Expressivity of Transformers: Logic, Circuits, and Formal Languages

Day 1: Introduction

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)

29 July 2024

About us

David Chiang

Jon Rawski

Lena Strobl (TA)

Andy J Yang (TA)

Today's Goals

- **Situate ourselves** within the research field of theoretical analysis of transformers.
- **Connect** transformers to formal models such as automata, Boolean circuits, and formal logic.
- **Examine** results and why they may seem contradictory.
- **Explore** the many choices in transformer design and their implications.

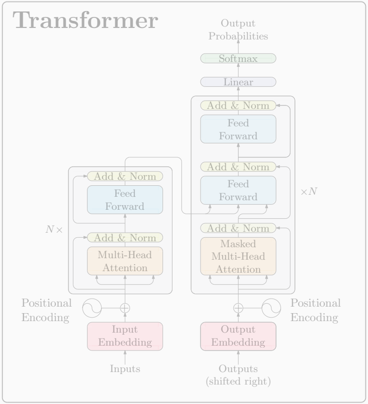
Expected level and prerequisites

- **Neural networks.** Students should be familiar with some basic neural network architectures, including feedforward networks.
- **Mathematics.** Students should be familiar with basic vector and matrix operations, including multiplication, dot (inner) products.
- **Theory of computation.** Students should be familiar with basic formal language theory, finite automata, and Turing machines. They should be familiar with first-order logic but not necessarily first-order logic for defining languages. They do not need to be familiar with circuit complexity.

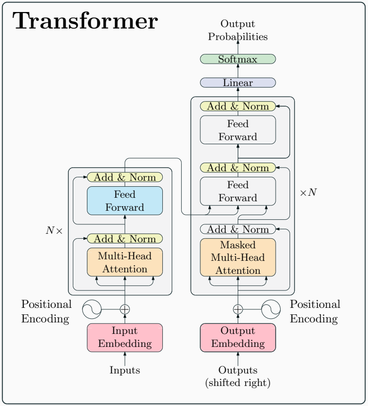
Motivation

Motivation

What **can** it do
and what **can** it **not do**?



Motivation



What **can** it do
and what **can** it **not do**?

Situating ourselves

Analyzing transformers

How would you analyze what a language model can and cannot do?

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

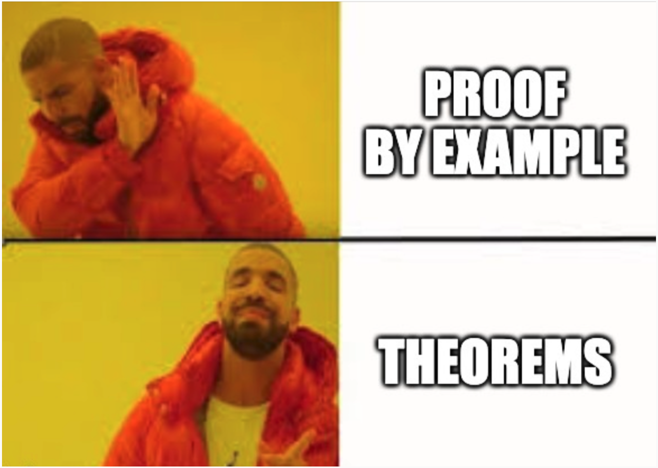
Theoretically.

- ??

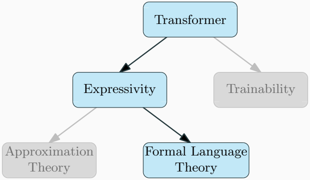
Analogy with Sorting: How do you evaluate a sorting algorithm?

- empirically: sort a bunch of lists of interest
- theoretically: how long and with how much compute does sorting a list of length n take?

Analyzing transformers



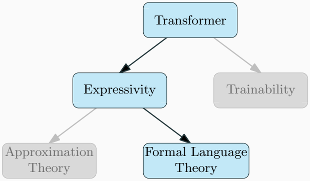
Situating ourselves



Transformer

- Transformers [Vaswani et al., 2017] are the neural network architecture underlying nearly every state-of-the-art model in natural language processing tasks, and have been extended to other fields as well.

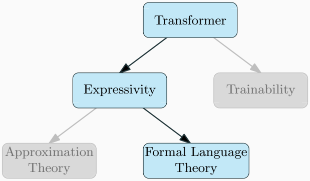
Situating ourselves



Expressivity and learnability

- Seeks boundary conditions: what class of problems can and can't be solved intrinsically by a particular class of models?
- *Learnability* concerns what problems models can or can't be trained to solve from data instances.
- Expressivity is a prerequisite for learnability.

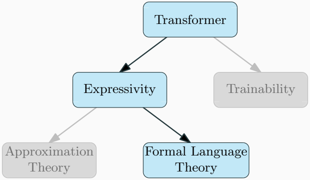
Situating ourselves



Course goal

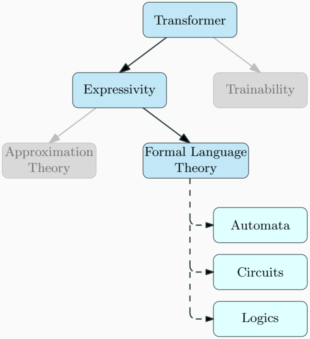
- This course is a survey of current knowledge about the expressivity of transformers from the point of view of formal languages.

Research Question



How would you go about analyzing a transformer in terms of formal langauge theory?

Research Question



How would you go about analyzing a transformer in terms of formal langauge theory?

- We want to characterize the expressivity of transformers in relation to formal models, such as automata, boolean circuits or formal logic.

Importance of understanding expressivity

Why do **you** think results of expressivity are important?

