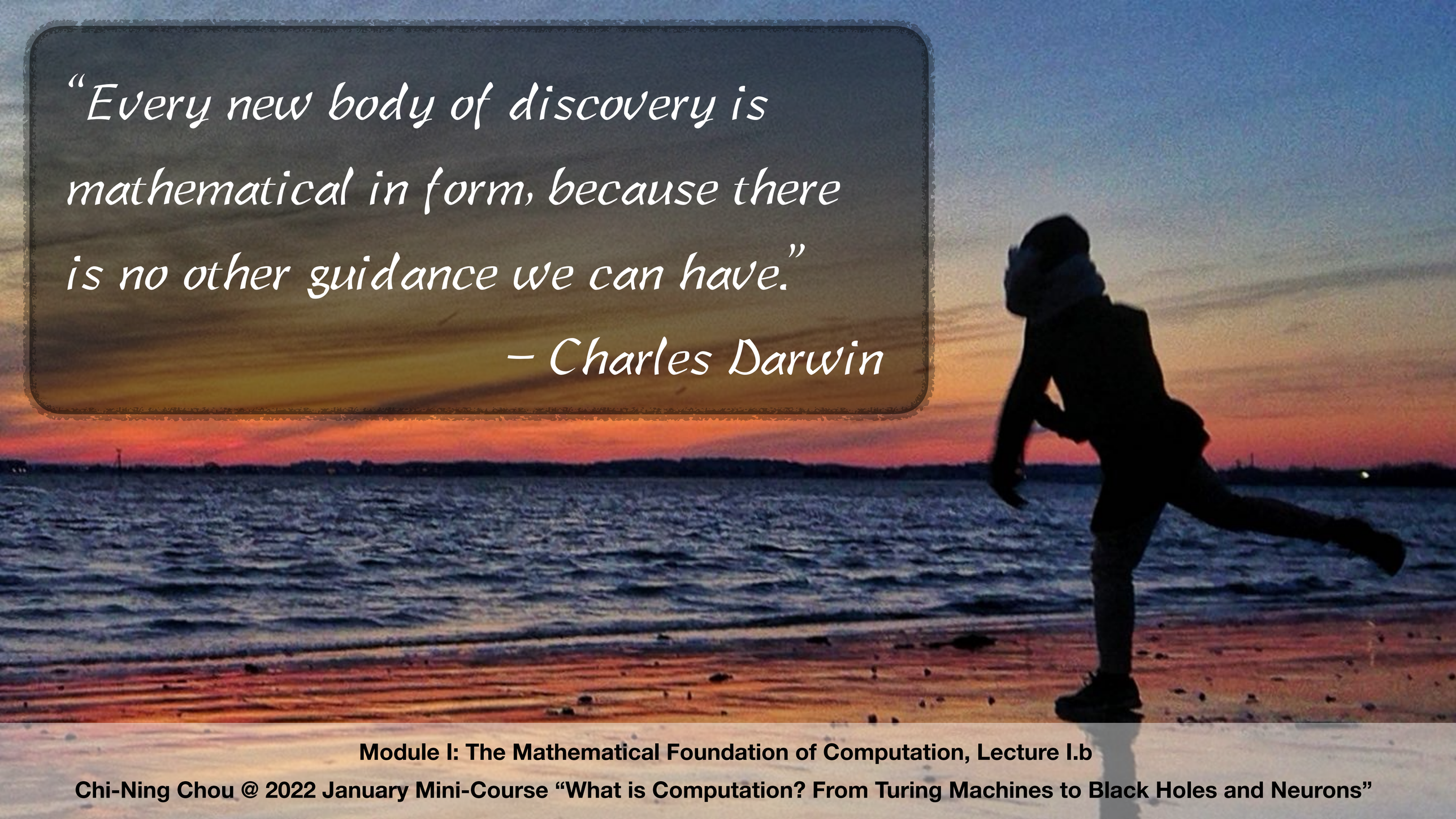


Module I: The Mathematical Foundation of Computation, Lecture I.b

Chi-Ning Chou @ 2022 January Mini-Course "What is Computation? From Turing Machines to Black Holes and Neurons"

Modern Developments: Models, Resources, Reductions

Module I: The Mathematical Foundation of Computation

"Every new body of discovery is mathematical in form, because there is no other guidance we can have."

