) = x_i y_i$.

Selection operator. Define

$$\mathrm{Sel}_N\colon \{0,1\}^{2N} \times [2N]^N \;\longrightarrow\; \{0,1\}^N, \qquad \big(\mathrm{Sel}_N(x, \mathbf{i})\big)_j \;=\; x_{i_j} \quad \text{for } j = 1, \ldots, N,$$

where $x = (x_1, \ldots, x_{2N}) \in \{0,1\}^{2N}$ and $\mathbf{i} = (i_1, \ldots, i_N) \in [2N]^N$. Thus, Sel_N outputs the length-N subsequence of x at coordinates $i_1, \ldots, i_N$.

Composed function. The final mapping is the composition

$$g\colon \{0,1\}^{2N} \times [2N]^N \;\longrightarrow\; \{0,1\}, \qquad g(x, \mathbf{i}) \;=\; f\big(\mathrm{Sel}_N(x, \mathbf{i})\big).$$

In other words, g receives a length-$2N$ binary string together with N indices, extracts the indexed bits via Sel_N, and feeds the resulting length-N string into f. Let CoIP2 denote the compositional IP mod 2 function,

$$\mathrm{CoIP2}(x, \mathbf{i}) \;=\; \mathrm{IP2}\big(\mathrm{Sel}_N(x, \mathbf{i})\big).$$

For the compositional indexing problem, it is straightforward to adapt techniques from [2] to show that three-layer Transformers with width $O(\log N)$ can compute CoIP2. Similarly, two-layer Transformers with width $O(N)$ can compute it. The key open question is whether two-layer Transformers with width $o(N)$—or even $O(\log N)$—can compute CoIP2. Proving that two-layer Transformers require width $\omega(\log N)$ would yield a separation between two and three layers.

In this problem, the model receives the pair $(x, \mathbf{i})$. Upon seeing the indices $\mathbf{i} = (i_1, \ldots, i_N)$, if the first layer only retrieves the input bits $x_{i_1}, \ldots, x_{i_N}$, then we have that the second layer cannot compute functions like IP2 based on prior works. However, the first layer need not be restricted to this behavior – this is the main challenge. A useful observation is that to compute functions like IP2, the attention block must first gather input bits at paired indices $(i_1, i_{\frac{N}{2}+1}), \ldots, (i_{\frac{N}{2}}, i_N)$. When the Transformer receives the input index i_k, while it can retrieve the bit x_{i_k}, one can show (by adapting arguments from [5]) that it cannot simultaneously retrieve bits x_{i_j} for any other index i_j with $j \neq k$. Thus one can prove that the first layer cannot perform the one-hop retrieval or index lookup, which could be potentially useful for proving lower bounds for the second layer. For the reasons described above, we hypothesize that CoIP2 could be hard for log-width two-layer Transformers but the problem remains open.

References

1 Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36:36677–36707, 2023.
2 Satwik Bhattamishra, Michael Hahn, Phil Blunsom, and Varun Kanade. Separations in the representational capabilities of transformers and recurrent architectures. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
3 Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer. *arXiv preprint arXiv:2412.02975*, 2024.
4 Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 539–550, 1988.
5 Clayton Sanford, Daniel Hsu, and Matus Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv preprint arXiv:2408.14332*, 2024.

4.6 Probability

David Chiang (University of Notre Dame, US), Ryan Cotterell (ETH Zürich, CH), Jiaoda Li (ETH Zürich, CH), Anthony W. Lin (RPTU Kaiserslautern-Landau, DE), Jon Rawski (San José State University, US), Noah A. Smith (University of Washington – Seattle, US), Andy Yang (University of Notre Dame, US), Anej Svete (ETH Zürich, CH)

4.6.1 Discussed Problems

Most theoretical papers on transformers treat them as language recognizers, where the input is a string and the output is a truth value (true for accept, false for reject). But transformers are usually used as *language models*, which define a probability distribution $P(w_t \mid w_1 \cdots w_{t-1})$ where $w_t \in \Sigma \cup \{\text{EOS}\}$. Instead of asking what languages are recognized by transformers, should we be asking what probability distributions are modeled by transformers?

We dug into this question, focusing on UHATs, which, as language recognizers, were shown by [1] to be equivalent to B-RASP and LTL. (Other transformer variants were also discussed; C-RASP in particular led to a spin-off topic in Section 4.9.)