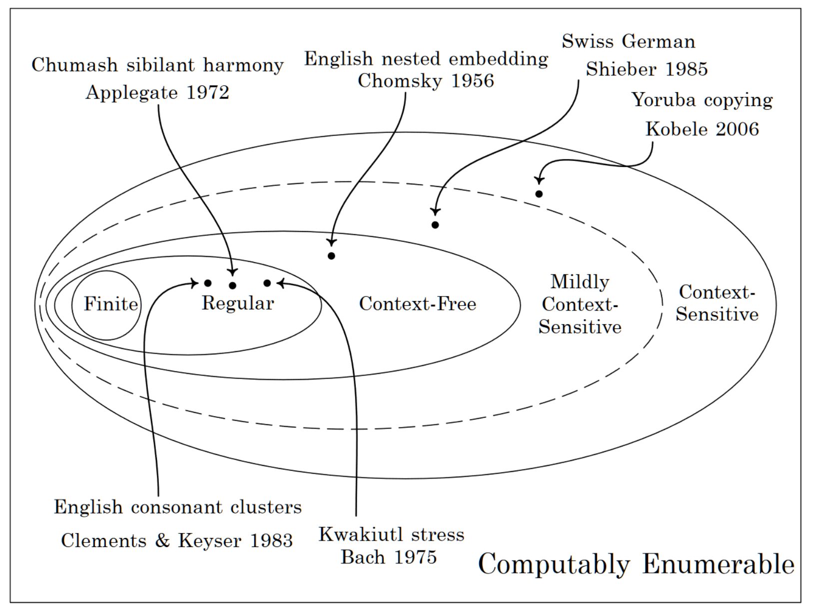
Importance of understanding expressivity

- Understanding the expressivity of transformers is crucial for both theory and practice.
- Theoretically, it helps us identify the boundaries of their capabilities, avoiding costly and tiresome experimentation.
- Practically, it informs the design of more effective models and algorithms, optimizing their performance for specific tasks in natural language processing and beyond.

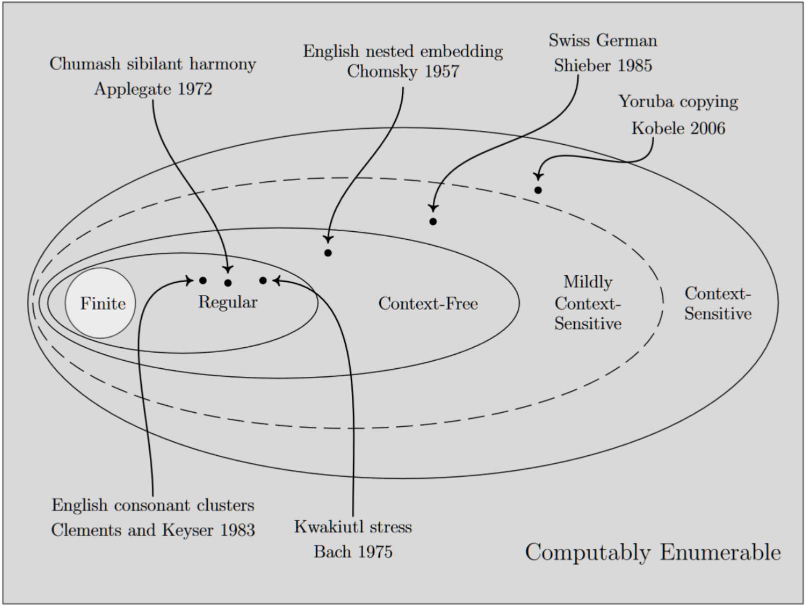
Chomsky vs Piaget (1980): expressivity & language learning

[T]hat is exactly what generative grammar has been concerned with for twenty-five years: the whole complicated array of structures beginning, let's say, with finite-state automata, various types of context-free or context-sensitive grammars, and various subdivisions of these theories of transformational grammars—these are all **theories of proliferating systems of structures designed for the problem of trying to locate this particular structure, language, within that system**. *So there can't be any controversy about the legitimacy of that attempt.*

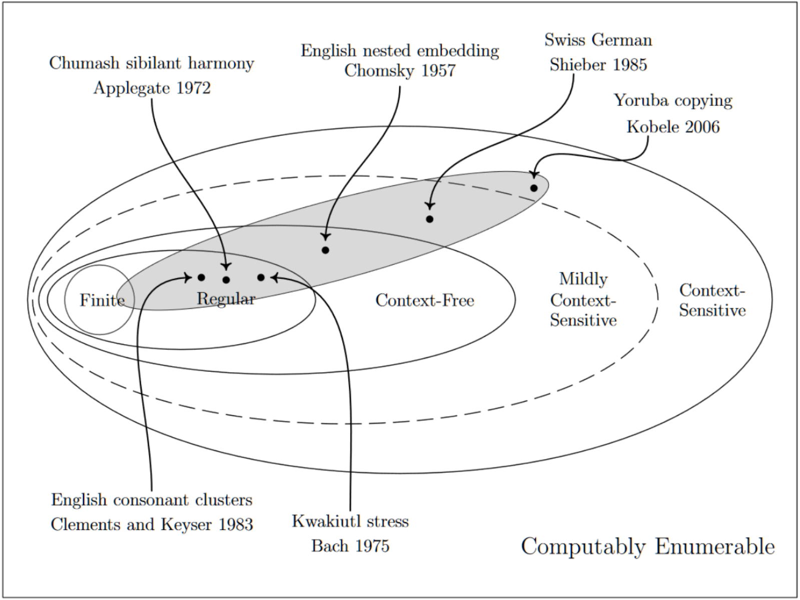
Expressivity and natural language [Rawski and Heinz, 2019]



