CSCI 361 Lecture 22:
Wrap Up and Evals 🎉

Shikha Singh

Announcements & Logistics

- Final exam in on **Dec 12 (Fri) at 1.30-3.30 pm in Schow 30A**
  - 2 hr closed-notes exam
  - Cumulative with an emphasis on decidability, undecidability, time complexity, NP hardness reductions, space complexity
  - Practice final will be released tomorrow
- Final Q&A/review session about practice exam or anything else
  - **Wed Dec 10**, **noon -1.30 pm** in Schow 30A
  - I'll order lunch/pizza!
  - Please go over assignments/practice exam over reading period and bring any questions you have

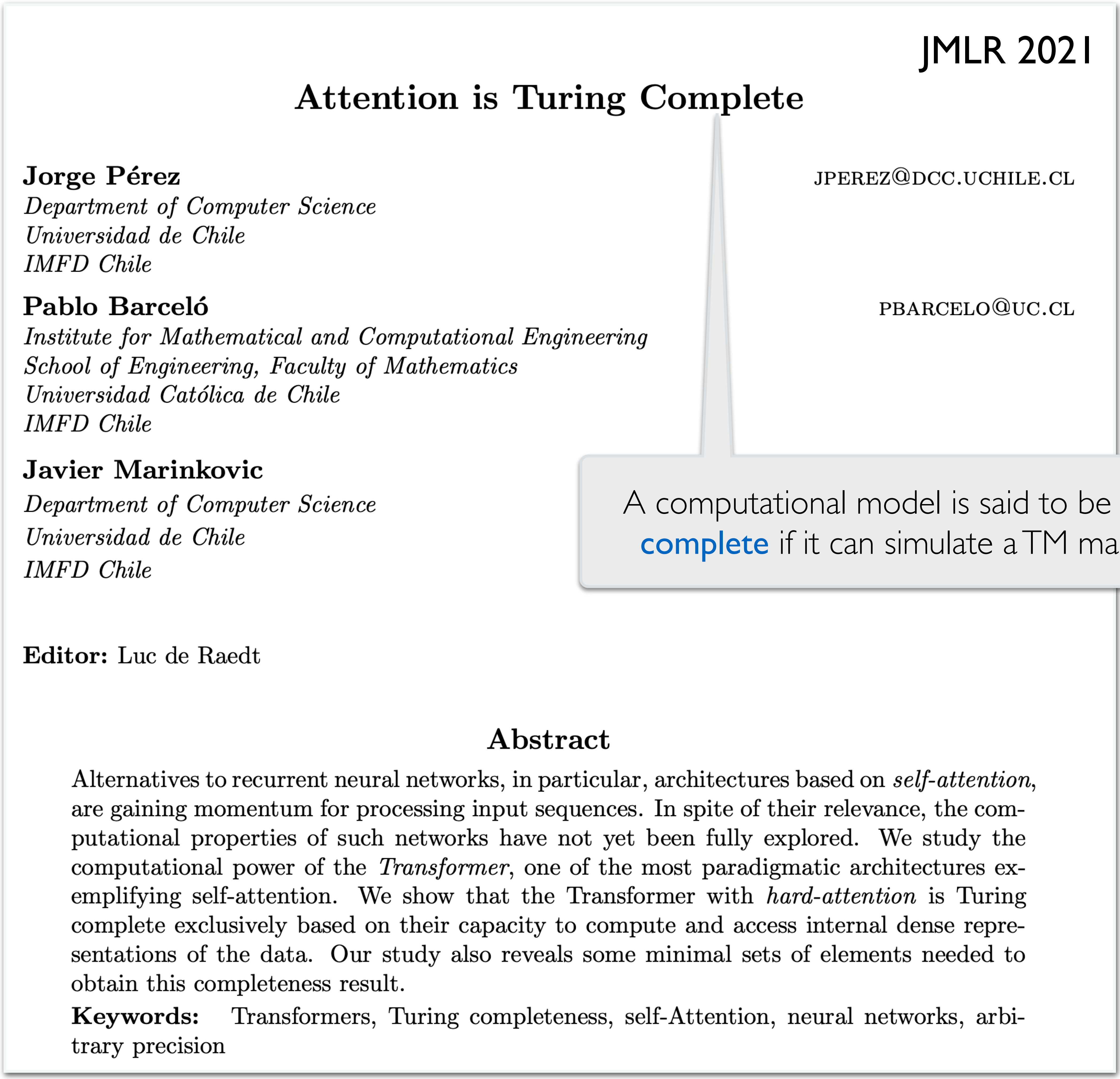
[Part 1]

Can LLMs Solve all Our Problems?

What Problems can LLMs Solve?

- Computability Power
    - What is the power of the transformer-like model of computation similar to the models we discussed in class?
    - Ignore efficiency
    - Just want type of problems can it solve
    - Where do they fall in terms of DFA/PDA/TM/beyond?

JMLR 2021



Attention is Turing Complete

Jorge Pérez JPEREZ@DCC.UCHILE.CL
Department of Computer Science
Universidad de Chile
IMFD Chile

Pablo Barceló PBARCELO@UC.CL
Institute for Mathematical and Computational Engineering
School of Engineering, Faculty of Mathematics
Universidad Católica de Chile
IMFD Chile

Javier Marinkovic
Department of Computer Science
Universidad de Chile
IMFD Chile

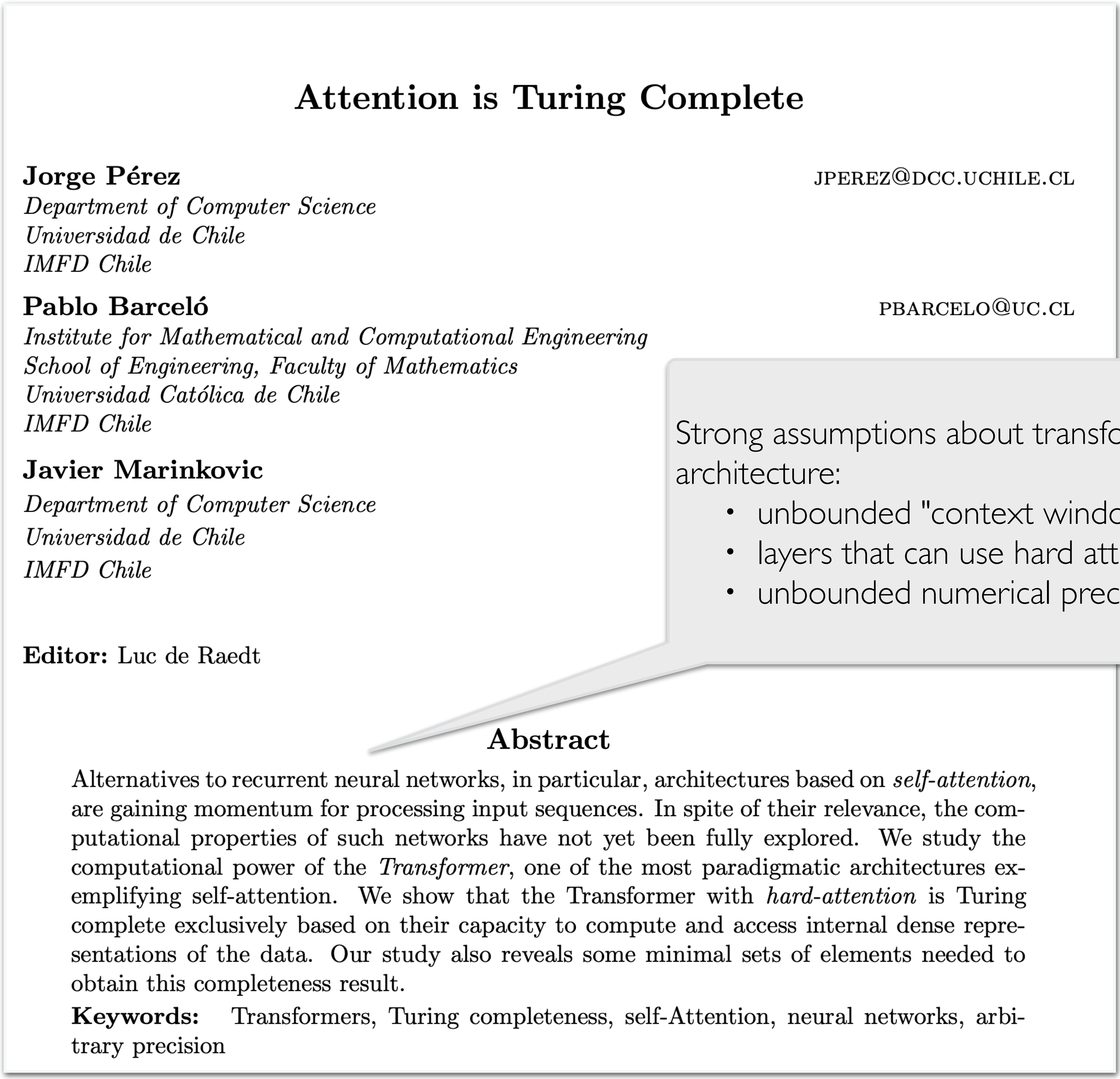
Editor: Luc de Raedt

Abstract

Alternatives to recurrent neural networks, in particular, architectures based on *self-attention*, are gaining momentum for processing input sequences. In spite of their relevance, the computational properties of such networks have not yet been fully explored. We study the computational power of the *Transformer*, one of the most paradigmatic architectures exemplifying self-attention. We show that the Transformer with *hard-attention* is Turing complete exclusively based on their capacity to compute and access internal dense representations of the data. Our study also reveals some minimal sets of elements needed to obtain this completeness result.