– Charles Darwin

Last Lecture

- Mathematical definitions of computation.
- Turing machine.
- Gödel's incompleteness theorems.
- Uncomputability theorems.
- Church-Turing thesis.

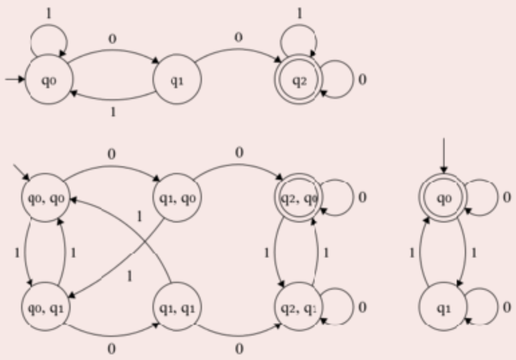
- Different computational models.
- Different computational resources.
- Efficiency and complexity.
- Reductions.
- Cook-Levin theorem.

This Lecture

Turing Machine is not Alone!

Different computational models reveal different aspects of computation

Weaker Models



Finite automata

[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}

Regular expression

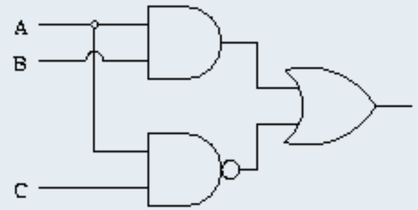
"Turing-Complete" Models

λ-calculus

Programming language

Video games

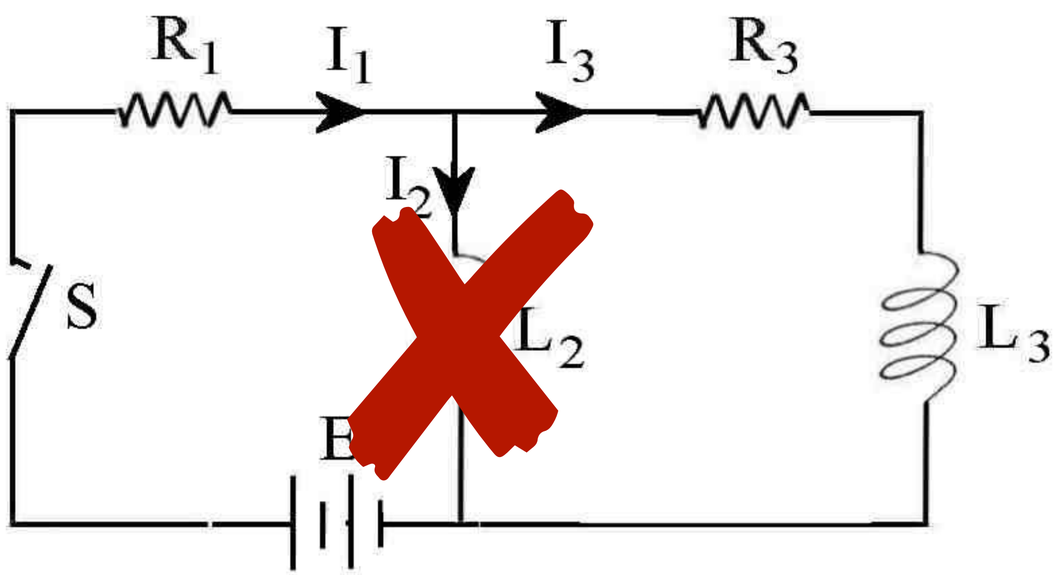
Other Models

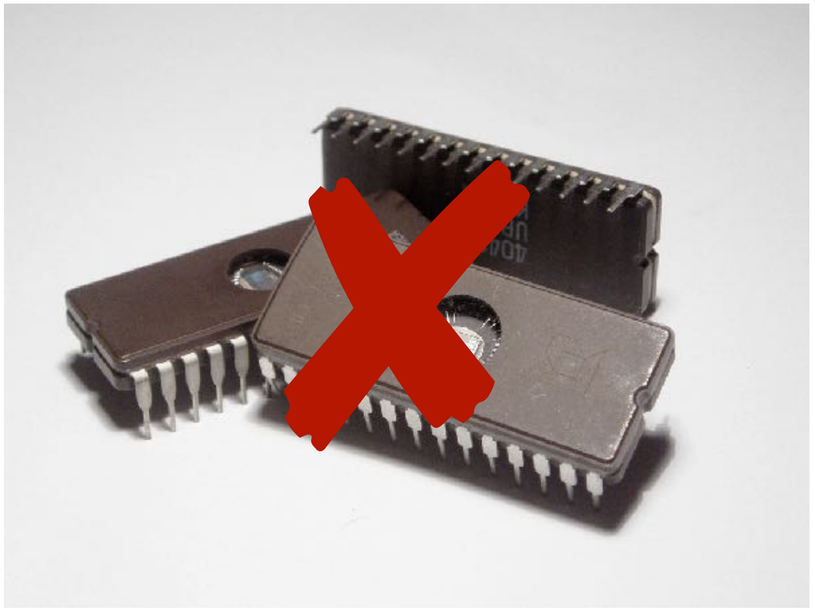


Circuit

Communication model