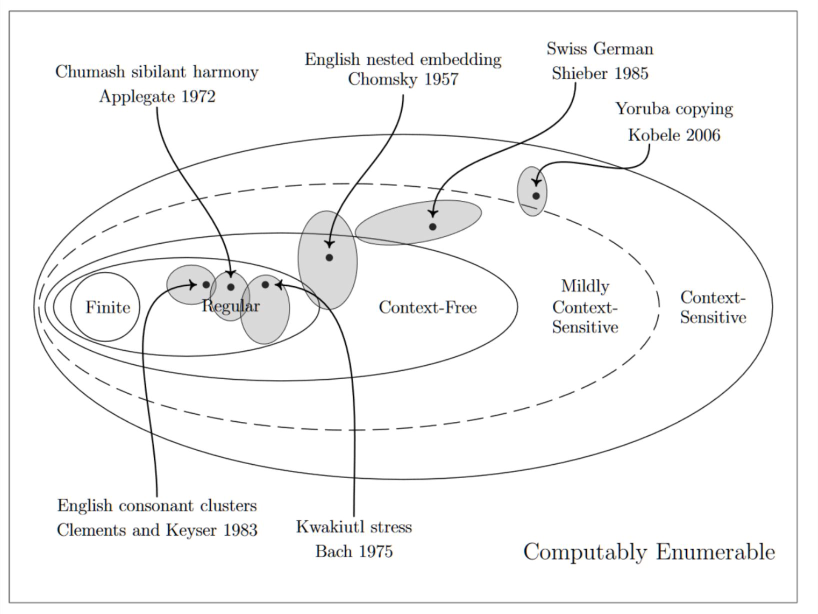
Expressivity and natural language [Rawski and Heinz, 2019]



Expressivity and natural language [Rawski and Heinz, 2019]



Expressivity and natural language [Rawski and Heinz, 2019]



Formal languages and neural Nets: Old friends

McCulloch and Pitts [1943] described ‘a logical model for the behaviour of nervous systems that turned out to be the model of a finite-state machine’ (from Perrin 1943)

Kleene [1951] attempted to generalize and nail down the kinds of “events” their neural nets could capture, and with which operations.

He called them “regular” events, which would later get simplified to regular expressions

Fun fact: Kleene called them ‘regular’ because he couldn’t think of a better name; M&P called them ‘prehensile events’

Formal languages and neural Nets: Old friends

McNaughton and Papert [1971] investigated a class of "counter-free" neural nets.

They showed these correspond to a restricted version of finite-state machines, called "counter-free" automata

They then showed these define the same languages as first-order logic with precedence and Linear Temporal Logic, the so-called "star-free" languages [Schützenberger, 1965]

Sine then, many studies on types of automata and types of neural network

Star-free regular language

Star-free languages over a finite alphabet Σ can be constructed using concatenation, union and complement.

They are the languages of star-free regular expressions, defined in BNF as:

$$\alpha ::= \emptyset \mid \epsilon \mid \sigma \mid \alpha_1 \cup \alpha_2 \mid \alpha_1 \alpha_2 \mid \alpha^{\mathsf{C}}$$

where $\sigma \in \Sigma$

Star-free regular language

Example

Let $\Sigma = \{a, b\}$.

- Σ^* is star-free because $\Sigma^* = \emptyset^C$.
- $(ab)^*$ is star-free because
  $(ab)^* = (b\Sigma^* \cup \Sigma^* a \cup \Sigma^* aa\Sigma^* \cup \Sigma^* bb\Sigma^*)^C$.
- $(aa)^*$ is regular but not star-free.

Counter-free behavior: abstract characterization of star-free

A stringset is counter-free iff there exists some $n > 0$ such that for all strings $u, v, w \in \Sigma^*$, where $|v| \geq 1$, and for all $i \geq 1$

$$uv^n w \in L \Leftrightarrow uv^{n+i} w \in L.$$

Example (English possessor recursion)

$$\begin{array}{ll} \text{my mother's mother resembled my mother} & \in L \\ \hline \text{my mother's } \underbrace{\text{(mother's)}}_{\geq 1} \text{ mother resembled my mother} & \in L \end{array}$$

Many Ways to Get Star-Free

Theorem [Schützenberger, 1965, McNaughton and Papert, 1971]

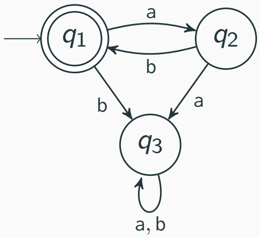
For any regular language L, the following are equivalent:

- L is star-free.
- L is counter-free.
- Its minimal DFA is counter-free.
- it is definable in first-order logic
- it is definable in linear temporal logic

Star-free regular language: Counter-free automata

Intuitively, a counter-free DFA is one that can test whether something happens, but not how many times it happens.

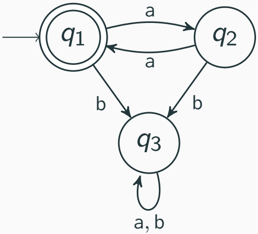
For every $q \xrightarrow{\mathbf{w}} q$, $\mathbf{w}$ cannot be $\mathbf{x}^k$ where $\mathbf{x} \in \Sigma^*$ and $k > 1$.



This DFA recognizes $(\mathrm{ab})^*$, is counter-free.

The only cycles are on ab (from q_1 to itself), a and b (from q_3 to itself), and none of these strings is of the form $\mathbf{x}^k$ for $k > 1$.

Star-free regular language: Counter-free automata



This DFA recognizes $(aa)^*$, is not counter-free.

It is not counter-free because it has a cycle on aa, which is a^2.

Star-free regular language: First-order logic

a set of finite strings that satisfy a closed formula of a logic

First-order logic (FO)

formulas are the smallest set containing

- Variables $x, y, \ldots$
- Atomic formulas $Q_a(x), x = y, x < y$
- $\phi_1 \land \phi_2, \phi_1 \lor \phi_2, \phi_1 \rightarrow \phi_2, \neg\phi_1$
- $\forall x.\phi, \exists x.\phi$

FOM

- add MAJORITY quanifiers $Mx.\phi$

$\mathrm{BIT}(x, y)$

- holds iff the y-th bit of x is 1

First-Order exercises

Exercise

Determine the formal languages of the following logical sentences.

