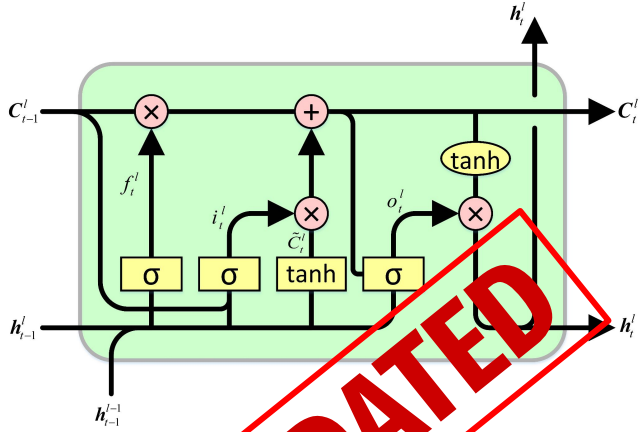
Transformers or RNNs or SSMs: Who's more sensitive?

Language, Computation and Cognition Lab

Transformers are great!

Transformers are great!

But sometimes they are stupid.



OUTDATED

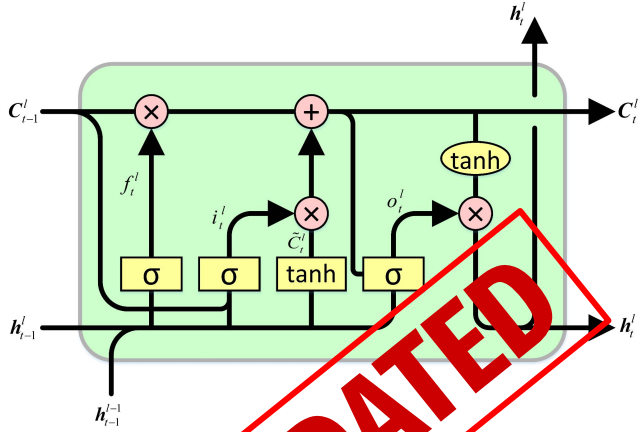
LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

And LSTMs are impractical and outdated



LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

OUTDATED

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

And LSTMs are impractical and outdated

But on some tasks they are better than Transformers!

Meanwhile State Space Models are shiny and promising

Meanwhile State Space Models are shiny and promising

But they desperately fail some tasks that are easy for Transformers!

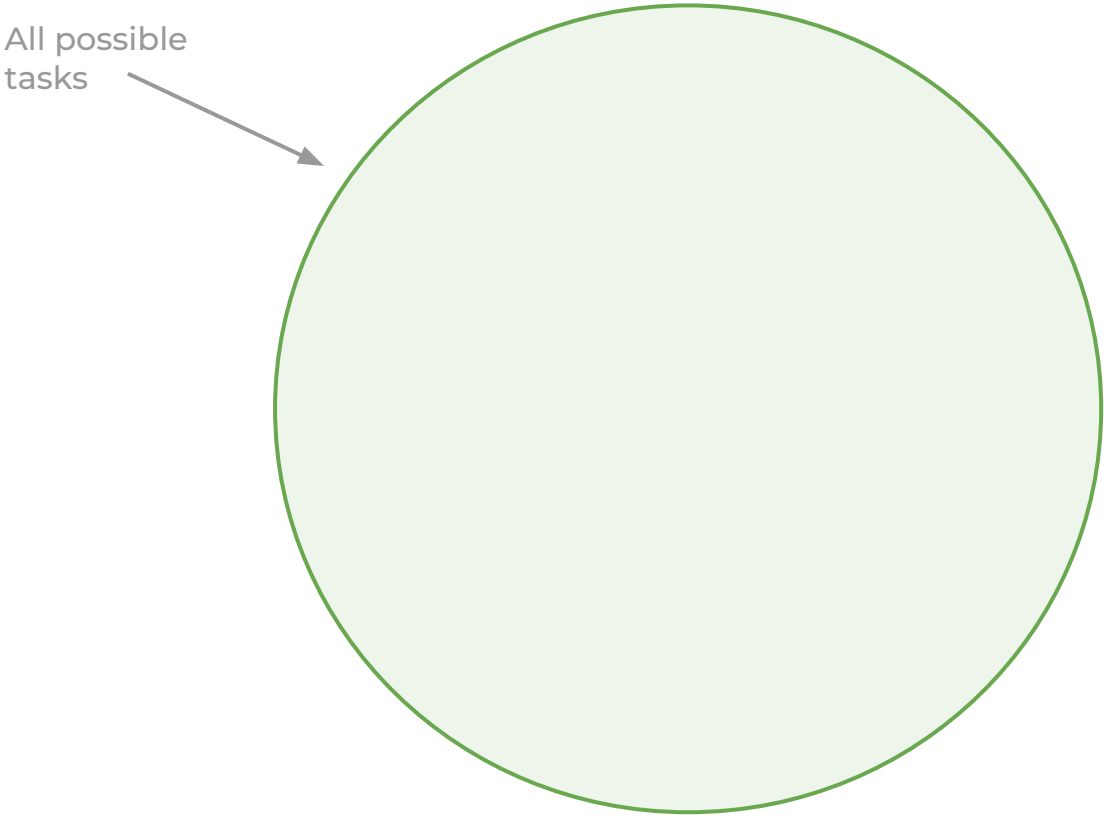
Then how can we even say that one architecture is better than the others?

Then how can we even say that one architecture is better than the others?

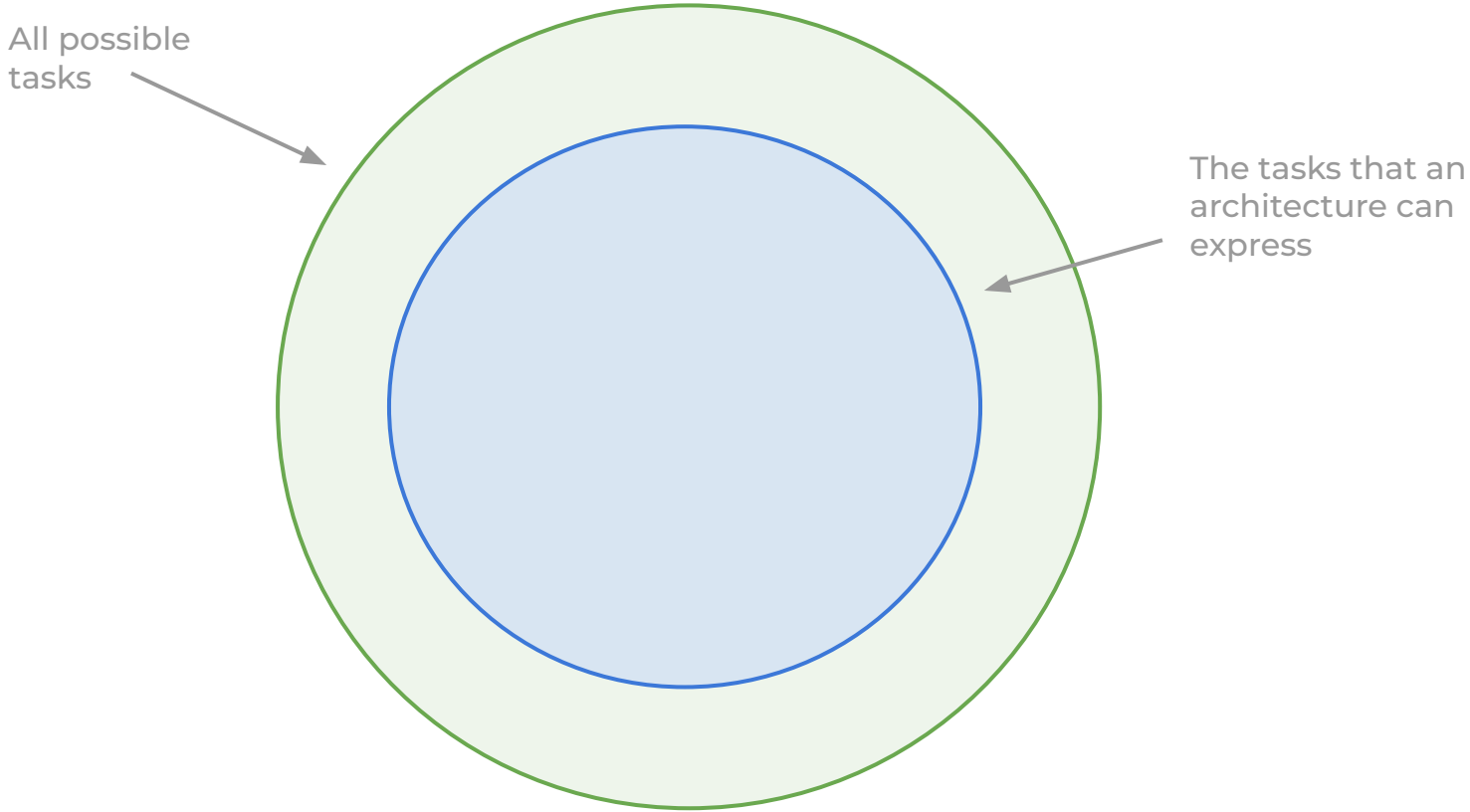
We can’t! But different architectures can solve different tasks.