Example 1: Circuits

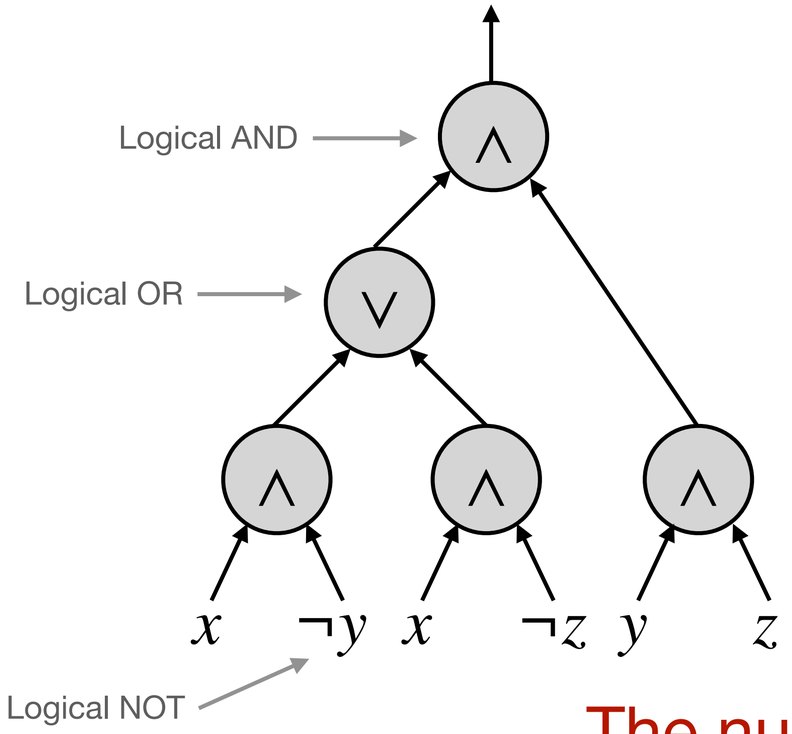




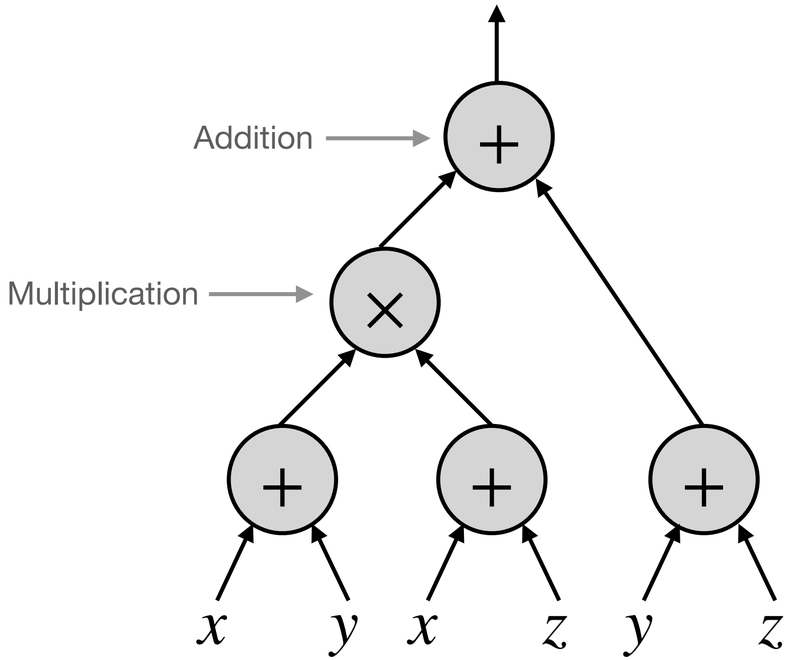
Example 1: Circuits

Circuits are computational models that can be described by “gates” and “wires”.

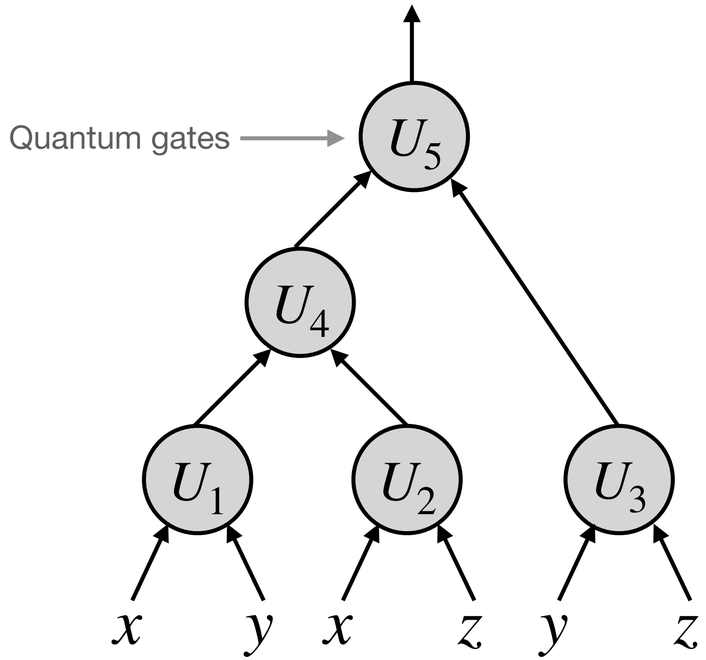
Boolean Circuits



Arithmetic Circuits



Quantum Circuits

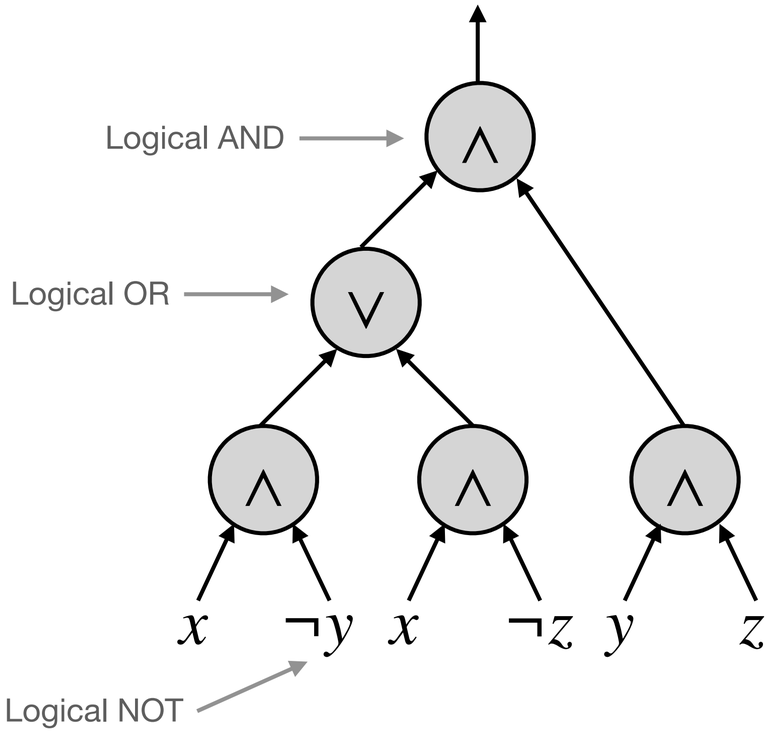


The number of inputs of a circuit is pre-fixed!