Keywords: Transformers, Turing completeness, self-Attention, neural networks, arbitrary precision

A computational model is said to be Turing complete if it can simulate a TM machine



Attention is Turing Complete

Jorge Pérez JPEREZ@DCC.UCHILE.CL
Department of Computer Science
Universidad de Chile
IMFD Chile

Pablo Barceló PBARCELO@UC.CL
Institute for Mathematical and Computational Engineering
School of Engineering, Faculty of Mathematics
Universidad Católica de Chile
IMFD Chile

Javier Marinkovic
Department of Computer Science
Universidad de Chile
IMFD Chile

Editor: Luc de Raedt

Strong assumptions about transformer architecture:

- unbounded "context windows"
- layers that can use hard attention
- unbounded numerical precision

Abstract

Alternatives to recurrent neural networks, in particular, architectures based on *self-attention*, are gaining momentum for processing input sequences. In spite of their relevance, the computational properties of such networks have not yet been fully explored. We study the computational power of the *Transformer*, one of the most paradigmatic architectures exemplifying self-attention. We show that the Transformer with *hard-attention* is Turing complete exclusively based on their capacity to compute and access internal dense representations of the data. Our study also reveals some minimal sets of elements needed to obtain this completeness result.

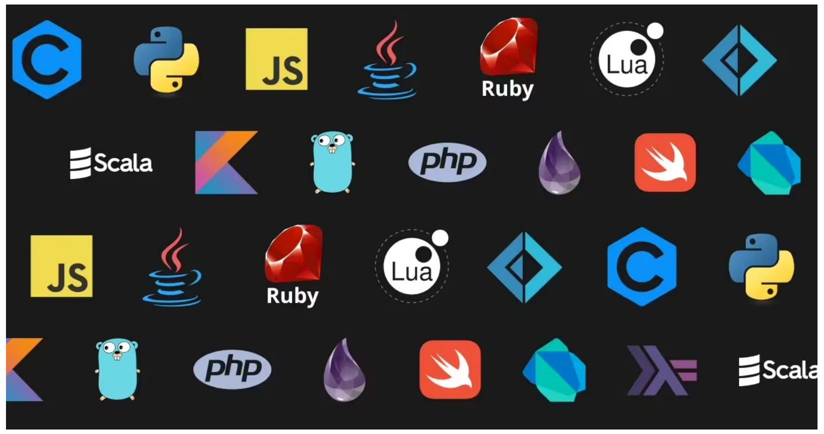
Keywords: Transformers, Turing completeness, self-Attention, neural networks, arbitrary precision

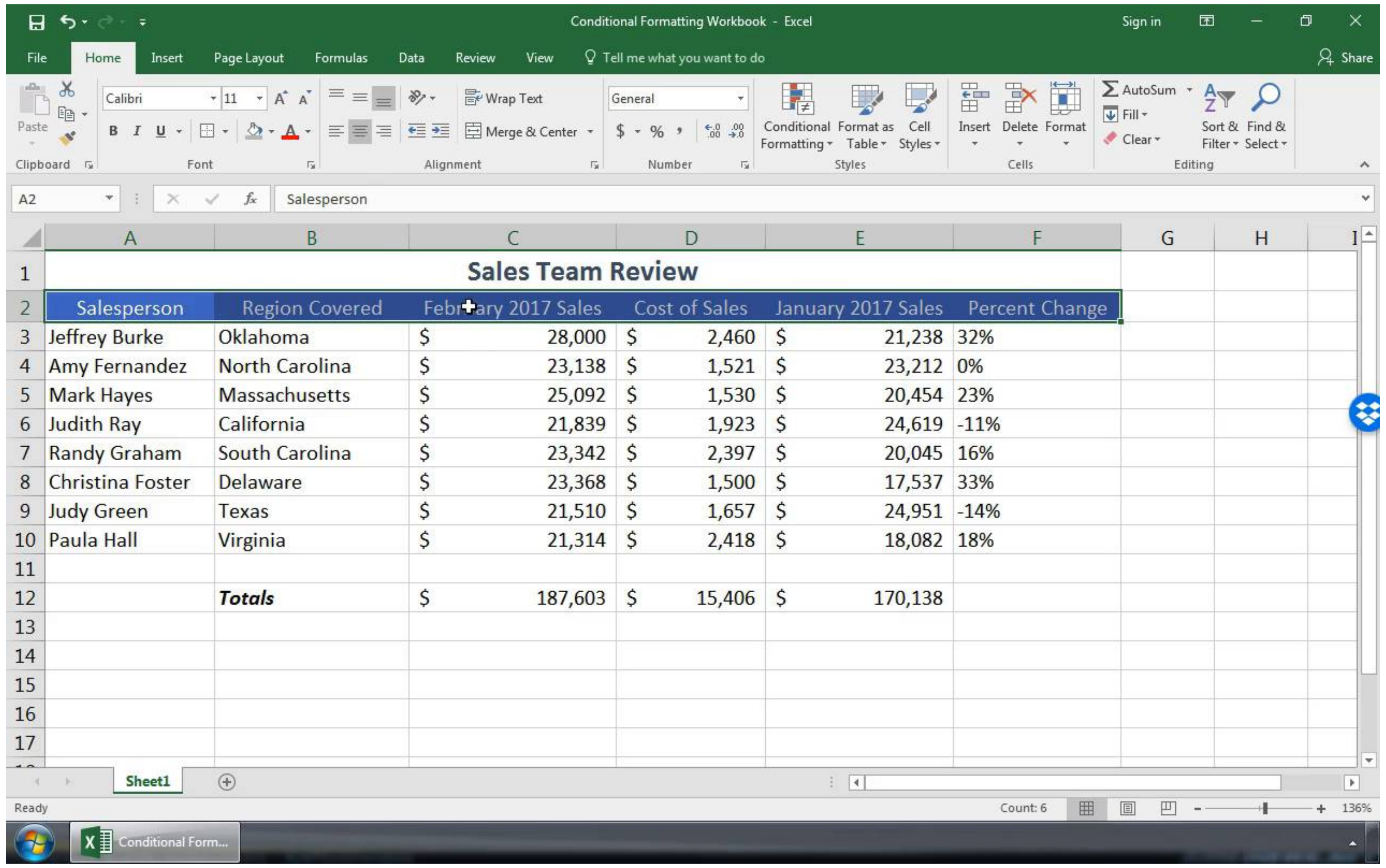
Overall Takeaways

- Parameters of simulating a TM are explicitly hard-coded in the architecture (not learned)
- Real world architectures have limitations not covered in the proof
- Highlight the expressive power of attention-based LLMs
- Still governed by Church-Turing thesis
    - Cannot solve the halting or other undecidable problems

Turing Complete Models

- Is being Turing-complete that impressive?
    - Means as expressive as a general-purpose computer