And theoretical analysis is here to help!

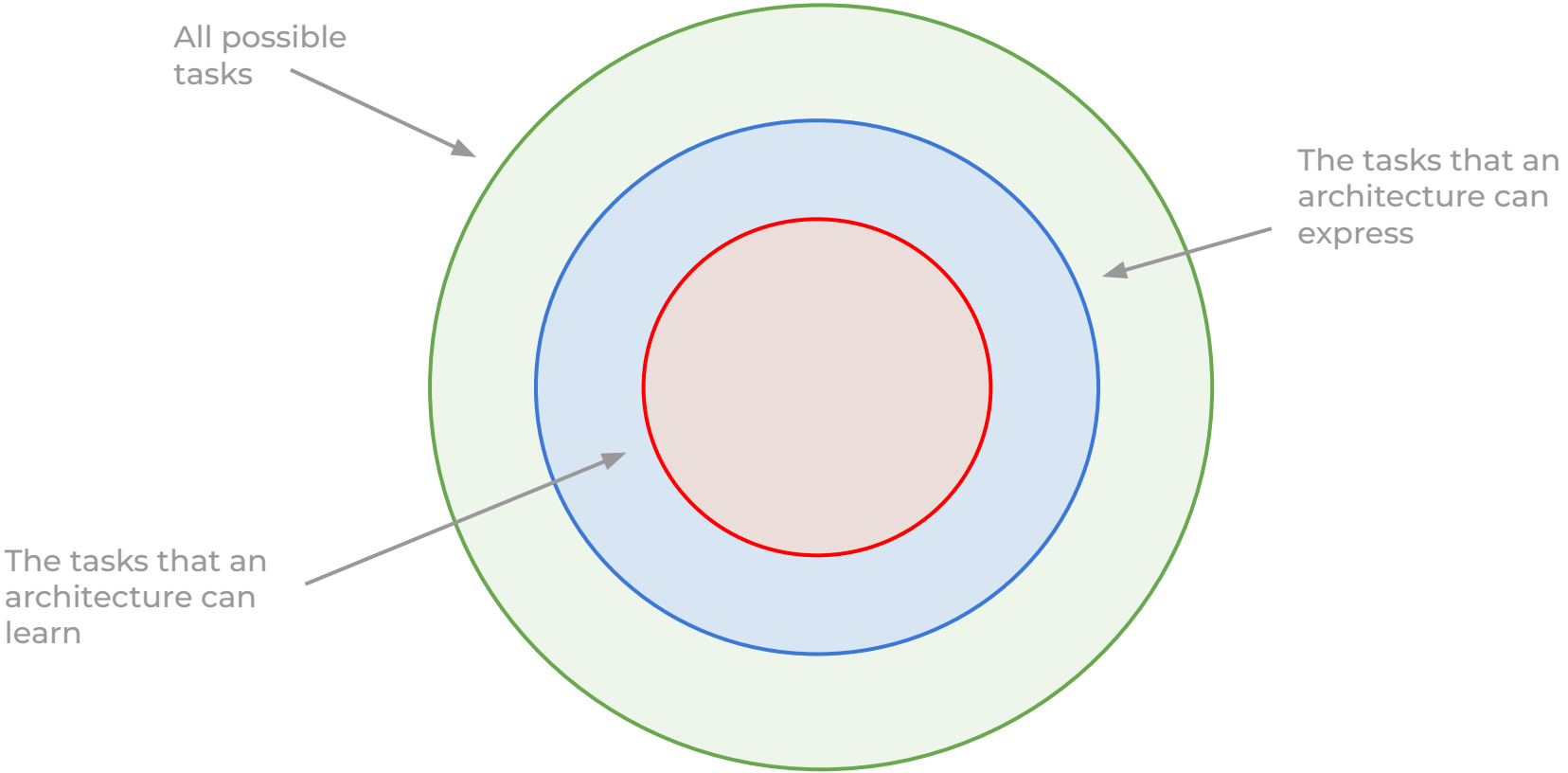
The research questions



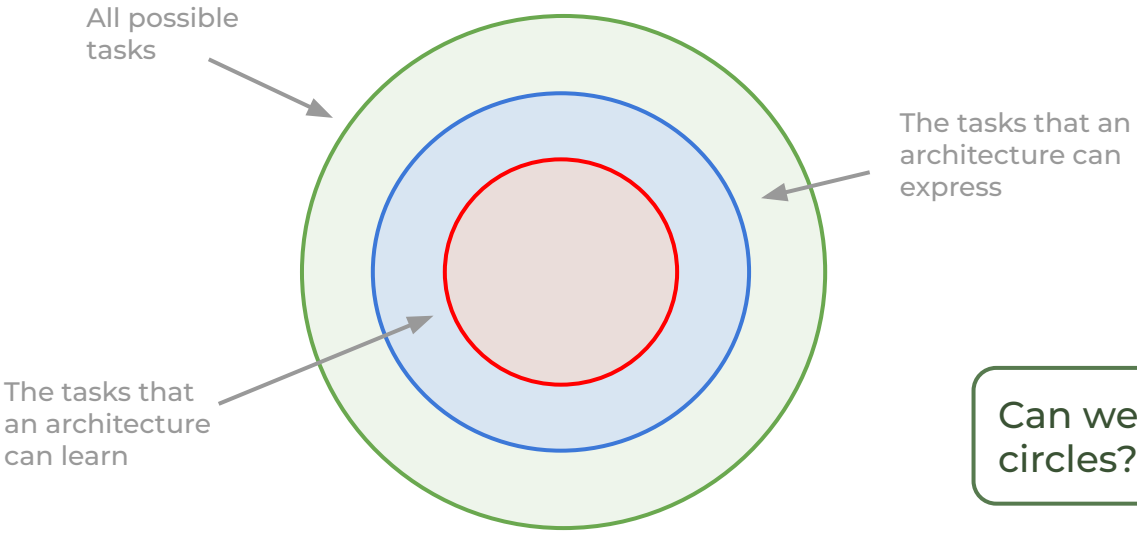
The research questions



The research questions



The research questions



Can we find the exact borders of those circles?

And understand why they are like that?

How can we formalize the tasks?