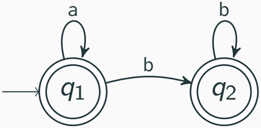
1. $(\forall x)[Q_a(x)]$
2. $(\exists x)[Q_a(x)]$
3. $(\exists x)[(Q_a(x) \wedge (\forall y)[(Q_a(y) \to x = y)])]$
4. $(\exists x)[(\exists y)[((Q_a(x) \wedge Q_b(y) \wedge x < y)]]$

Exercise

Write FO sentences for the following languages:

1. All words which begin with a ($= a\Sigma^*$)
2. All words which end with a ($= \Sigma^* a$)

Automata vs Logic: a^*b^* for $\Sigma = \{a, b\}$



The formula

$$\phi = \forall x. \forall y. Q_a(x) \wedge Q_b(y) \rightarrow x < y$$

defines the regular language a^*b^*. The formula says that every a must precede every b, which is true iff the string matches a^*b^*.

Star-free regular language: Linear temporal logic

	a	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b **since** Q_a	0	1	1	0	1	1	1

$$\mathbf{w}, 0 \models Q_a$$

Star-free regular language: Linear temporal logic

	a	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b **since** Q_a	0	1	1	0	1	1	1

$$\mathbf{w}, 3 \models Q_a \vee \neg Q_b$$

Star-free regular language: Linear temporal logic

	a	b	c	a	b	b	b
Q_a	1	0	0	1	0	0	0
Q_b	0	1	0	0	1	1	1
Q_c	0	0	1	0	0	0	0
$Q_a \vee \neg Q_b$	1	0	1	1	0	0	0
Q_b **since** Q_a	0	1	1	0	1	1	1

$$\mathbf{w}, 6 \models Q_b \textbf{ since } Q_a$$

LTL

For input string $\mathbf{w} = \mathbf{w}_1 \cdots \mathbf{w}_n$ and position $i \in [n]$, we define $\mathbf{w}, i \models \phi$ as follows:

$\mathbf{w}, i \models Q_a$	$\mathbf{w}_i = a$
$\mathbf{w}, i \models \phi_1 \vee \phi_2$	$\mathbf{w}, i \models \phi_1$ or $\mathbf{w}, i \models \phi_2$
$\mathbf{w}, i \models \neg\phi_1$	$\mathbf{w}, i \not\models \phi_1$
$\mathbf{w}, i \models \phi_1$ **since** ϕ_2	for some $j < i$, we have $\mathbf{w}, j \models \phi_2$, and for all k such that $j < k < i$, we have $\mathbf{w}, k \models \phi_1$

For an input string $\mathbf{w} \in \Sigma^+$ of length n we write $\mathbf{w} \models \phi$ if and only $\mathbf{w}, n \models \phi$.

LTL exercises

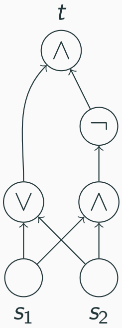
Let's redefine these same languages as before using LTL

(a) All words which begin with a (so $a\Sigma^*$)

(b) All words which end with a (so $\Sigma^* a$)

Example (XOR circuit)

Here's a circuit with input length 2. It computes the XOR function.
We draw the inputs at the bottom and the output at the top.

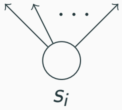


Circuit complexity

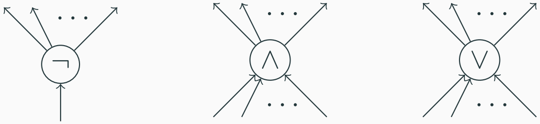
Definition (Boolean circuits)

A *(Boolean) circuit* C with input length n is a directed acyclic procedural graph with:

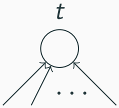
1. n *input* nodes



2. *Gate* nodes

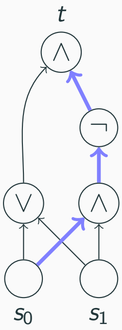


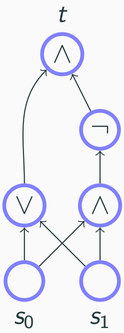
3. *Output* node t



Circuit complexity

depth(C) = 3





$|C| = 6$

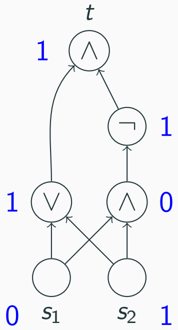
The depth of C, depth(C), is the length of the longest path from any s_i to t.

The longest path in C in is 3, therefore our depth(C) = 3.

The *size* of C, denoted $|C|$, is the number of nodes in C.

The number of nodes in C is 6, therefore $|C| = 6$.

Computation



Given: Input string $\mathbf{w} \in \{0,1\}^n$.

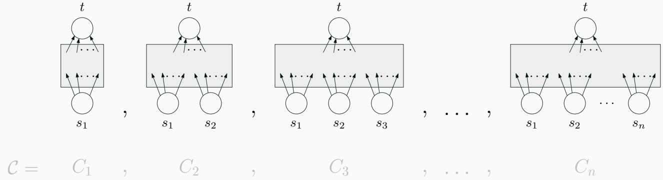
1. each input node s_i is assigned the value w_i
2. each gate node labeled f computes its value by applying f to the values of its in-neighbors.

We can think of the circuit as computing a Boolean function $C\colon \{0,1\}^n \to \{0,1\}$, mapping each input string to the value of t.

Circuit complexity

Definition (Boolean circuit families)

A *circuit family* is a sequence $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$ such that for each n, C_n is a circuit with input length n.