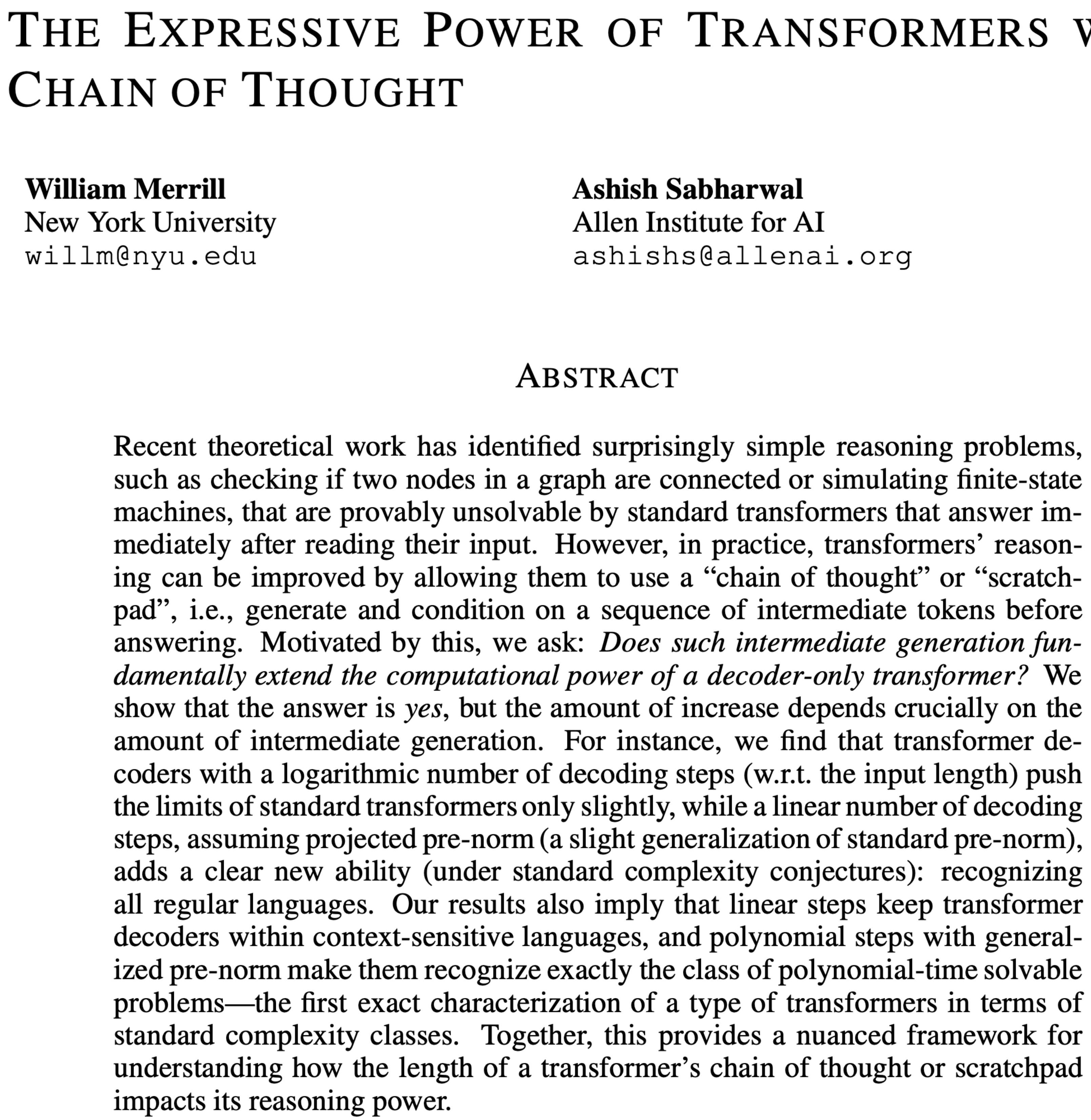

Conditional Formatting Workbook - Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Tell me what you want to do
Sheet1
Ready

Sales Team Review

Salesperson	Region Covered	February 2017 Sales	Cost of Sales	January 2017 Sales	Percent Change
Jeffrey Burke	Oklahoma	$ 28,000	$ 2,460	$ 21,238	32%
Amy Fernandez	North Carolina	$ 23,138	$ 1,521	$ 23,212	0%
Mark Hayes	Massachusetts	$ 25,092	$ 1,530	$ 20,454	23%
Judith Ray	California	$ 21,839	$ 1,923	$ 24,619	-11%
Randy Graham	South Carolina	$ 23,342	$ 2,397	$ 20,045	16%
Christina Foster	Delaware	$ 23,368	$ 1,500	$ 17,537	33%
Judy Green	Texas	$ 21,510	$ 1,657	$ 24,951	-14%
Paula Hall	Virginia	$ 21,314	$ 2,418	$ 18,082	18%
	Totals	$ 187,603	$ 15,406	$ 170,138	

Complexity Theoretic Power

- Complexity question
  - What is the complexity theoretic power of LLMs?
  - Are they able to solve more problems than the already existing classical notion of efficient algorithms?
  - In particular, can they go beyond the class **P**



The Expressive Power of Transformers with Chain of Thought

William Merrill
New York University
willm@nyu.edu

Ashish Sabharwal
Allen Institute for AI
ashishs@allenai.org

Abstract

Recent theoretical work has identified surprisingly simple reasoning problems, such as checking if two nodes in a graph are connected or simulating finite-state machines, that are provably unsolvable by standard transformers that answer immediately after reading their input. However, in practice, transformers' reasoning can be improved by allowing them to use a "chain of thought" or "scratchpad", i.e., generate and condition on a sequence of intermediate tokens before answering. Motivated by this, we ask: *Does such intermediate generation fundamentally extend the computational power of a decoder-only transformer?* We show that the answer is *yes*, but the amount of increase depends crucially on the amount of intermediate generation. For instance, we find that transformer decoders with a logarithmic number of decoding steps (w.r.t. the input length) push the limits of standard transformers only slightly, while a linear number of decoding steps, assuming projected pre-norm (a slight generalization of standard pre-norm), adds a clear new ability (under standard complexity conjectures): recognizing all regular languages. Our results also imply that linear steps keep transformer decoders within context-sensitive languages, and polynomial steps with generalized pre-norm make them recognize exactly the class of polynomial-time solvable problems—the first exact characterization of a type of transformers in terms of standard complexity classes. Together, this provides a nuanced framework for understanding how the length of a transformer's chain of thought or scratchpad impacts its reasoning power.

ICLR 2024

Efficiency of LLMs

- [Merrill & Sabharwal 2023] Limitations of "standard" transformers (even with ideal parameters)
    - Cannot simulate DFAs or solve sequential reasoning tasks
    - Intuition: standard transformers answer right away and lack intermediate recurrent state
- To improve upon these limitations, study of transformers with "chain of thought" or "scratchpad" to allow for recurrent state
    - Complexity depends on number of such steps