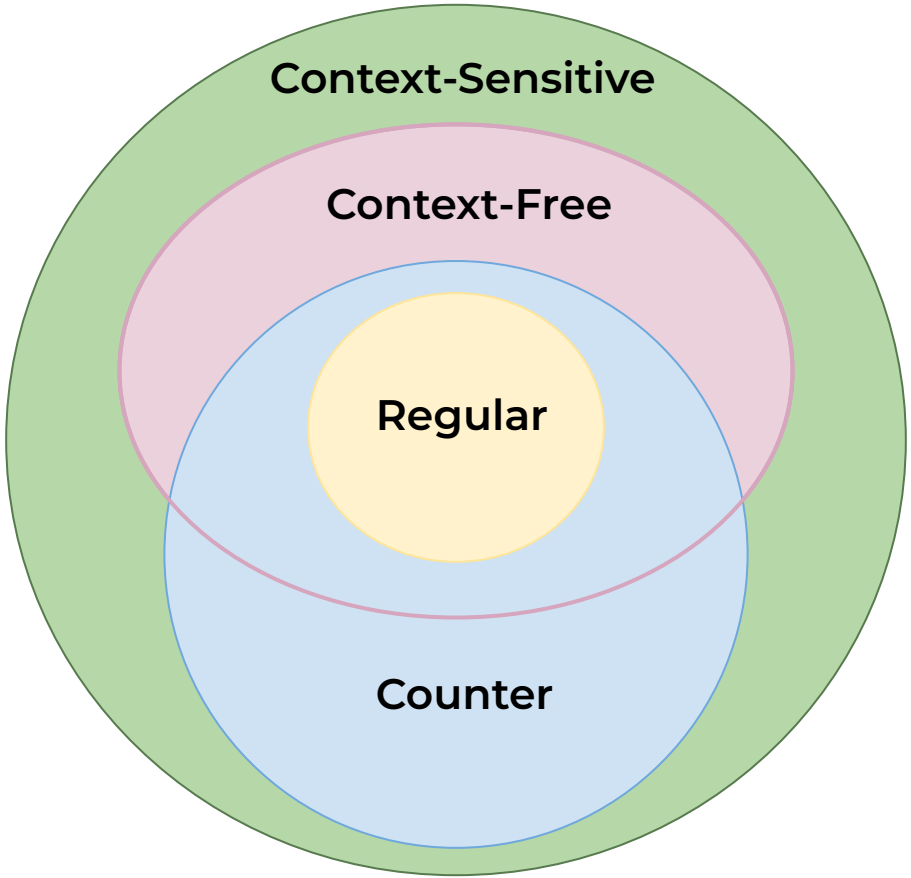
Formal Languages

Palindrome

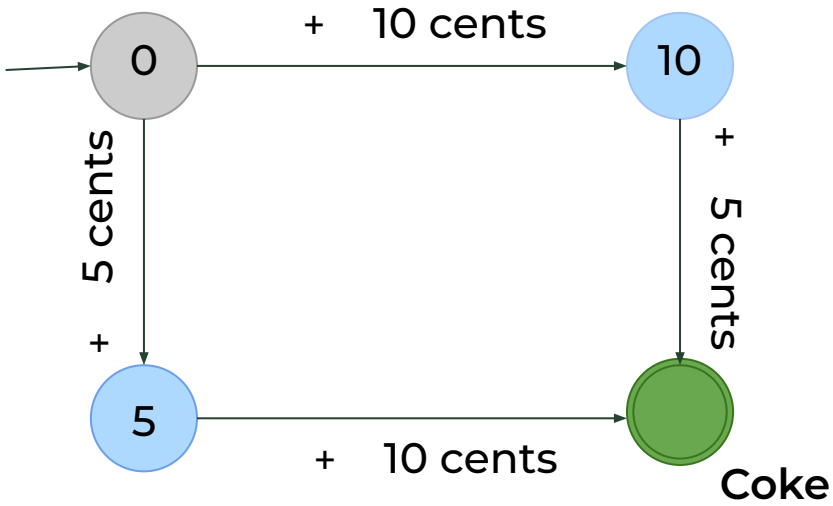
Alphabet **Σ =** {English letters}

Rule: empty string (ε) or a string s, where s^-1 = s

Palindromes = {ε, noon, level, rotator, refer, madam...}


Context-Sensitive
Context-Free
Regular
Counter

Vending Machine is a Deterministic Finite Automata!



Regular Languages

Def.

A language recognized by a Deterministic Finite Automata (DFA)

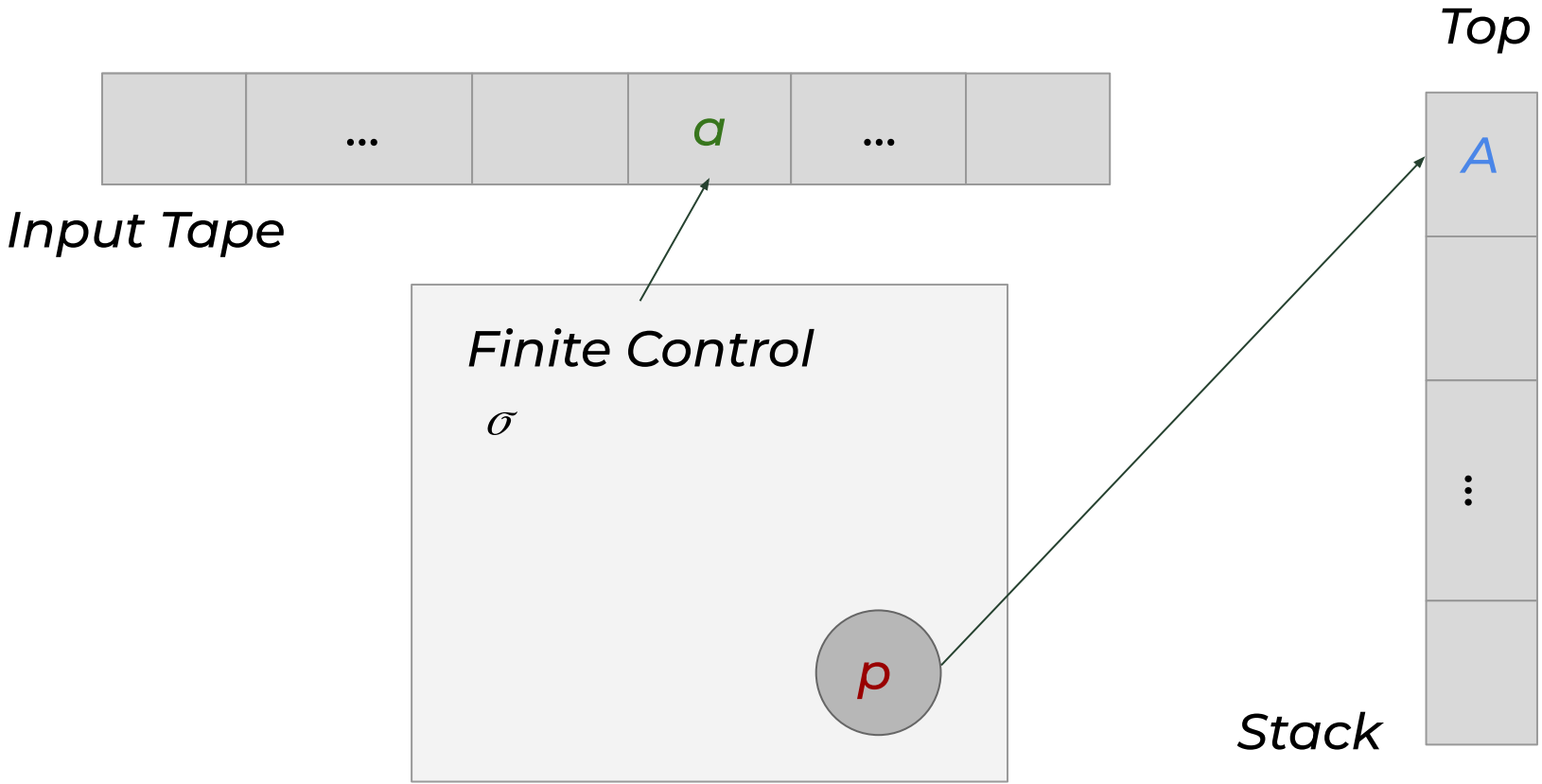
Example: Website Password

Your password should contain:
- >= 1 lower and uppercase letters
- >= 1 digits
- >= 1 special symbols

password strength: medium

Type your new password...

Stacks Make Pushdown Automata



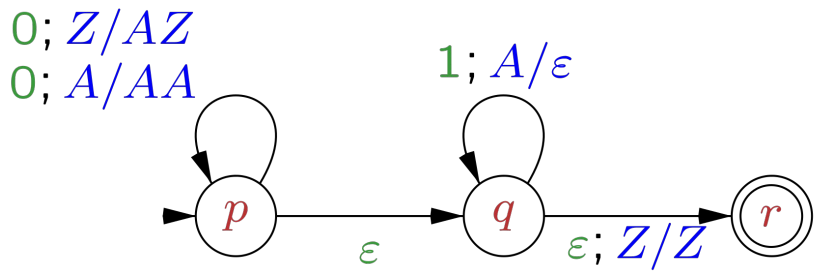
Counter Languages

Def.

A language recognized by a Pushdown Automata (PDA)

Let’s Make a PDA Together!