4.6.2 Possible Approaches

We began by defining a generalization of B-RASP which maintains its restrictions while adding probabilities (or, more generally, weights), which we call PB-RASP. We examined various ways of relating them to probabilistic automata, and therefore to distributions over strings.

4.6.3 Conclusions

The definition of PB-RASP that we arrived at has much in common with the weighted first-order logic (FO) of [2]. Both have three layers: first, an unweighted logic (B-RASP or FO); second, the ability to choose weights conditioned on unweighted formulas, defining "step" functions; third, multiplication of weights across all positions. One advantage of this formulation is that it cleanly separates two differences between language recognizers and

language models: on the one hand, language models are *weighted* while language recognizers are unweighted, and on the other hand, language models are *autoregressive* while language recognizers are not.

As language recognizers, UHATs and B-RASP are equivalent to counter-free automata, but as language models, counter-free automata split into (at least) two levels of expressivity, counter-free DFAs and counter-free NFAs. A key finding of this working group is that PB-RASP, and therefore language models based on UHATs, are equivalent to counter-free DFAs, not NFAs.

Follow-up discussions of this question have led to further developments, which we hope to write about elsewhere in the very near future.

References

1 Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In *Advances in Neural Information Processing Systems*, volume 37, pages 10202–10235, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/13d7f172259b11b230cc5da8768abc5f-Paper-Conference.pdf`.

2 Manfred Droste and Paul Gastin. Aperiodic weighted automata and weighted first-order logic. In *Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science*, 2019. doi: `10.4230/LIPICS.MFCS.2019.76`.

4.7 Recurrence

Gail Weiss (EPFL – Lausanne, CH), Brian DuSell (ETH Zürich, CH), Robert Frank (Yale University, US), Martin Grohe (RWTH Aachen, DE), Laura Strieker (Leibniz Universität Hannover, DE)

This working group discussed recurrence as it relates to the expressive and practical power of neural language models. The group met for two sessions on Tuesday, identifying some potential future experiments before dispersing.

4.7.1 Discussed Problems

Our discussion distinguished between two types of recurrence: *horizontal*, in which a model updates a state for every input token in an ordered sequence, and *vertical*, in which the model may continue to increase its computation depth in response to its own current state. Martin shared insight on the use of vertical recurrence for evaluable problems such as minimal graph colouring, while Brian shared results on the direct advantages of horizontally recurrent models over attention-based non-recurrent ones [1, e.g.]. We discussed how the advantages (and weaknesses) of recurrence could impact the processing of natural language sequences specifically. We discussed the natural language motivations for horizontal recurrence, and how we may approach these motivations while maintaining the powerful parallelisation abilities of the non-recurrent transformer. The discussion also touched on potential limitations of the causal attention mask used in autoregressive transformers.

- What problems are vertical and horizontal recurrence used to solve, and are these to be expected in natural language?

- What is the performance and efficiency of modern recurrent or semi-stateful models, and how successful are current attempts at integrating statefulness into modern models?
- What are the main weaknesses of non-recurrent models as they relate to natural language? What are the main weaknesses of recurrent models?
- What methods other than recurrence are there to improve length generalization in transformers?
- What other capabilities of language models have been sacrificed in the name of parallelisation?

A related interesting point that arose during the discussion was the impact of the causal attention mask deployed in transformer-based natural language modeling, which allows high parallelization during training at the cost of further reduced processing capacity over individual input tokens. In this setting, the embeddings for early tokens in a sequence are based on far less of the input than they could be, even when being used for predictions that may read the entire sequence.

4.7.2 Possible Approaches

Reflecting on the combined issues of lacking recurrence and limiting causal mask, we realised that the processing of a long input sequence could potentially be improved by applying horizontally recurrent computation over the sequence, with varying computation depth for different tokens and a gradually lifted causal mask, allowing the processing of earlier sub-blocks to shift from local next-token predictions to fuller-sequence embeddings. We left the careful implementation and exploration of such an architecture to future work.

Other motivations that arose for recurrence involved length generalization of networks to long inputs, and the ability to perform arbitrarily deep computations (depth generalization). For these, we found several works that may inspire future experiments, covering for example: the incorporation of horizontal (Jamba, [2]) and vertical (universal transformers, [3]) recurrence, or similar temporal bias (Transformer-XL, [4]), into transformers, works on stateful but highly parallelisable models (e.g., S4, [5]), and even interesting works on parallelizing the implementation of recurrent neural networks (RNNs) while not impacting their actual recurrence, to allow their more effective scaling [6].