Efficiency of LLMs

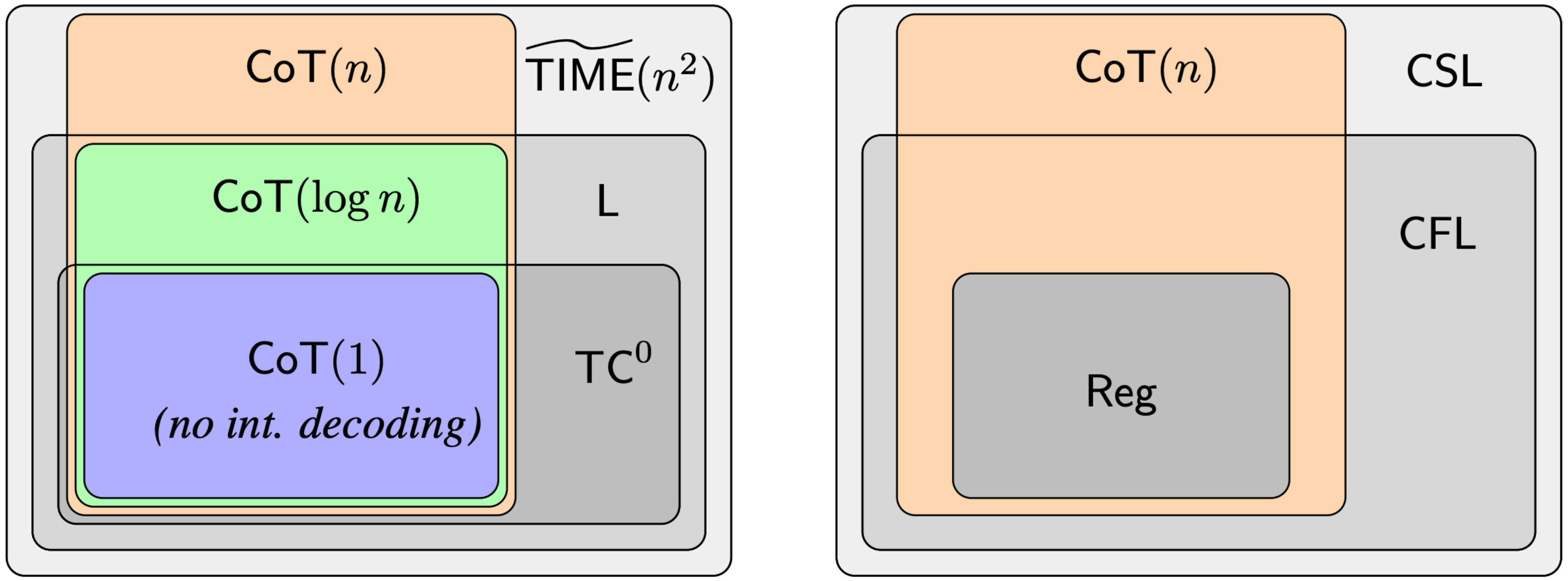


Figure 1: Summary of results: transformers with intermediate generation against various classes of formal languages. A logarithmic number of chain-of-thought steps remains in log-space (L). A linear number of steps adds more power, enabling recognizing all regular languages (Reg), but is contained within context-sensitive languages (CSL). We assume context-free languages (CFL) require $\tilde{\omega}(n^2)$ time to recognize. Some regions with area in the plot are not known to be non-empty.

Merrill & Sabharwal 2024

Efficiency of LLMs

- **$O(1)$ intermediate steps.** Standard models within class TC_0
  - Example problems in class: sorting n-bit numbers, multiplying two n-bit numbers, etc
- **Logarithmic steps.** Class **L** (log space complexity class) but still cannot solve simple problems like graph-connectivity
- **Linear steps.** Class **NC1** can simulate a finite automata
- **Polynomial steps.** Class **P** (with some strong assumptions)
- **Takeaway.** Most realistic models are quite limited (not even DFA-like); allowing quite generous extensions is still within **P**

Empirical Success

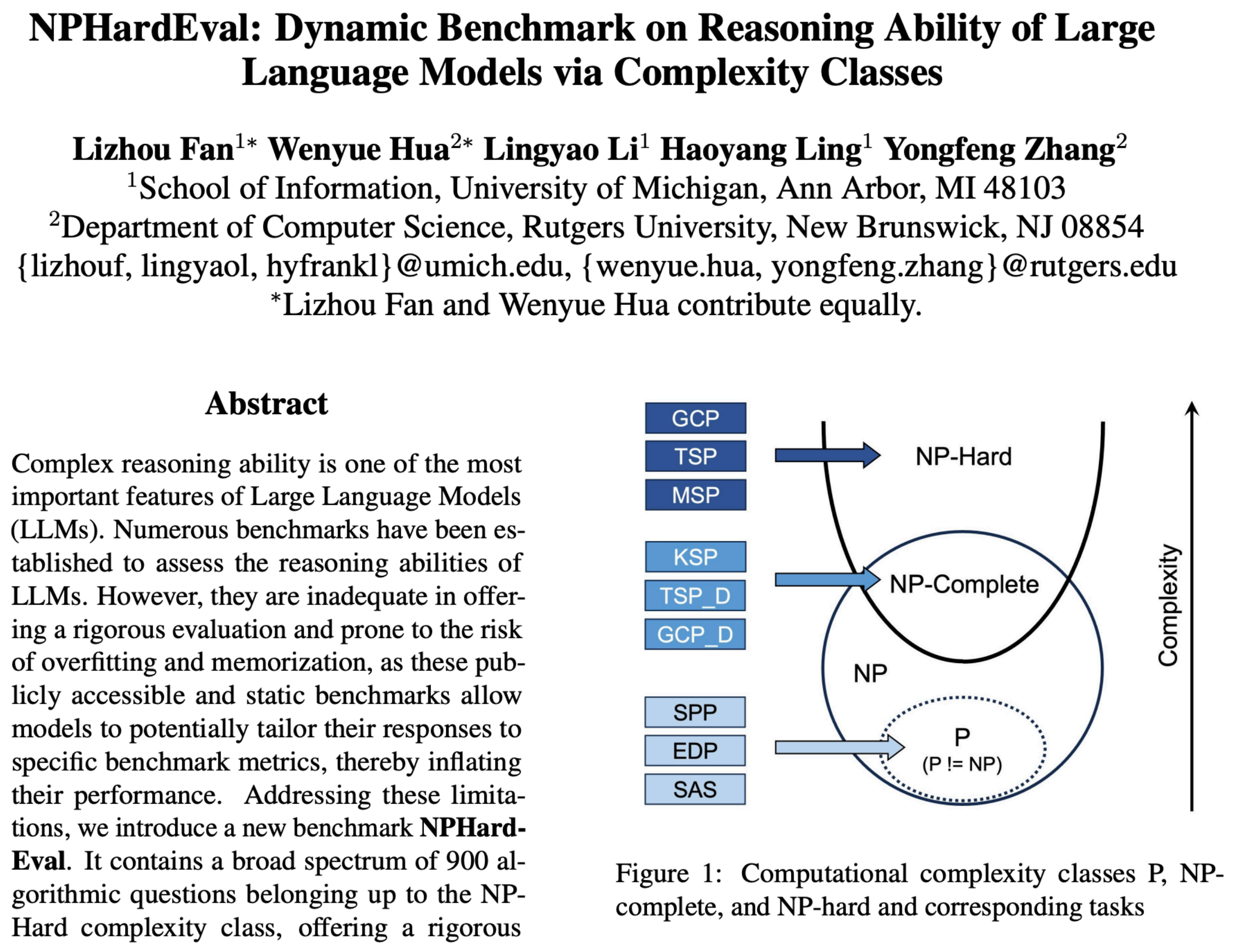
- Exciting recent work on theoretical complexity bounds
- How about empirical success?

Empirical Benchmarking

- [Fan et al. 2024] Evaluating reasoning difficulty of problems from three complexity class empirically
    - **P**: Sorted array search, edit-distance, shortest path
    - **NP**-complete: TSP, 3-coloring, etc.
    - **NP**-hard optimization problems
- Every small sparse graphs: starting from 6 nodes, 6 edges to 15 nodes, 24 edges
- Findings: accuracy drops sharply and failure rate increases as complexity increases

Does Fine Tuning Help?

- A little with tasks within **P**
- Hurts performance on NP complete and NP hard tasks



NPHardEval: Dynamic Benchmark on Reasoning Ability of Large Language Models via Complexity Classes

Lizhou Fan[1*] **Wenyue Hua**[2*] **Lingyao Li**[1] **Haoyang Ling**[1] **Yongfeng Zhang**[2]
[1]School of Information, University of Michigan, Ann Arbor, MI 48103
[2]Department of Computer Science, Rutgers University, New Brunswick, NJ 08854
{lizhouf, lingyaol, hyfrankl}@umich.edu, {wenyue.hua, yongfeng.zhang}@rutgers.edu
*Lizhou Fan and Wenyue Hua contribute equally.

Abstract

Complex reasoning ability is one of the most important features of Large Language Models (LLMs). Numerous benchmarks have been established to assess the reasoning abilities of LLMs. However, they are inadequate in offering a rigorous evaluation and prone to the risk of overfitting and memorization, as these publicly accessible and static benchmarks allow models to potentially tailor their responses to specific benchmark metrics, thereby inflating their performance. Addressing these limitations, we introduce a new benchmark **NPHard-Eval**. It contains a broad spectrum of 900 algorithmic questions belonging up to the NP-Hard complexity class, offering a rigorous