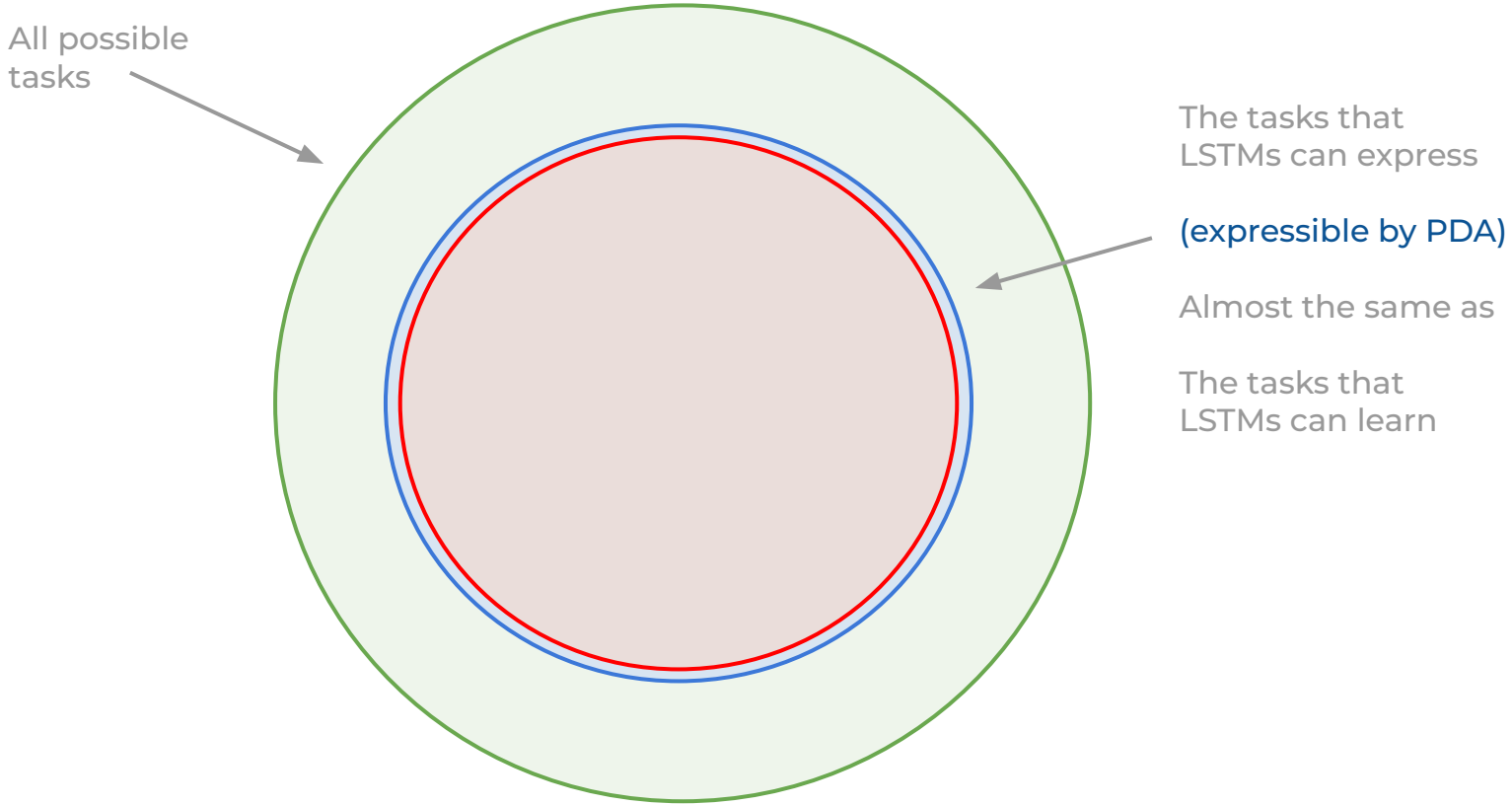
Language: {0^n 1^n | n >= 0}



LSTM is an Automata’s Big Brother

- Sequentialism
- Memory and state

The task map for LSTMs



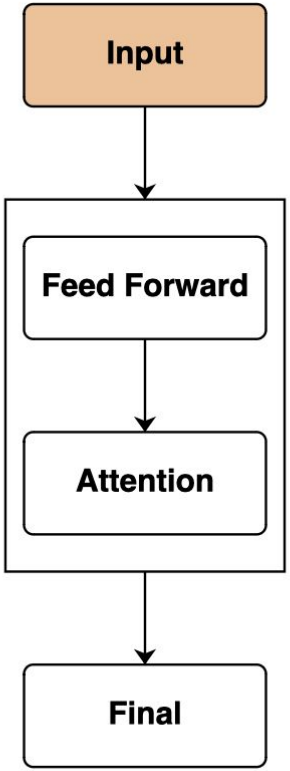
How does this look for Transformers ?

TOO MANY CHOICES

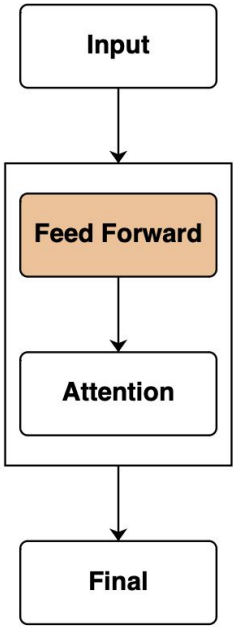
We will help you grasp this.

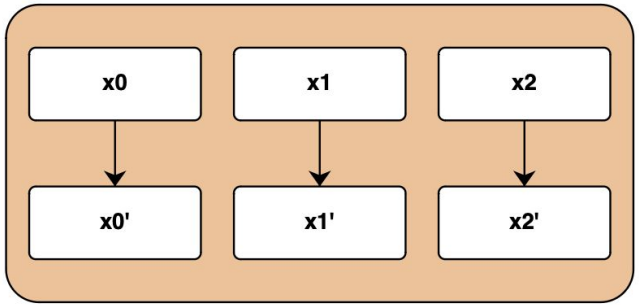
Restricted Access Sequence Processing

A new programming language, fun :)

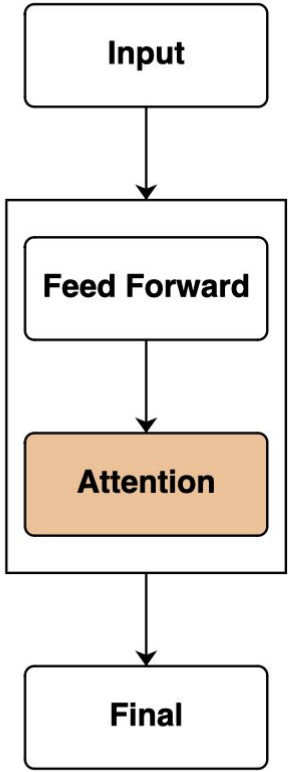


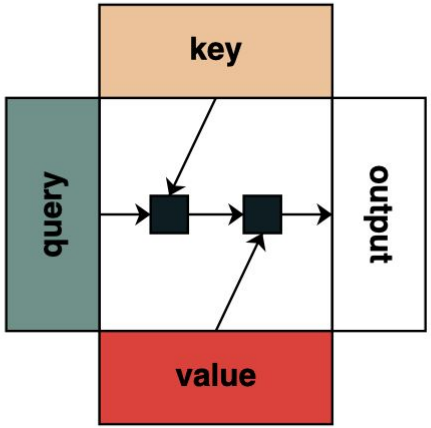
Restricted Access Sequence Processing





Restricted Access Sequence Processing





Restricted Access Sequence Processing

- Initialise tokens
- Operations on tokens
- Select pairs of tokens
- Aggregate the results of the selects
- Add layers / add heads
- Tracr : Compiles these into transformer weights

NOT THE ONLY WAY !!

And what about learnability?

If Transformers can express a language, it doesn't mean they can learn it!

The simplest example: **PARITY** function

Is the number of "1" in a bitstring even or odd?

PARITY of length 1000 is easy to express,
but nearly impossible to learn!

The reason: sensitivity

A function is **sensitive** if a slight change in the input changes the output.

Formally:

- $x \in \{\pm 1\}^n$ – bitstring of length n.
- $x^{\oplus i}$ – a bitstring identical to x, but i-th bit is reversed.
- f : a function $\{\pm 1\}^n \to \mathbb{R}$
- *Input Sensitivity*:

$$s(x, f) := \frac{1}{4} \sum_i |f(x) - f(x^{\oplus i})|^2$$

- *Average Sensitivity* $as_n(f)$ – average value of $s(x, f)$ over all inputs of length n.

The reason: sensitivity

- *Input Sensitivity*:

$$s(x, f) := \frac{1}{4} \sum_i |f(x) - f(x^{\oplus i})|^2$$

- *Average Sensitivity* $as_n(f)$ – average value of $s(x, f)$ over all inputs of length n.

Average Sensitivity = probability that changing one bit in the input changes the label (multiplied by n).