4.7.3 Conclusions

The linearly increasing computation depth of recurrent models provides them clear advantages in certain tasks, but the training routines for such models are not easily parallelisable, and hence they are difficult to exploit with today's resources. Nevertheless, a pragmatic targeting of the problems we wish to improve on for natural language processing specifically may allow for more feasible solutions.

References

1 William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

2 Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, Chen Almagor, Clara Fridman, Dan Padnos, Daniel Gissin, Daniel Jannai, Dor Muhlgay, Dor Zimberg, Edden M. Gerber, Elad Dolev, Eran Krakovsky, Erez Safahi, Erez Schwartz, Gal Cohen, et al. Jamba: Hybrid Transformer-Mamba language models. In *Proceedings of ICLR*, 2025. URL https://openreview.net/forum?id=JFPaD7lpBD.

3 Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *Proceedings of ICLR*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

4 Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285/.

5 Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *Proceedings of ICLR*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

6 Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=E34AlVLN0v.

4.8 Chain of Thought

Ashish Sabharwal (Allen Institute for AI – Seattle, US), William Merrill (New York University, US), Michaël Cadilhac (DePaul University – Chicago, US), Howard Straubing (Boston College, US), Laura Strieker (Leibniz Universität Hannover, DE), Michael Hahn (Universität des Saarlandes – Saarbrücken, DE)

This working group focused on how the amount of intermediate generation (i.e., the number of Chain of Thought tokens) affects the representation power of Transformers, especially in the sublinear regime. It is known that, on inputs of size n, $\mathsf{CoT}[\mathrm{poly}(n)] = \mathsf{P}$ and $\mathsf{CoT}[n]$ can solve certain NC^1-complete problems. On the other hand, allowing only $O(\log n)$ intermediate steps doesn't seem to add more expressive power: $\mathsf{CoT}[\log n] \subseteq \mathsf{TC}^0$, which coincides with the best known upper bound on $\mathsf{CoT}[0]$, i.e., no intermediate generation at all. What interesting things can we say about $\mathsf{CoT}[f(n)]$ for sub-linear functions f?

4.8.1 Discussed Problems

- Can $\mathsf{CoT}[\sqrt{n}]$, or, more generally, $\mathsf{CoT}[n^{1/c}]$, solve problems beyond TC^0?
- Can $\mathsf{CoT}[\log^2 n]$, or, more generally, $\mathsf{CoT}[\log^k n]$, solve problems beyond TC^0?

Consider a "parallel" version of intermediate generation, denoted $\mathsf{CoT}[g(n), f(n)]$, where the model takes $f(n)$ steps of generation and in each step gets to write $g(n)$ tokens (in parallel) rather than a single token, which is related to the practical method of speculative decoding [1].

- What can $\mathsf{CoT}[\mathrm{poly}(n), \log^k n]$ solve?

4.8.2 Possible Approaches

Going from linear to $n^{1/c}$ is possible via the standard *padding construction* from complexity theory. Let Q be an NC^1-complete language that is also in $\mathsf{CoT}[n]$, i.e., can be recognized using n CoT steps. Consider the language $Q' = \{w\, 1^{|w|^c} \mid w \in Q\}$ where 1 is some symbol

not used in Q, i.e., every string in Q is appended with polynomially many 1's. Then (a) Q' is also NC^1-complete and (b) can be recognized by $\mathsf{CoT}[n^{1/c}]$.

In fact, the same applies even when L is a P-complete problem. It follows that:

- For any $c > 0$, there is a P-complete problem in $\mathsf{CoT}[n^{1/c}]$.
- If $\mathsf{NC} \neq \mathsf{P}$, then there exists a language L that, for any $c > 0$ and $k \in \mathbb{Z}_{\geq 0}$, is in $\mathsf{CoT}[n^{1/c}]$ but not in $\mathsf{CoT}[\log n]$.

This should also extend in some form to the parallel version of CoT, i.e., $\mathsf{CoT}[\mathrm{poly}(n), n^{1/c}]$ vs. $\mathsf{CoT}[\mathrm{poly}(n), \log^k n]$.