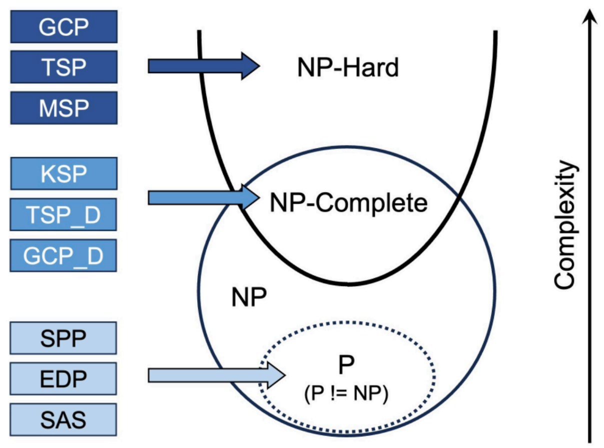
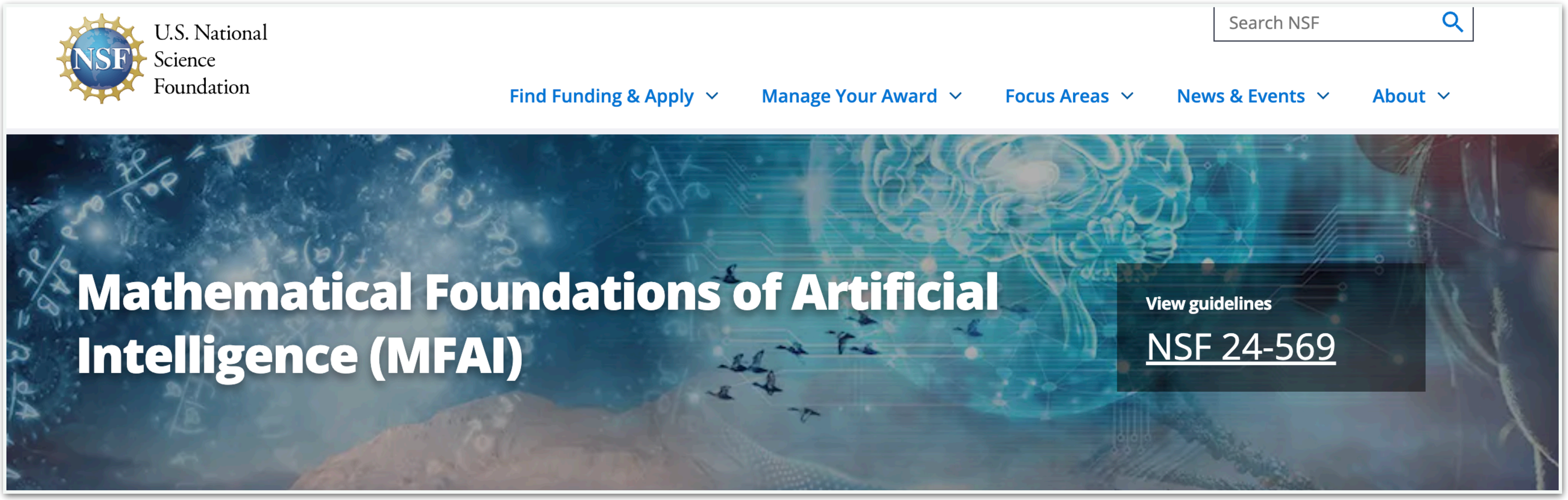


Figure 1: Computational complexity classes P, NP-complete, and NP-hard and corresponding tasks

> *"While fine-tuning yields improvements in solving polynomial time problems, its impact on the more complex NP-complete and NP-hard problems are negative. This suggests the inherent difficulty of hacking NP-complete, and potentially NP-hard, problems through the basic fine-tuning with question-and-answer approach."*

What Problems can LLMs Solve?

- We don't know yet but likely quite restricted (well within **P**)
- Growing need for mathematical foundations of AI
  - Need to understand their power and limitations



U.S. National Science Foundation

Mathematical Foundations of Artificial Intelligence (MFAI)

View guidelines

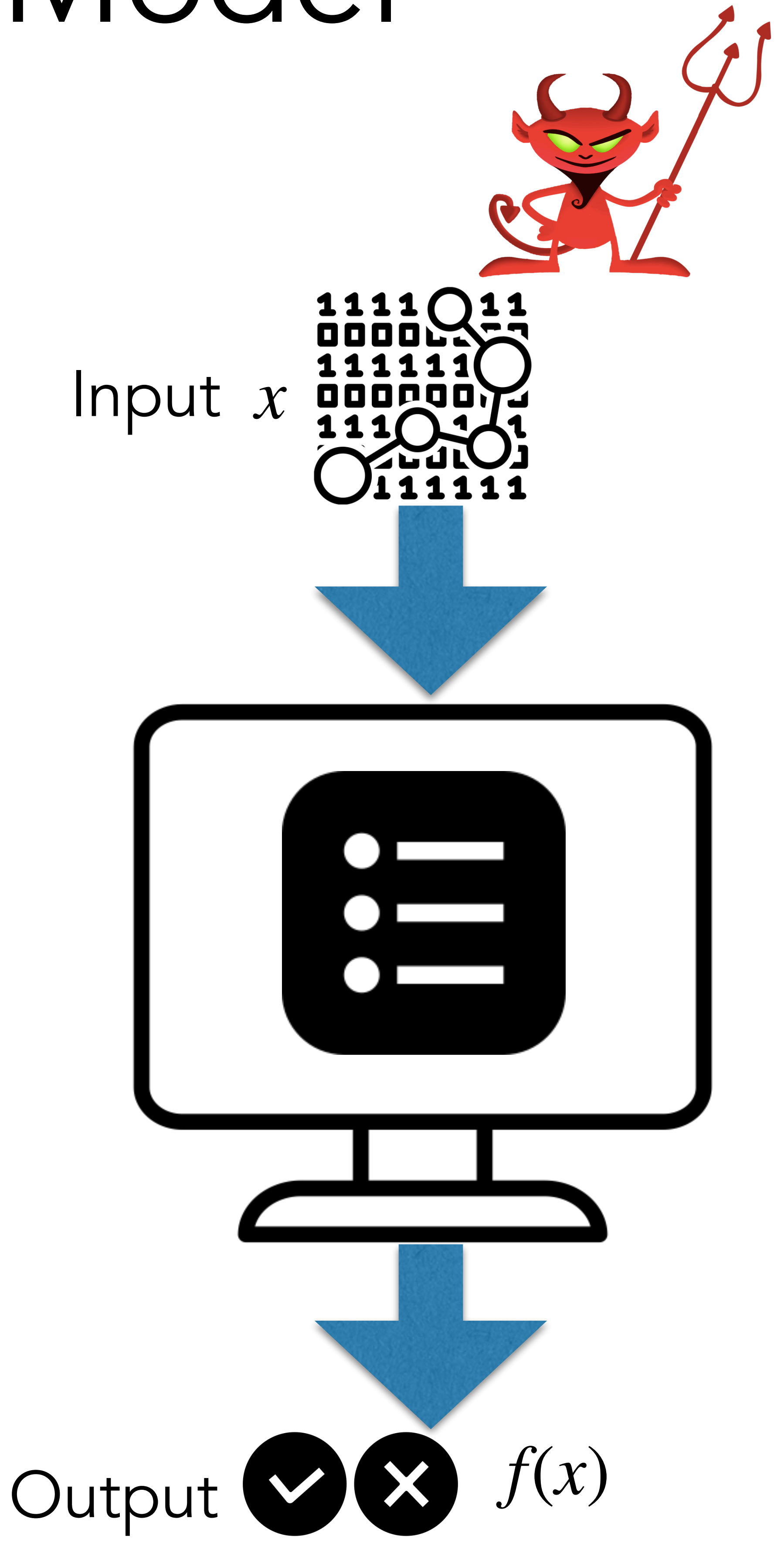
NSF 24-569

[Theory CS]

Machine-Learning Augmented Algorithms

Algorithm Analysis Model

- For any algorithm, some inputs are easy, others are hard
- **Worst-case paradigm:** Performance measured as the **maximum number of steps** taken on any input of size n
  - Adversarial analysis



Why Give Worst Case Guarantees?