We treat $\mathcal{C}$ as a function on $\{0,1\}^*$ as follows. For every $\mathbf{w} \in \{0,1\}^*$ with length n, $\mathcal{C}(\mathbf{w}) = C_n(\mathbf{w})$. Then the language defined by $\mathcal{C}$ is

$$L(\mathcal{C}) = \{\mathbf{w} \in \{0,1\}^* \mid \mathcal{C}(\mathbf{w}) = 1\}.$$

Circuit complexity

The *depth* and *size* of $\mathcal{C}$ are the functions $n \mapsto \text{depth}(C_n)$ and $n \mapsto |C_n|$.

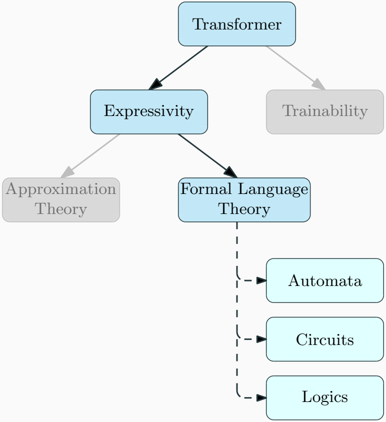
Circuit complexity classes Since transformers have constant depth, circuit classes with constant depth are of particular interest.

- AC^0 is the class of languages that can be recognized by families of circuits with unbounded fan-in, $O(\text{poly}(n))$ size, and $O(1)$ depth.
- TC^0 is like AC^0, but also allows MAJORITY gates, which have unbounded fan-in and output 1 iff at least half of their inputs are 1.
- NC^1 is the class of languages that can be recognized by families of circuits with fan-in at most 2, $O(\text{poly}(n))$ size, and $O((\log n))$ depth.

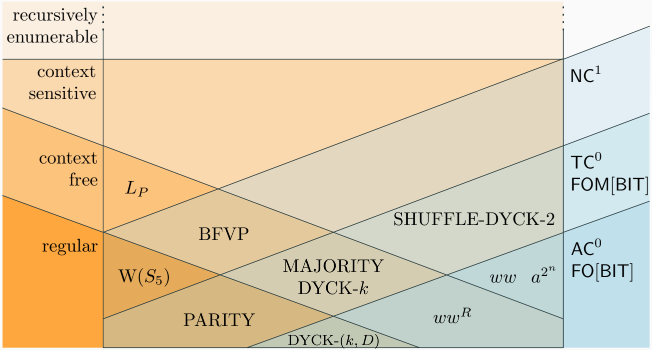
Let's take a 10 minute break!

Seemingly contradictory results

Recap

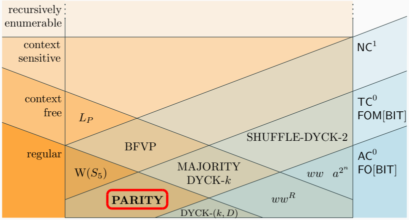

Transformer
Expressivity
Trainability
Approximation Theory
Formal Language Theory
Automata
Circuits
Logics

The Chomsky hierarchy, language, and language classes



What **can** transformers do? What can they **not** do?

The Chomsky hierarchy, language, and language classes



$$\text{PARITY} = \{x \in \{0,1\}^* \mid x \text{ has odd number of 1s}\}$$

$$\begin{aligned} 00110 &\notin \text{PARITY} \\ 10 &\in \text{PARITY} \\ 0101111 &\in \text{PARITY} \end{aligned}$$

Current Results

What has been shown so far

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

Current Results

Seemingly contradictory

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

How did this happen? Let's investigate.

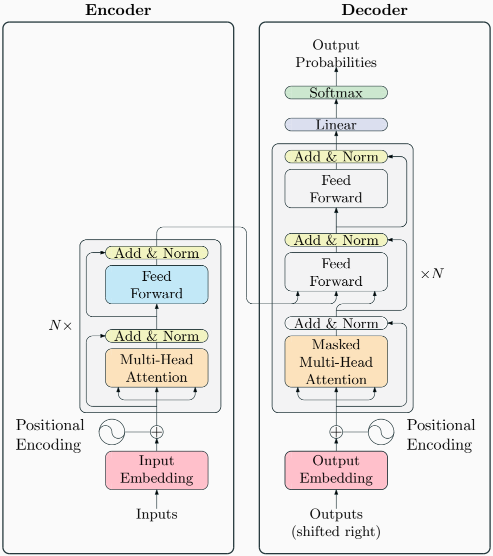
Current Results

The big picture

Lower bound	Source	PE	Attention	Notes
∋ MAJORITY	Pérez et al. 2019	none	average-hard	
∋ SHUFFLE-DYCK-k	Bhattamishra et al. 2020a	none	softmax, future mask	
⊇ SSCMs	Bhattamishra et al. 2020a	none	softmax, future mask	
∋ DYCK-k	Yao et al. 2021	i/n, i/n^3, n	softmax & leftmost-hard	
⊇ P	Pérez et al. 2021	i, $1/i$, $1/i^2$	average-hard	poly(n) steps
∋ PARITY	Chiang and Cholak 2022	i/n, $(-1)^i$	softmax	
⊇ FOC[MOD; +]	Chiang et al. 2023	sinusoidal	softmax	
⊇ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
⊇ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	
Upper bound	**Source**	**Precision**	**Attention**	**Notes**
∌ PARITY, DYCK-1	Hahn 2020	$\mathbb{R}$	leftmost-hard	
∌ PARITY, DYCK-2	Hahn 2020	$\mathbb{R}$	softmax, future mask	$\varepsilon_N > 0$, vanishing KL
⊆ AC^0	Hao et al. 2022	$\mathbb{Q}$	leftmost-hard	
⊆ TC^0	Merrill et al. 2022	$\mathbb{F}$	average-hard	
⊆ FOC[MOD; +]	Chiang et al. 2023	$O(1)$	softmax	
⊆ L-uniform TC^0	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
⊆ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
⊆ 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