FIRST	Is the first bit of the string 0 or 1?	AS = 1
MAJORITY	Is there more 1s or 0s?	AS = sqrt(n)
PARITY	Is the number of 1s odd or even?	AS = n

The reason: sensitivity

Transformers have a **low-sensitivity bias**!

The more sensitive the function is, the harder it is for a Transformer to learn the function.

Random Transformers have low sensitivity

Random LSTMs have high sensitivity

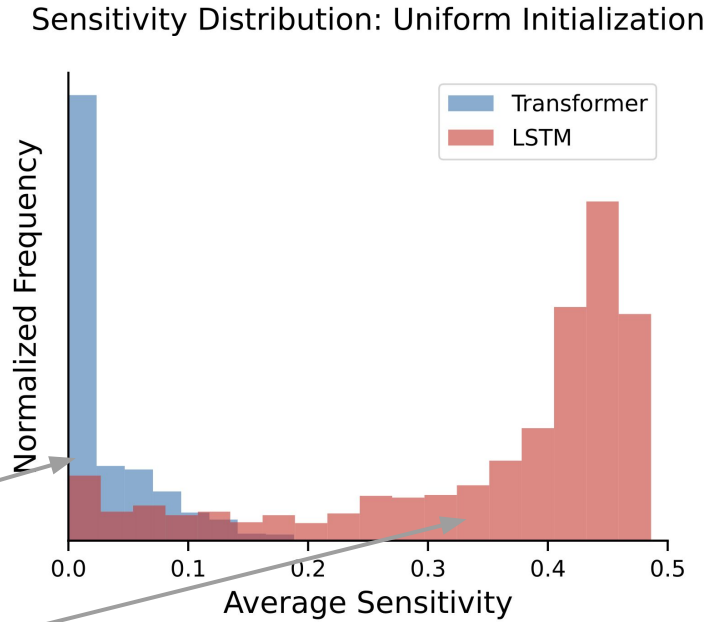


Figure from Bhattamishra et al. 2023

The reason: sensitivity

WHY?

Transformers have a **low-sensitivity bias**!

The more sensitive the function is, the harder it is for a Transformer to learn the function.

Random Transformers have low sensitivity

Random LSTMs have high sensitivity

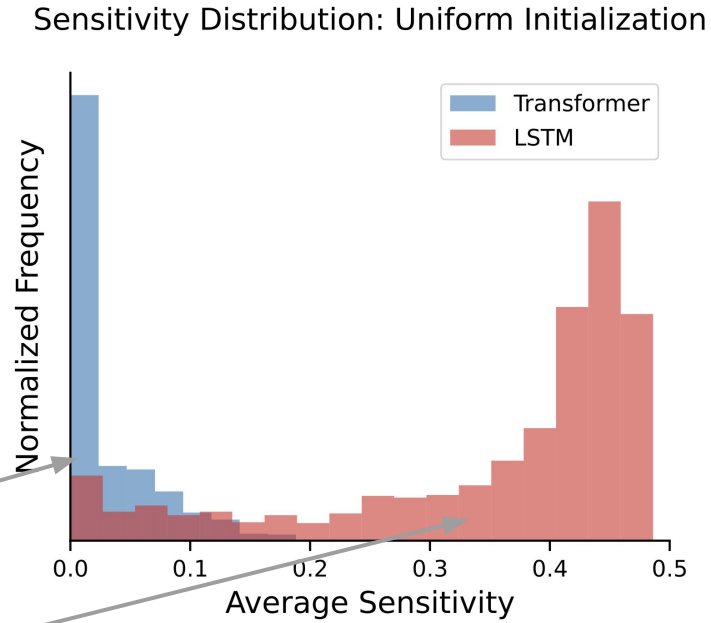
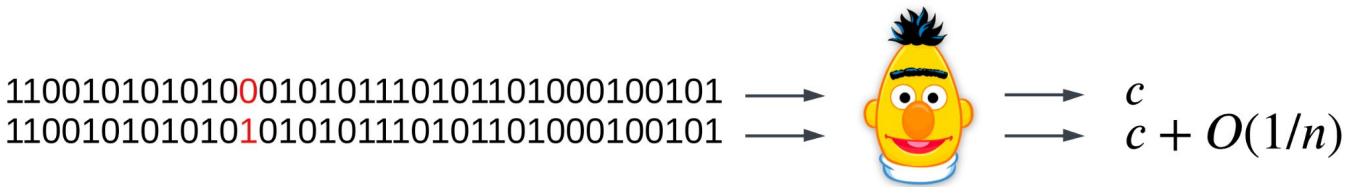


Figure from Bhattamishra et al. 2023

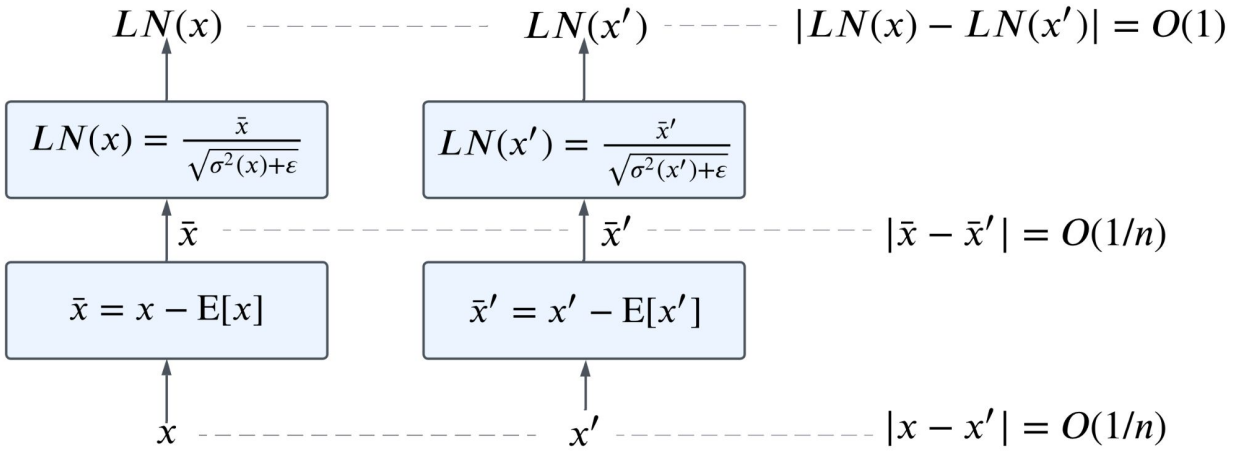
The proof that sensitive functions are hard

1. Hahn 2020: difference in outputs of a Transformer without LayerNorm on inputs differing by 1 bit is bounded as O(1/n)



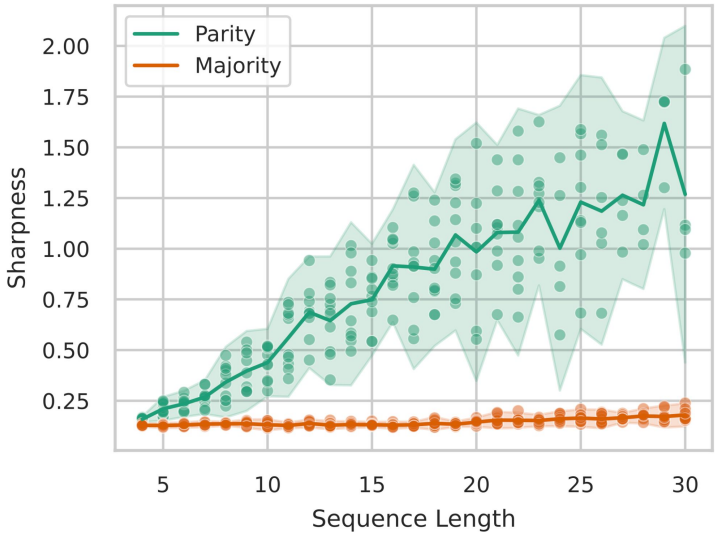
The proof that sensitive functions are hard

2. LayerNorm can mitigate this by multiplying inputs with a large coefficient, but only if the standard deviation of hidden representations is very low.