* More on quantum circuits in Lecture II.b.

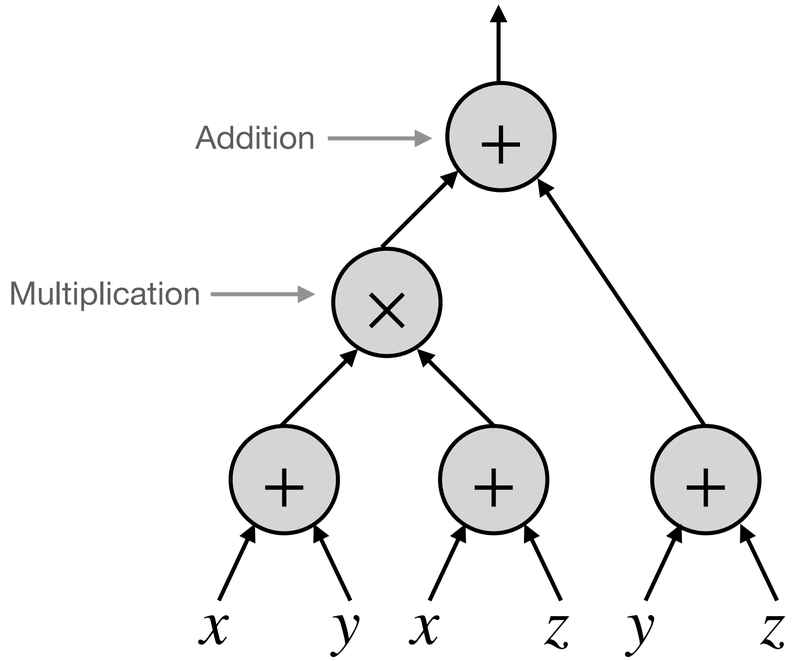
Example 1: Circuits

Boolean Circuits



$(x = 1,\ y = 1,\ z = 0)$

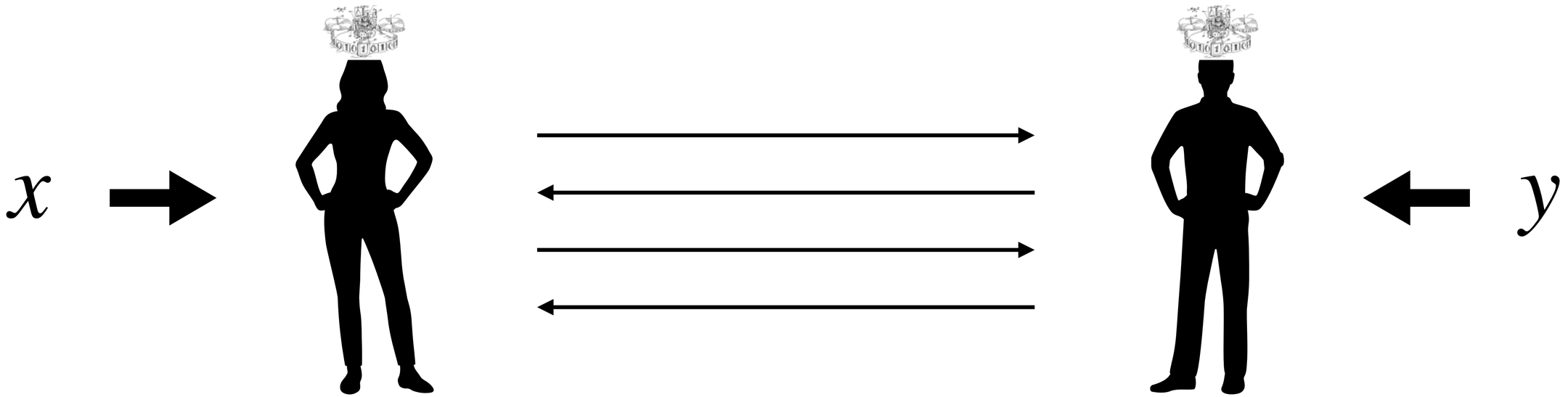
Arithmetic Circuits



$(x = 1,\ y = 2,\ z = 3)$

Example 2: Communication Models

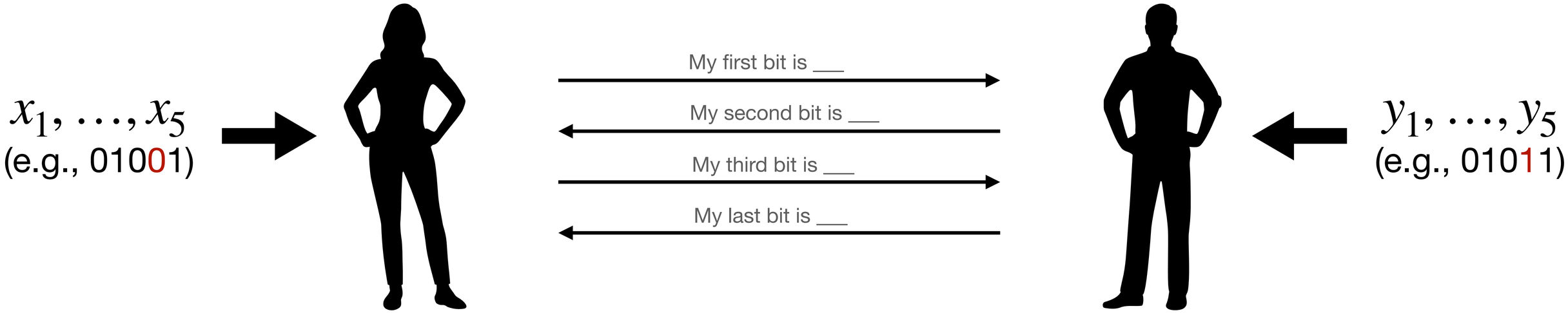
$$f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$



Goal: Compute $f(x, y)$ through communication!

Example 2: Communication Models

$$EQ_5(x_1 \ldots, x_5, y_1, \ldots, y_5) = \begin{cases} 1, & x_1 = y_1, x_2 = y_2, \ldots, x_5 = y_5 \\ 0, & \text{else.} \end{cases}$$



- The “communication protocol” is fixed before seeing the inputs.
- 5 bits of communication is needed!