Transformer

Transformer



Decisions to make: Input layer

Strings are mapped to sequences of vectors by $emb \colon \Sigma^* \xrightarrow{\text{lp}} (\mathbb{R}^d)^*$

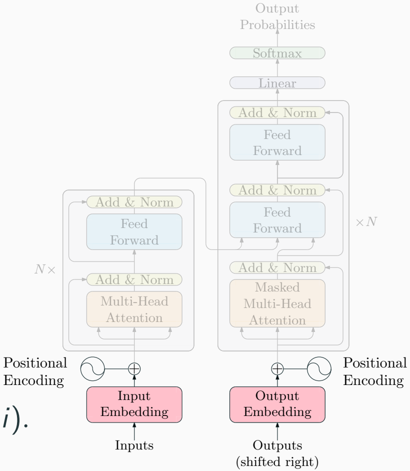
$$\text{WE} \colon \Sigma \to \mathbb{R}^d$$

and a *position(al) embedding*

$$\text{PE}_n \colon [n] \to \mathbb{R}^d$$

for $n \in \mathbb{N}_{>0}$:

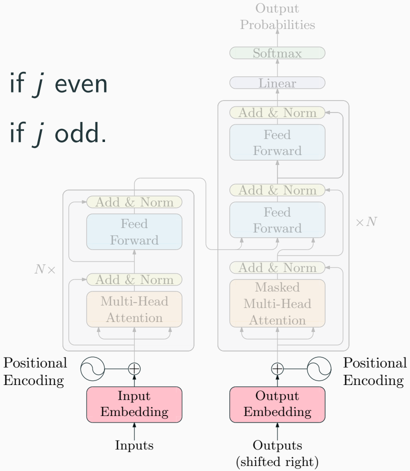
$$emb(w_0 \cdots w_{n-1})[i] = \text{WE}(w_i) + \text{PE}_n(i).$$



Decisions to make: Input layer

[Vaswani et al., 2017] introduced the following PE:

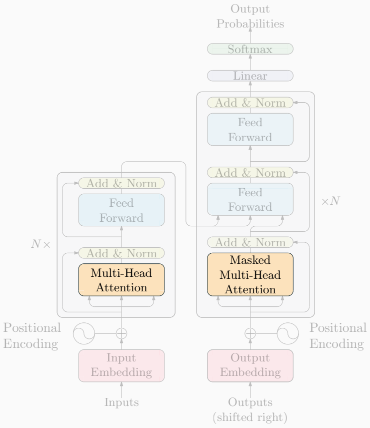
$$\mathrm{PE}_n(i)[j] = \begin{cases} \sin(10000^{-j/d} \cdot i) & \text{if } j \text{ even} \\ \cos(10000^{-(j-1)/d} \cdot i) & \text{if } j \text{ odd.} \end{cases}$$



Theoretical papers have explored other position embeddings:

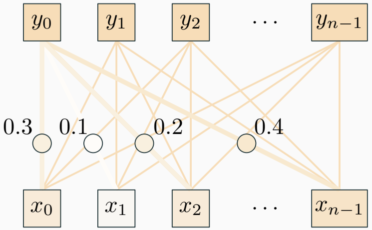
- i itself [Pérez et al., 2021]
- i/n [Yao et al., 2021, Chiang and Cholak, 2022]
- $1/i$ or $1/i^2$ [Pérez et al., 2021]

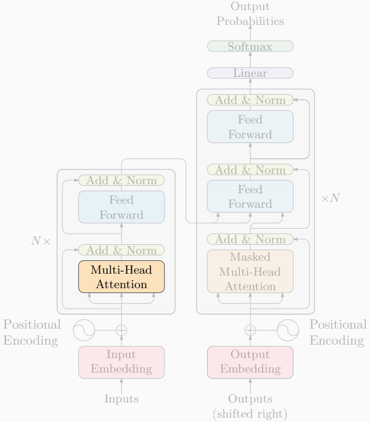
Decisions to make: Attention mechanism



Decisions to make: Attention mechanism

Softmax attention

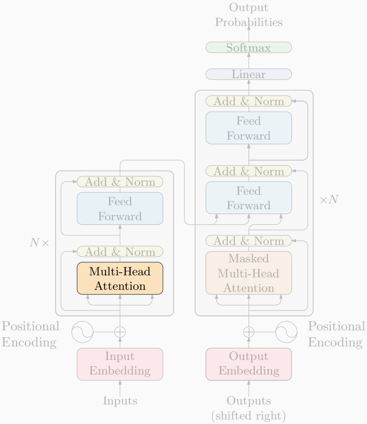




Decisions to make: Attention mechanism

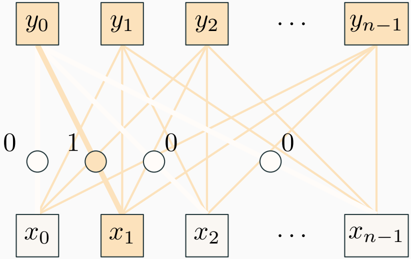
Simplified attention

Some theoretical analyses simplify attention by replacing the softmax with variants that focus attention only on the position(s) with the maximum value.