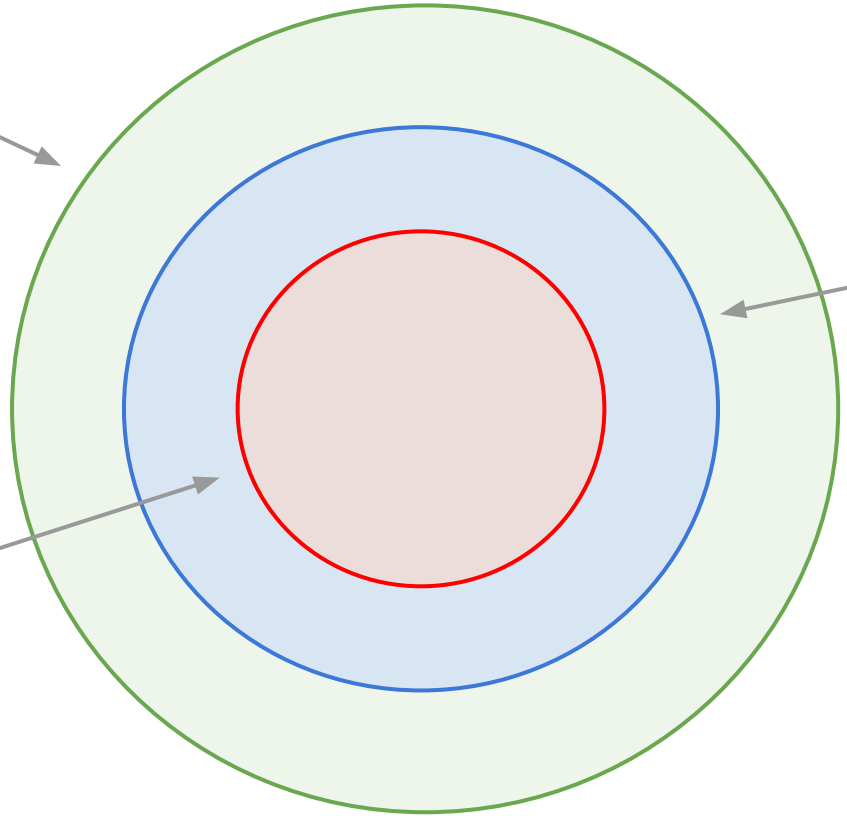


The proof that sensitive functions are hard

3. And this makes the minima of sensitive Transformers very brittle.



The task map for Transformers

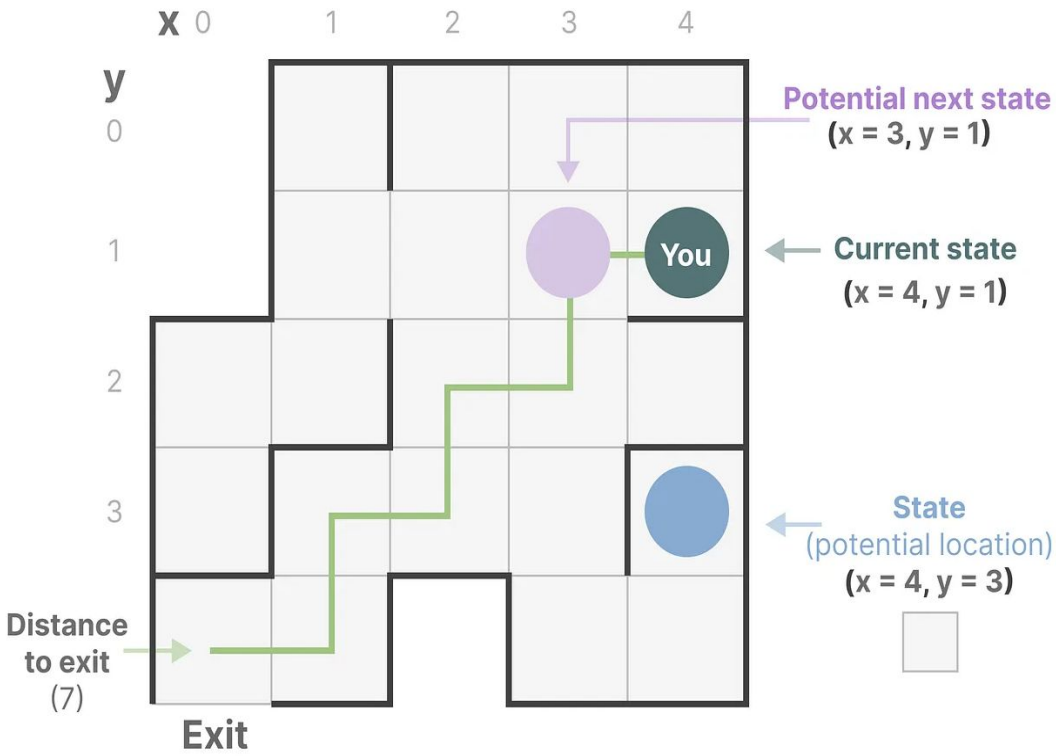


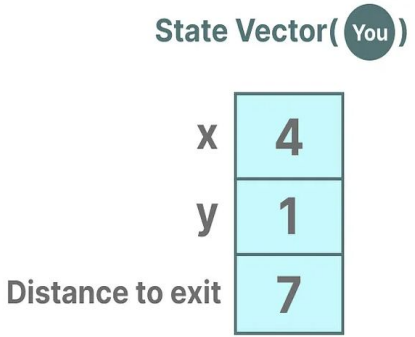
State Space Models: A New Hope

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)

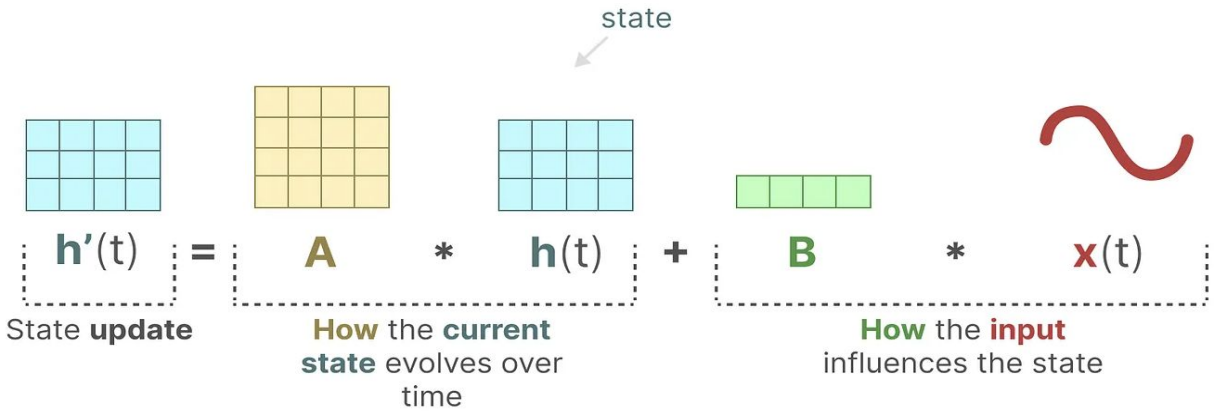
Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

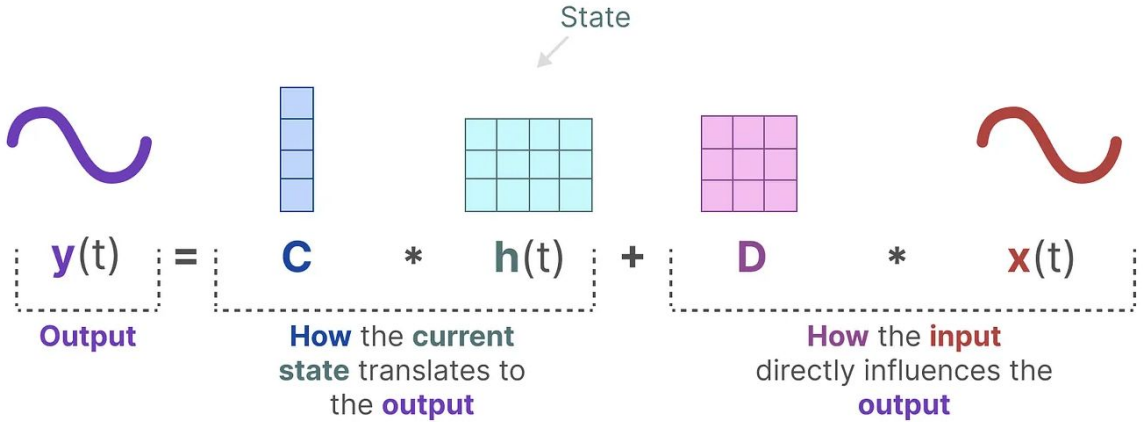
What is a State Space ?