- Worst-case analysis: dominant algorithm design paradigm

Powerful guarantee

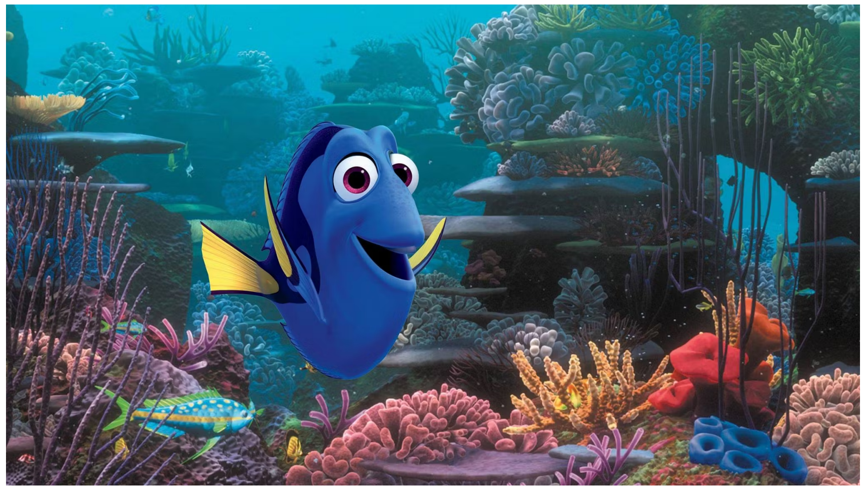
Universal

Mathematically compare algorithms

- **Downside:** Can often be too pessimistic, not predictive of performance on typical instances (in practice)

Uninformed vs Informed Optimization

- Worst-case algorithms cannot take advantage of domain-specific structure: start from scratch every time
- ML heuristics do a lot better taking advantage of correlations in input
    - Downside: little or no guarantees



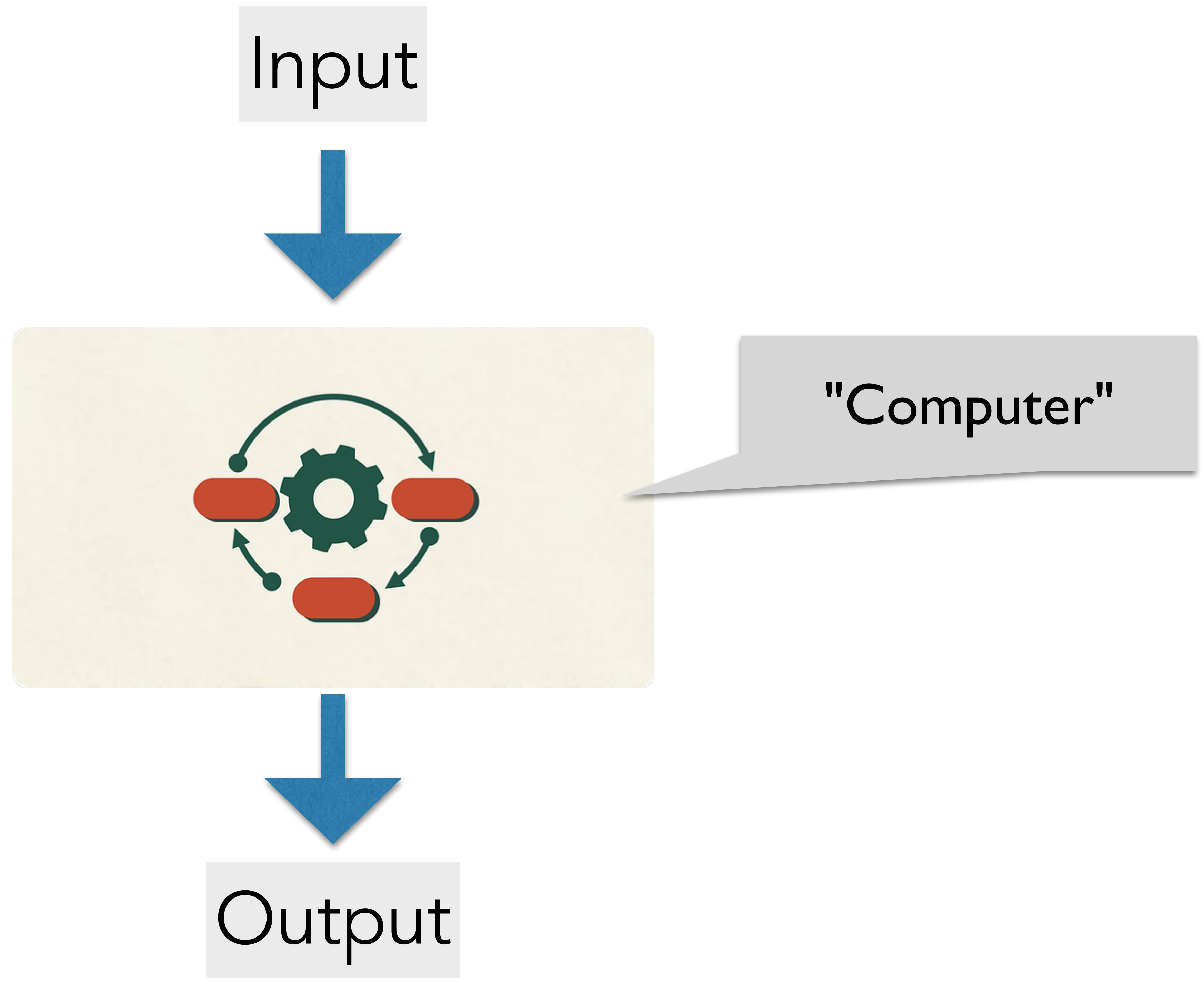
Algorithms with Predictions

- Combines ML predictions with worst-case analysis
- Best of both worlds:
    - Do well when predictions are good
    - Be prepared for the worst-case when predictions are bad

[Part 2]

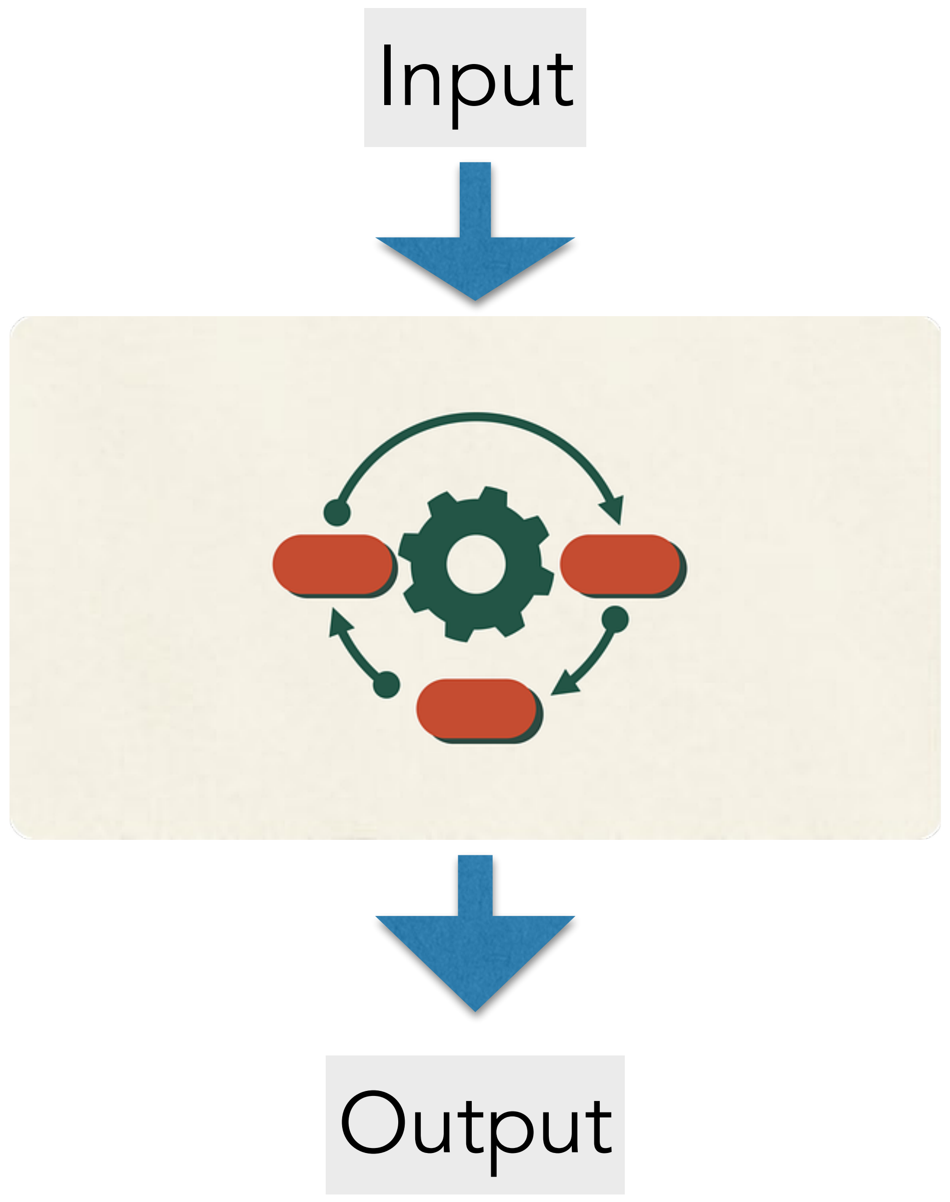
Course Wrap Up

What is Computation



Defining Computation

- **Computation**: manipulation of information/data to solve a problem
- **Computational problem**: the input/output pairs
- **Algorithm**: description of how this data is manipulated



Topics and Theme

What are the fundamental capabilities and limitations of computers?