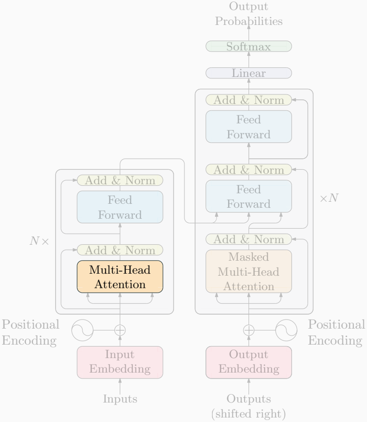


Decisions to make: Attention mechanism

Unique-hard attention

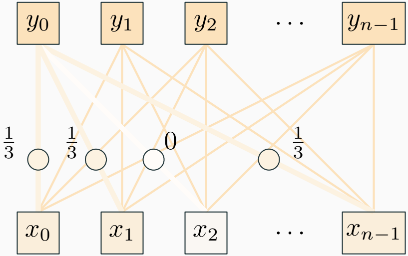


Leftmost maximal element is used.

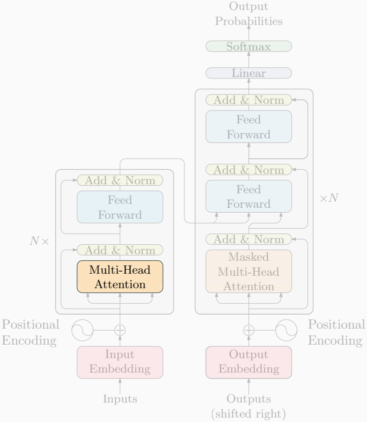


Decisions to make: Attention mechanism

Average-hard attention

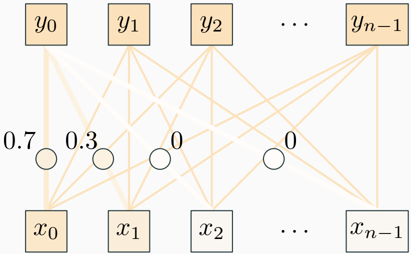


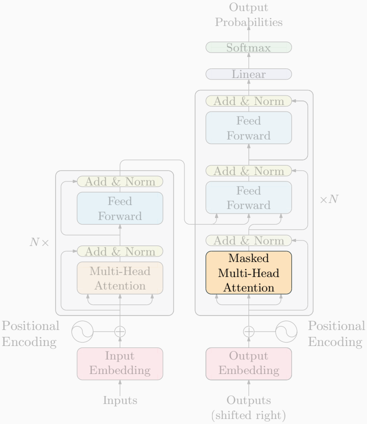
Maximal elements share weight equally.