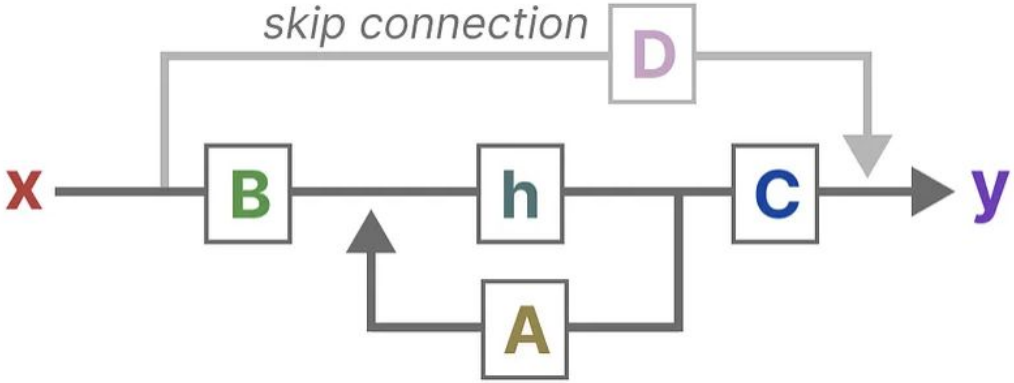


Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

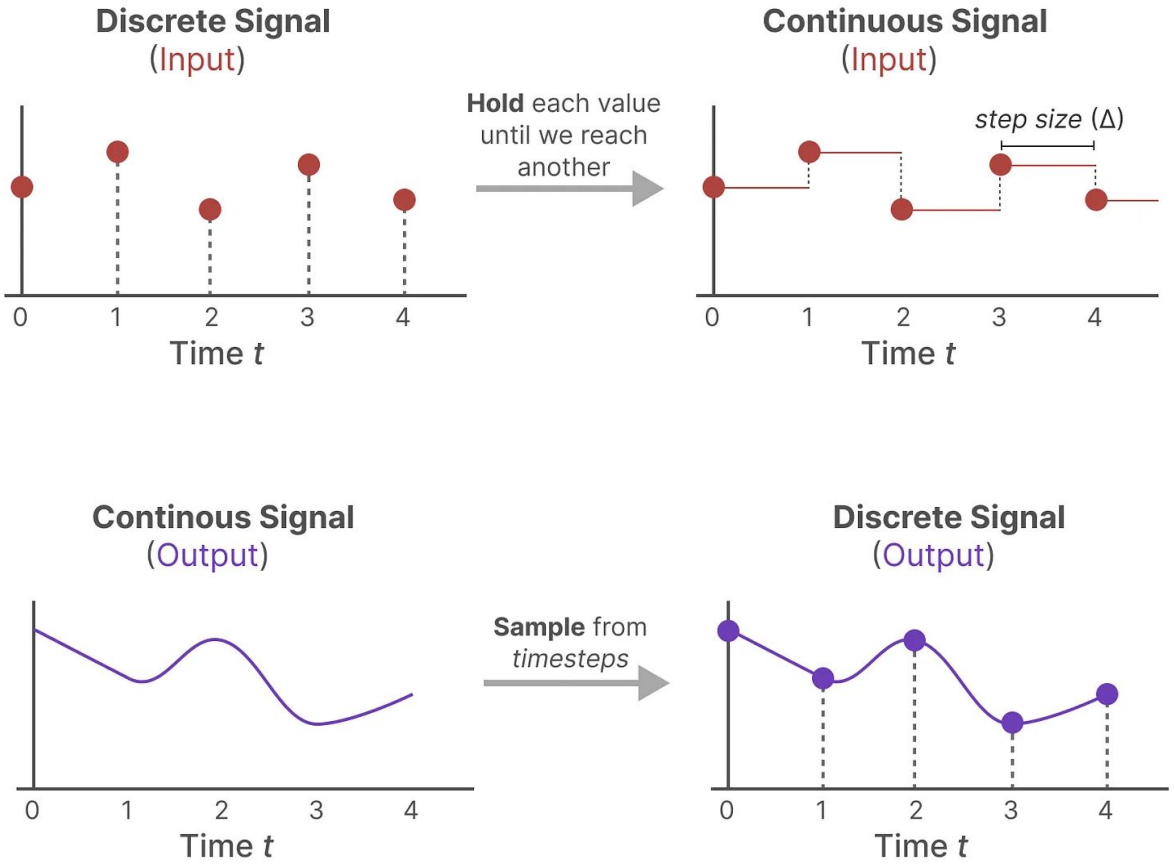




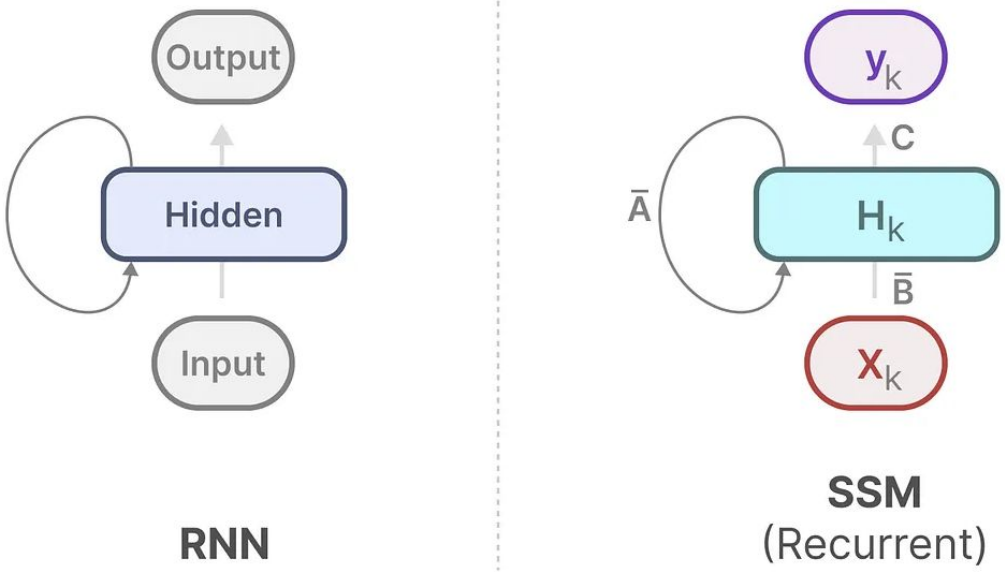
Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state



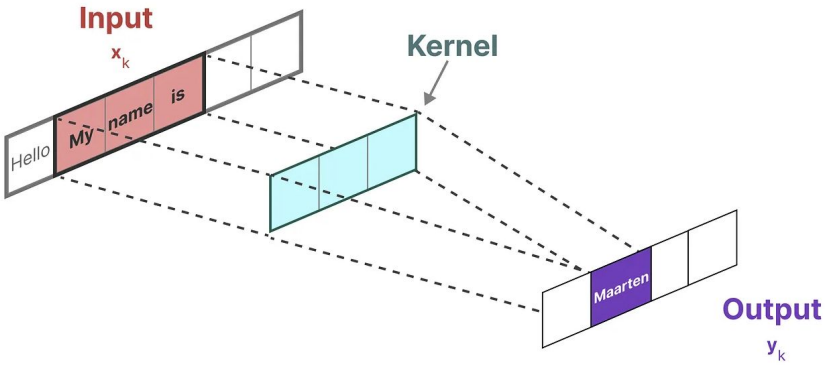
Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state



Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state



Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

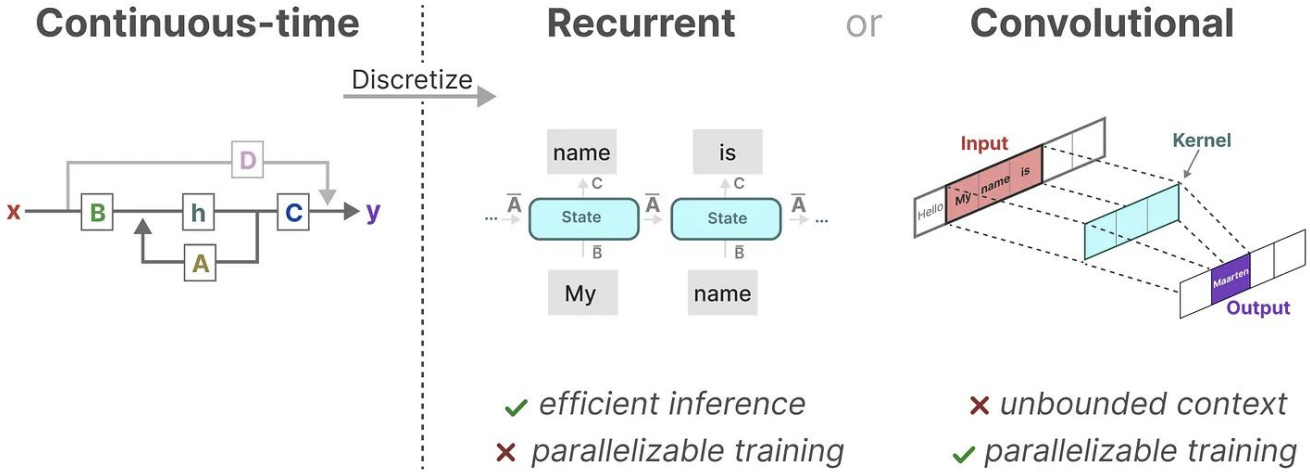


$$\text{kernel} \longrightarrow \overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{k}\overline{\mathbf{B}}, \ldots)$$

$$\mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}$$

output input kernel

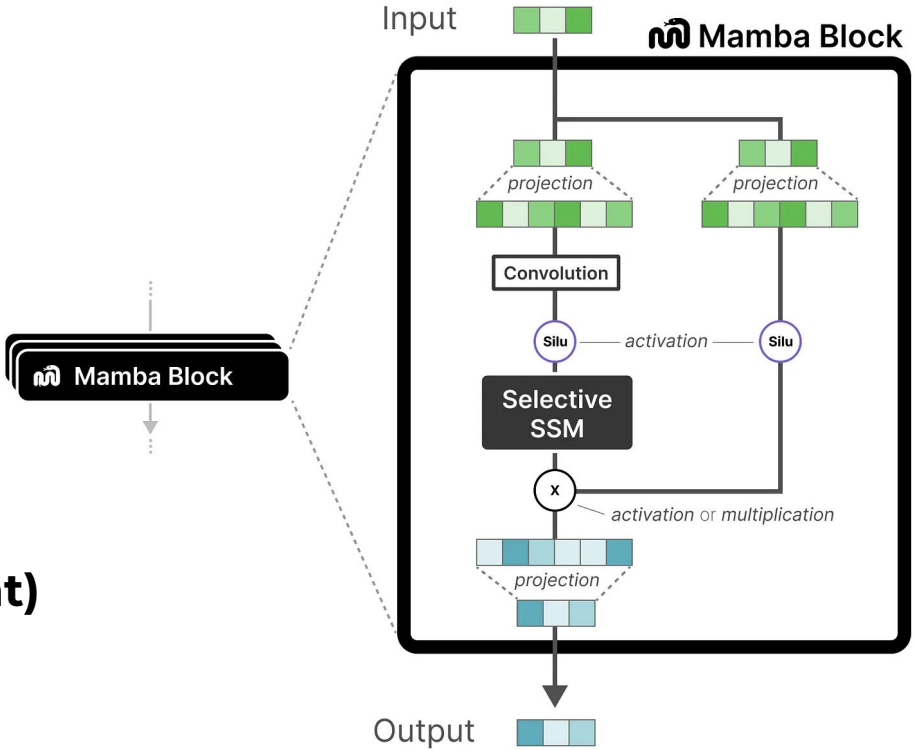
Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state



Source : https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

But why now ? and why Mamba ?

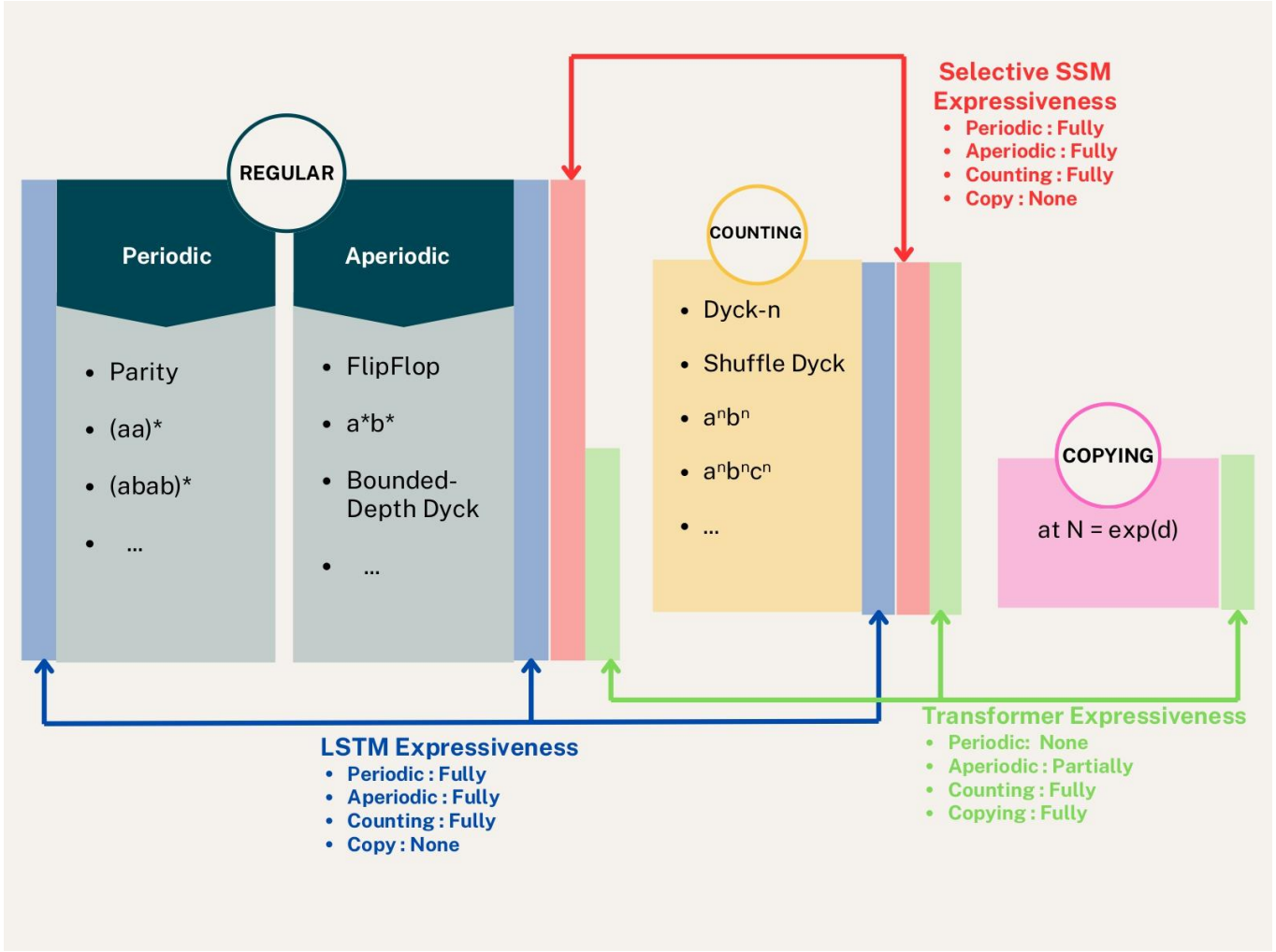
- LSTMS say Compressing is hard
- Mamba : Lets selectively compress
- Make B, C Matrices depend on the input
- Add some norms and non linearity



<u>Details skipped</u> (but important)
1. Hardware aware
2. Selective parallel scan

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)
Mamba	**Fast!** (parallelizable)	**Fast!** (scales **linearly** with sequence length + **unbounded** context)

All that glitters is not Gold


REGULAR
Periodic
Aperiodic
• Parity
• (aa)*
• (abab)*
• ...
• FlipFlop
• a*b*
• Bounded-Depth Dyck
• ...
COUNTING
• Dyck-n
• Shuffle Dyck
• $a^n b^n$
• $a^n b^n c^n$
• ...
COPYING
at N = exp(d)
Selective SSM Expressiveness
• Periodic : Fully
• Aperiodic : Fully
• Counting : Fully
• Copy : None
LSTM Expressiveness
• Periodic : Fully
• Aperiodic : Fully
• Counting : Fully
• Copy : None
Transformer Expressiveness
• Periodic: None
• Aperiodic : Partially
• Counting : Fully
• Copying : Fully

Takeaways

- Architectures have distinct abilities and weaknesses

- Identify those weaknesses
  - Theoretical analysis
  - Formal languages

- And we can make BETTER ARCHITECTURES