It should be possible to show:

- $\mathsf{CoT}[\mathrm{poly}(n), \log^k n] \subseteq \mathsf{TC}^k$ by viewing a length-d CoT as a circuit of depth d.
- $\mathsf{CoT}[\mathrm{poly}(n), \log^k n] \supseteq \mathsf{NC}^k \supseteq \mathsf{TC}^{k-1}$, via NC^k's equivalence to alternating log-space, $\log^k$-time Turing machines.

For poly-log CoT, we have the following:

- Adding n many $(\log n)$-bit numbers is in $\mathsf{CoT}[\log^2 n]$. The idea is to first compress this addition to adding $\log n$ $(\log n)$-bit numbers, then add the resulting $\log n$ numbers sequentially. This problem is clearly in TC^0 but it's not known whether fixed depth transformers can solve it.

4.8.3 Conclusions

We considered the expressivity landscape for transformers with sublinear CoT, identifying critical regimes and open questions, and answering some of them. In particular, we showed that $\mathsf{CoT}[\sqrt{n}]$, or more generally, $\mathsf{CoT}[n^{-c}]$, can solve P-complete problems. Below this, we know that $\mathsf{CoT}[\log n] \subseteq \mathsf{TC}^0$. The intermediate regime of $\mathsf{CoT}[\log^k n]$ remains an interesting open question, as well as more rigorously analyzing CoT steps interwoven with parallel generation steps.

References

1 Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286, 2023. URL `https://proceedings.mlr.press/v202/leviathan23a.html`.

4.9 Automata

Andy Yang (University of Notre Dame, US), Michaël Cadilhac (DePaul University – Chicago, US), Ryan Cotterell (ETH Zürich, CH), Michael Hahn (Universität des Saarlandes – Saarbrücken, DE), Anthony W. Lin (RPTU Kaiserslautern-Landau, DE)

This working group focused on developing an automata-theoretic characterization of C-RASP and associated questions. This question started as an off-shoot of the probability group, looking ahead towards creating a probabilistic C-RASP.

First, there was discussion of characterizing C-RASP using counter/Parikh automata. A potential way forward would be to restrict to the class of partially ordered automata while adding some counting abilities. Next, a connection to typed wreath products of $(\mathbb{Z}, \mathbb{Z}_+)$ was also discussed. However, no conclusive results were obtained.

An effort to characterize the regular languages in C-RASP developed out of this discussion. It is known that C-RASP contains all R-trivial languages and all bounded-Dyck languages. It is also known that C-RASP avoids $\Sigma^* bb \Sigma^*$ and all non-aperiodic languages. An exact characterization remains elusive.

5 Other Questions

Below is a list of all the topics that were proposed for working groups. Many were merged to form the working group topics. Others were not discussed but would be valuable for future discussion. Some of these topics have been paraphrased.

- Definitions
  - How can we get some intuition into these architectural variations? (Michael Benedikt)
  - Better transformer definitions (Andy Yang?)
- The Big Picture
  - How do we convince engineering people that theory is important? Motivating examples, etc. (Brian DuSell)
  - Setting up for a good week: surveying what's there and seeing what's missing in expressivity (Laura Strieker)
  - What theory would be most important/relevant/useful to practical implementations of NLMs? (George Cybenko)
  - Give me something I can use to make strong models, cheaper, either from scratch, or after training. (Noah A. Smith)
  - What makes a successful "Theory of Neural Networks"? Is our work approaching that? (Andy Yang)
  - Identify (and start to fill) *gaps* between theoretical models we analyze and actual models practitioners deploy. (Ashish Sabharwal)
  - What transformer questions will take 100 years to solve? (Andy Yang)
- Learnability
  - What are the inductive biases of neural architectures (transformers, RNNs, etc.) when we ignore length generalization? (E.g., recognition up to length N.) How do we measure this? E.g., sample complexity. (Brian DuSell)
  - Learning vs. generalizing vs. expressing (Michael Hahn)
  - What is a canonical form of a transformer class to get a learning result? For example, a canonical DFA, C-RASP, etc. (Jon Rawski)
  - "Learnable Languages" but more rigorous. (Jon Rawski)
  - Interference: explaining difficulty learning two tasks, simultaneously and understanding which task pairs might interfere
  - Learning linear combinations of sparse automata. Expressibility and learnability. (Paul S. Lintilhac)
  - Can we systematically map natural language tasks to formal language complexity properties (degree, sensitivity, circuit depth) to predict generalization? (Paul S. Lintilhac)
- Interpretability
  - How can theory help interpretability? (Michael Hahn)
  - Formalizations, theory, practice, connection to formal verification, complexity theory, automata learning (Anthony W. Lin)
  - Can we translate Transformers to logic and use this for interpretability and/or formal verification? (William Merrill)