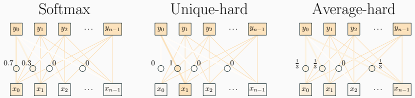
Decisions to make: Attention mechanism

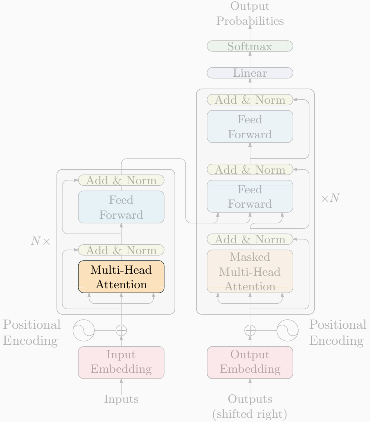
Masked attention



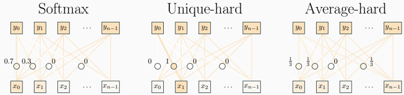


Decisions to make: Attention patterns





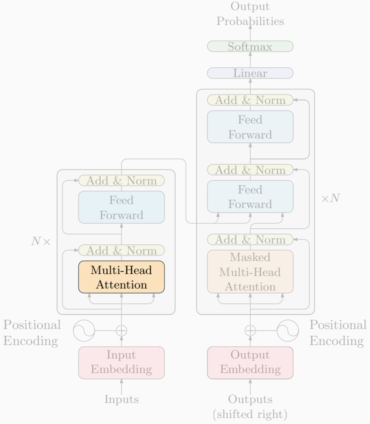
Decisions to make: Precision



Produces	$\mathbb{R}$	$\mathbb{Q}$	$\mathbb{Q}$

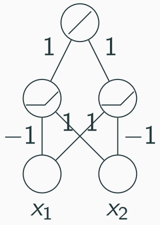
Precision

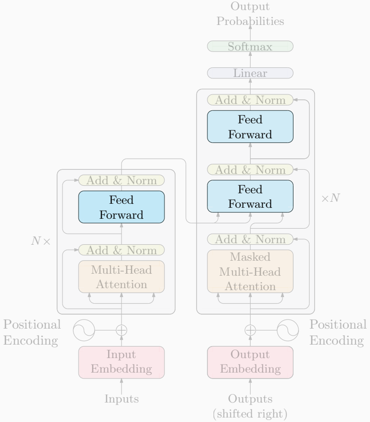
- infinite
- $O(\log n)$
- $O(1)$



Decisions to make: Feed-forward networks

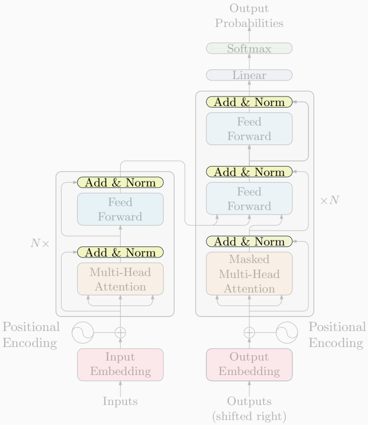
$\text{XOR}(x_1, x_2)$





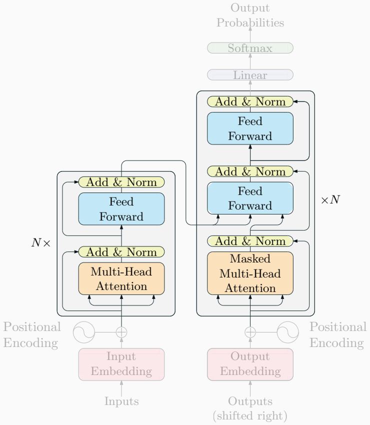
Layer normalization and hidden layers

Layer normalization



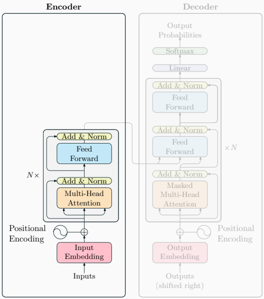
include or omit

Hidden layers

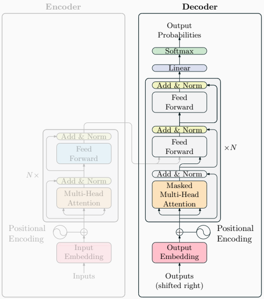


pre-norm or post-norm

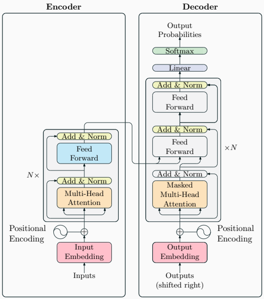
Decisions to make: Architecture



Encoder-only

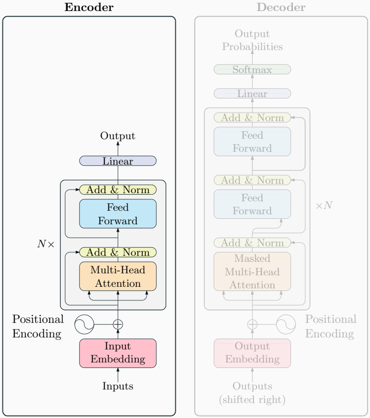


Decoder-only



Encoder-Decoder

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



Decisions to make: Summary

Residuals

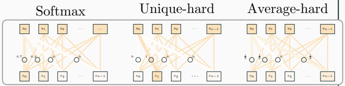
- pre-norm
- post-norm

Layer norm

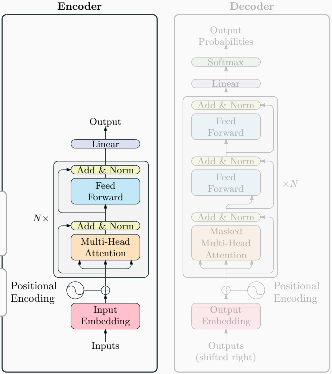
- include
- omit

Precision

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

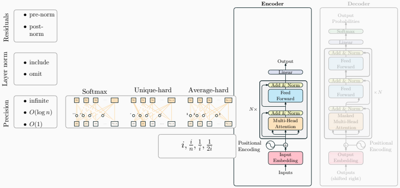


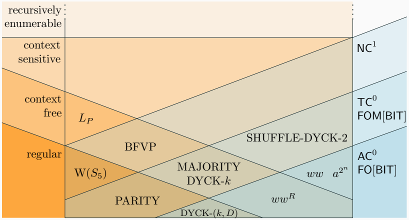
$$i, \frac{i}{n}, \frac{1}{i}, \frac{1}{2i}$$



Summary and course overview

Summary and Course Overview





Lower bound	Source	PE	Attention	Notes
∋ MAJORITY	Pérez et al. 2019	none	average-hard	
∋ SHUFFLE-DYCK-k	Bhattamishra et al. 2020a	none	softmax, future mask	
⊇ SSCMs	Bhattamishra et al. 2020a	none	softmax, future mask	
∋ DYCK-k	Yao et al. 2021	i/n, i/n^3, n	softmax & leftmost-hard	
⊇ P	Pérez et al. 2021	i, $1/i$, $1/i^2$	average-hard	poly(n) steps
∋ PARITY	Chiang and Cholak 2022	i/n, $(-1)^i$	softmax	
⊇ FOC[MOD; +]	Chiang et al. 2023	sinusoidal	softmax	
⊇ FO[Mon]	Barceló et al. 2024	arbitrary	leftmost-hard	
⊇ LTL+C[Mon]	Barceló et al. 2024	arbitrary	average-hard	
Upper bound	**Source**	**Precision**	**Attention**	**Notes**
∌ PARITY, DYCK-1	Hahn 2020	ℝ	leftmost-hard	
∌ PARITY, DYCK-2	Hahn 2020	ℝ	softmax, future mask	$\varepsilon_N > 0$, vanishing KL
⊆ AC^0	Hao et al. 2022	ℚ	leftmost-hard	
⊆ TC^0	Merrill et al. 2022	𝔽	average-hard	
⊆ FOC[MOD; +]	Chiang et al. 2023	$O(1)$	softmax	
⊆ L-uniform TC^0	Merrill & Sabharwal 2023a	$O(\log n)$	softmax	
⊆ FOM[BIT]	Merrill & Sabharwal 2023b	$O(\log n)$	softmax	
⊆ L-uniform TC^0	Strobl 2023	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

Automata

Circuits

Logics

We will in depth current results about the expressivity of transformers from the point of view of formal languages.

Course overview

Day 1

. . . things that were. . .

Day 2-4

. . . things that are. . .

Day 5

. . . and some things...
that have not yet come to pass.

References i

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7654–7664, May 2022. doi:10.18653/v1/2022.acl-long.527. URL https://aclanthology.org/2022.acl-long.527.

S. C. Kleene. Representation of events in nerve nets and finite automata. Technical Report RM-704, RAND, 1951. URL https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM704.pdf.

Warren McCulloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943. doi:10.1007/BF02478259.

Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. MIT Press, 1971. URL https://archive.org/details/CounterFre_00_McNa.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is Turing-complete. *Journal of Machine Learning Research*, 22:75:1–75:35, 2021. URL http://jmlr.org/papers/v22/20-302.html.

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

M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 3770–3785, August 2021. doi:10.18653/v1/2021.acl-long.292. URL https://aclanthology.org/2021.acl-long.292.