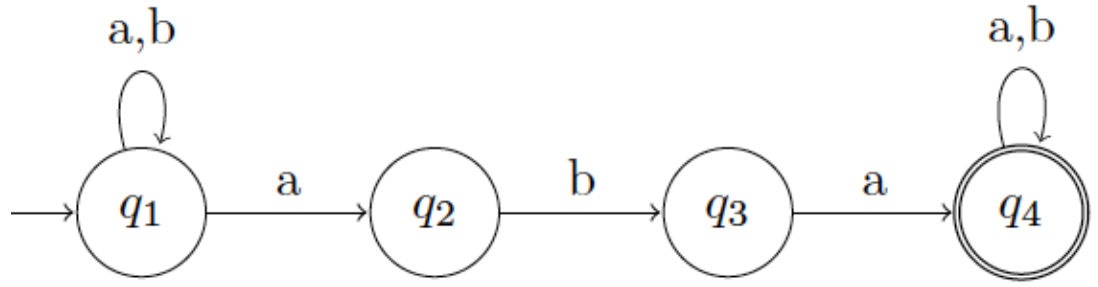
- **Automata Theory**
  - Finite automaton and Regular Languages
  - PDAs and CFLs
- **Computability Theory**:
  - Model a modern computer as a Turing machine
  - Identify what problems can and cannot be solved by it
- **Complexity Theory**:
  - Classify problems based on efficiency of solving it

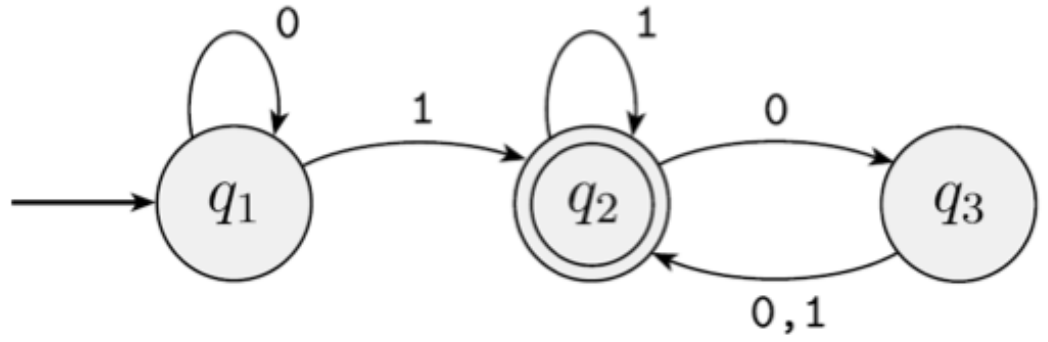
Regular Languages and DFA

- Equivalence of models:
  - DFA ⇔ NFA ⇔ Regular Expressions
- Closure properties
  - Closed under intersection, union, complement, concatenation, star, set difference, reverse, etc.

$$(ab \cup a)^*$$

- Minimal DFAs and Equivalence classes
- How to prove a language is not regular:
  - Pumping lemma
  - Myhill Nerode





Power and Limitations

- DFAs are good at simple repetitive or sequencing problems
- Do not have enough memory to count to an arbitrary number
- Examples of languages for which no DFA exists?
  - $\{0^n 1^n \mid n \geq 0\}$
  - $\{w \in \{0,1\}^* \mid$ number of 0s same as number of 1s$\}$
  - $\{ww \mid w \in \{0,1\}^*\}$
  - $\{ww^R \mid w \in \{0,1\}^*\}$
- To add more computation power, let's add some memory: a stack

Context-Free Languages

- Context-free grammars for generating CFLs
- Push-down automaton for recognizing CFLs
- Equivalence: (Non-deterministic) PDA $\Longleftrightarrow$ CFGs
- Closure properties:
    - Closed under union, Kleene star, reverse, concatenation
    - Not closed under intersection, complement
    - Intersection of a CFL and regular language is context-free
- How to prove a language is not context-free:
    - Pumping lemma: uses the fact that to generate arbitrarily long strings, must reuse a variable (recursion)

Power and Limitations

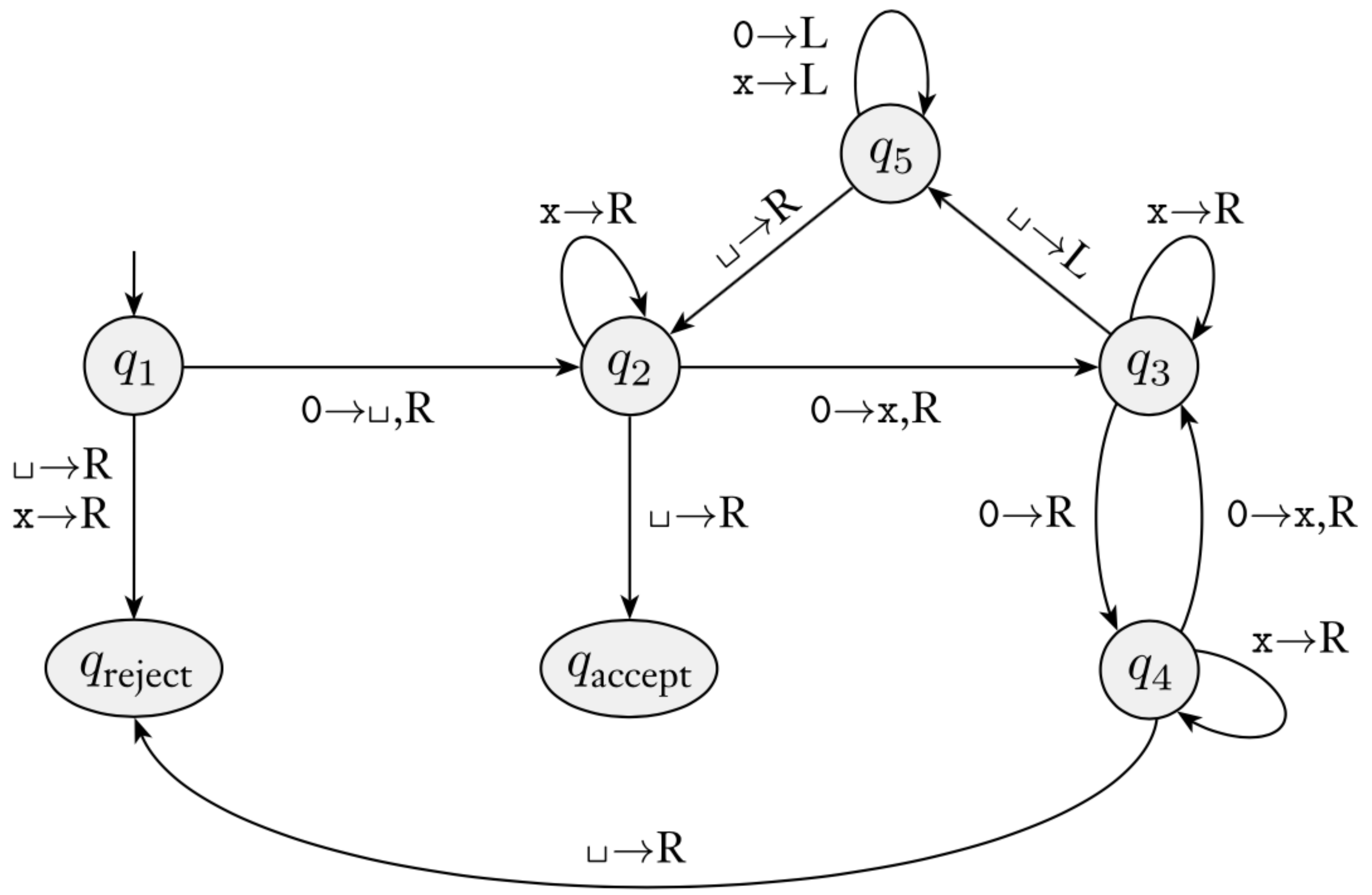
- Access to a stack (recursion), add considerable power over a DFA
- Examples of CFLs that are **not regular**?
  - $\{0^n 1^n \mid n \geq 0\}$
  - $\{w \in \{0,1\}^* \mid$ number of 0s same as number of 1s in $w\}$
  - $\{ww^R \mid w \in \{0,1\}^*\}$
  - $\{w \mid w \in \{0,1\}^*$ and $w = w^R\}$ (Palindromes)
  - $\{a^i b^j c^k \mid i, j, k \geq 0$ and $i = j$ or $j = k\}$
- Still has limitations: examples of languages that are not context-free?