- Verification of transformers, e.g., how hard is it to test whether a given UHAT is equivalent to a given DFA? (David Chiang)
- Can we prove/make more rigorous these informal statements about limitations with respect to compositions of functions with formal languages? (Paul S. Lintilhac)
- Interpretability. How can we use theoretical results, constructions, and derivations for (coarsely) interpreting trained models? (Anej Svete)
- What's an "interpretation" I would like or find useful or insightful? (Gail Weiss)

- Continuity and probability
  - Transformers on time series? (Anthony W. Lin)
  - Rational or real numbers. Are SSMs with arbitrary precisions still in uniform TC^0? What about other networks? (David Chiang)
  - Language models as weighted/probabilistic logics, automata, etc. (Jon Rawski)
- Recurrence
  - Can we parallelise training for computation depth? (Gail Weiss)
  - Recurrence bias transformers still parallel? (Gail Weiss)
- Uniformity
  - Why are some results relying on nonuniform complexity classes? What are the uniform/nonuniform variants of the existing nonuniform/uniform results? (Michaël Cadilhac)
  - How different axes of increasing memory influence expressivity (Satwik Bhattamishra)
  - Constant-context-length transformer complexity: what is the right scale parameter if not context length? (Clayton Sanford)
  - Motivating, clustering, and relating definitions. e.g., what concerns motivate "full uniformity"? Philosophical? Practical? (William Merrill)
  - What languages are expressible up to a maximum length but not all lengths? (David Chiang)
  - How do we account for complexity parameters beyond n? Number of parameters, training time, number of experts, etc. Related to uniformity? (William Merrill)
- Depth
  - Depth? Unconditional separations without masking? Relationship to chain of thought? (Clayton Sanford)
  - Can we develop other techniques to derive lower bounds for transformers: multilayer or hybrid architectures? For example, static encodings, chronograms, counting linear regions (Satwik Bhattamishra)
- Other questions
  - Moment computation over attention (The "real" soft attention): Many of our constructions use hard attention or close simulations, but a lot of attention units look like partitions/clusters. Is there a theory that includes this? (Clayton Sanford?)
  - A lot of attention has been paid to transformer variants that have better time complexity. What about variants that have *worse* time complexity? (Gail Weiss)
  - Are residual connections necessary for the full expressiveness of transformers? (Pablo Barcelo)
  - Can we view decoders as strictly local transducers? (Jon Rawski)
  - Why are transformers so good at natural language but so bad at formal languages? (Brian DuSell)
  - Big model $\rightarrow$ Small model
  - Does monotonicity help? (Michaël Cadilhac)
  - Can we overcome limitations by configuring the network to the task? (Paul S. Lintilhac)

Participants

- Joshua M. Ackerman
Dartmouth College – Hanover, US
- Pablo Barcelo
PUC – Santiago de Chile, CL
- Michael Benedikt
University of Oxford, GB
- Satwik Bhattamishra
University of Oxford, GB
- Michaël Cadilhac
DePaul University – Chicago, US
- David Chiang
University of Notre Dame, US
- Ryan Cotterell
ETH Zürich, CH
- George Cybenko
Dartmouth College Hanover, US
- Brian DuSell
ETH Zürich, CH
- Robert Frank
Yale University, US
- Martin Grohe
RWTH Aachen, DE
- Michael Hahn
Universität des Saarlandes – Saarbrücken, DE
- Jiaoda Li
ETH Zürich, CH
- Anthony W. Lin
RPTU Kaiserslautern-Landau, DE
- Paul S. Lintilhac
Dartmouth College Hanover, US
- William Merrill
New York University, US
- Guillaume Rabusseau
University of Montreal, CA
- Jon Rawski
San José State University, US
- Ashish Sabharwal
Allen Institute for AI – Seattle, US
- Clayton Sanford
Google – New York, US
- Noah A. Smith
University of Washington – Seattle, US
- Howard Straubing
Boston College, US
- Laura Strieker
Leibniz Universität Hannover, DE
- Lena Strobl
University of Umeå, SE
- Anej Svete
ETH Zürich, CH
- Gail Weiss
EPFL – Lausanne, CH
- Andy Yang
University of Notre Dame, US