Non-Context-Free Languages

- Pairing/Counting examples we have seen:
  - $\{a^n b^n c^n \mid n \geq 0\}$, $\{a^n b^n a^n\}$, $\{ww \mid w \in \{a, b\}^*\}$
  - HW: language of palindromes with equal # of 1s and 0s
  - Strings over $\{a, b, c\}$ with equal # of a's, b's and c's
  - $\{a^n b^m a^n b^m \mid n, m \geq 0\}$
  - $\{w\ a^n\ w^R\ b^n \mid w \in \{a, b\}^*, n \geq 0\}$
- Non-linear counting examples:
  - $\{a^{2^n} \mid n \geq 0\}$, $\{a^p \mid \text{p is a prime}\}$, $\{a^{n^2} \mid n \geq 0\}$
  - Intuition: structure is too rigid to be able to be "pumped"

Moving Up: Turing Machines

- A finite automaton with infinite memory
- **Question.** What did we learn about Turing machines and languages decided by TMs?



TM and TM Decidable Languages

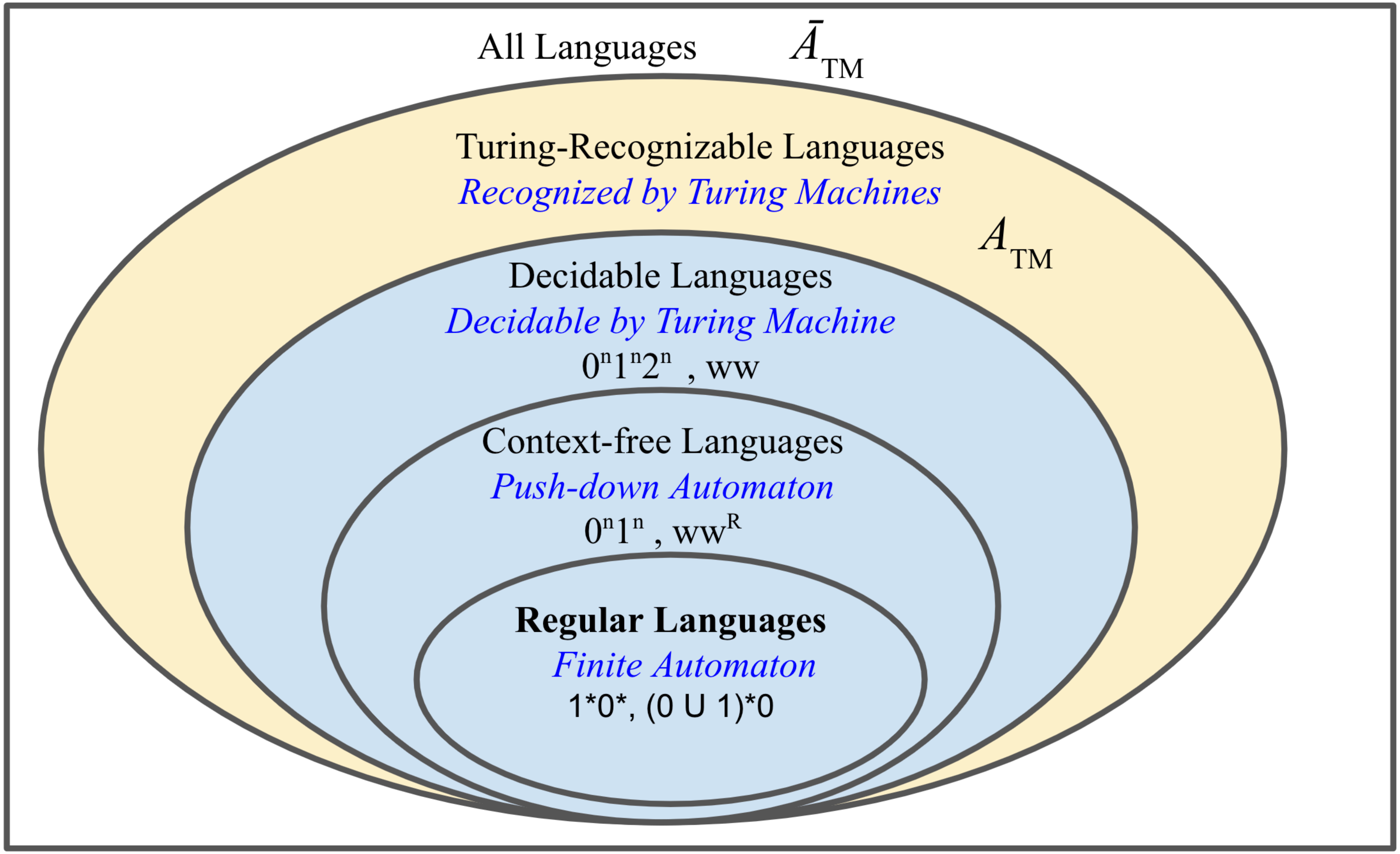
- Church-Turing Thesis
  - Anything that be computed by algorithms can be done on a TM
- Models of Turing machines:
  - Multi-tape, non-deterministic
- Properties of decidable languages:
  - Closed under union, intersection and complement
  - L is decidable iff L and $\overline{L}$ are TM recognizable
- Decidable languages about semantic properties of DFAs/CFGs?
  - A_{DFA}, A_{CFG}, E_{DFA}, E_{CFG}, EQ_{DFA}, etc.

Limits of Computation: Undecidability

- There are **infinitely many problems** that cannot be solved by any TM (counting argument)
- What do these problems look like?
  - Diagonalization to prove A_{TM} is undecidable
- Reductions to prove a bunch of other problems are undecidable
  - Halting problem, E_{TM}, EQ_{TM}, $\text{REGULAR}_{\text{TM}}$
  - Any non-trivial property of language of TM is undecidable
- Introduced mapping reductions (useful for TM recognizability)

Undecidability and Unrecognizability

- CFG related problems that are undecidable?
    - EQ_{CFG}, $\cap_{CFG}$
    - Proved using reduction from PCP
- How did we find Turing unrecognizable problems?
    - If an undecidable language is Turing recognizable, its complement must be **not Turing recognizable**
- Is there a language that is neither Turing recognizable not TM co-recognizable?
    - EQ_{TM}


All Languages
$\bar{A}_{TM}$
Turing-Recognizable Languages
Recognized by Turing Machines
A_{TM}
Decidable Languages
Decidable by Turing Machine
$0^n1^n2^n$, ww
Context-free Languages
Push-down Automaton
0^n1^n, ww^R
Regular Languages
Finite Automaton
1*0*, (0 U 1)*0

Complexity Theory

- **Question.** What did we learn about the different complexity classes?

Complexity Theory

- Time complexity classes: **P**, **NP**, **NP**-complete, **NP**-hard, **EXPTIME**
- **(Cook-Levin Theorem)** SAT is NP complete.
- Implications of P vs NP
- Reductions to prove other problems are NP complete:
    - Vertex Cover
    - Clique
    - 3Color
    - Hamiltonian Cycle, etc
- Final picture: $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \mathrm{EXPTIME}.$

We know **P $\neq$ EXPTIME**, so one of these containments is proper but we don't know which one

Takeaways

- Computational laws:
  - The simpler the model, the easier it is to verify its properties
  - The more power we add, the less we can verify
  - Computers can't do everything: many problems are not solvable
  - Many problems are not solvable in a reasonable amount of time
- TCS is a young field: there is a lot we still don't know

Thank you!

- You all should be proud of how much you've learned
- Good luck on the final exam and have great winter break!

Course Evaluations

Course Evals Logistics

- Two parts: **(1) SCS form** , **(2) Blue sheets** (both on GLOW)
- Your responses are **confidential** and we will only receive a report of your anonymized comments after we have submitted all grades for this course
- **SCS forms** are used for tenure/promotion & seen by CAP etc, **blue sheets are open-ended** comments directed only to your instructor

*To access the online evaluations, log into **Glow** (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "**Course Evaluations**." Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can ask a neighbor for help or reach out to ir@williams.edu.*