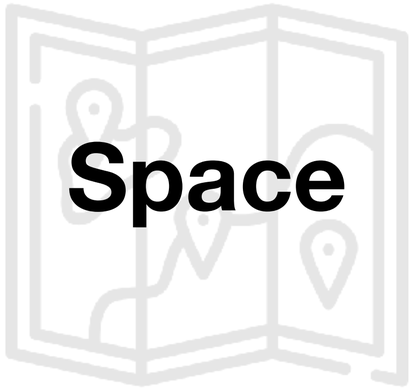
* When generalize to EQ_n, with “shared randomness”, $O(\log n)$ bits is sufficient!

* Another example in later slides and more mathematical details in references.

Different Computational Resources

Toward a better understanding of the essence of computation

Time

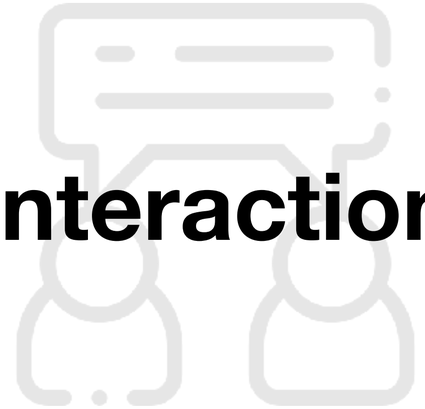


Space

Nondeterminism

Randomness

Quantumness



Interaction

In particular, all these resources can be *quantified*!

Example 1: Nondeterminism

The existence of creativity and free will!?

Q: The original definition of Turing machine follows deterministic rules, how to explain the "*Aha moment*" in our experience?

Solving math problems

Creating art

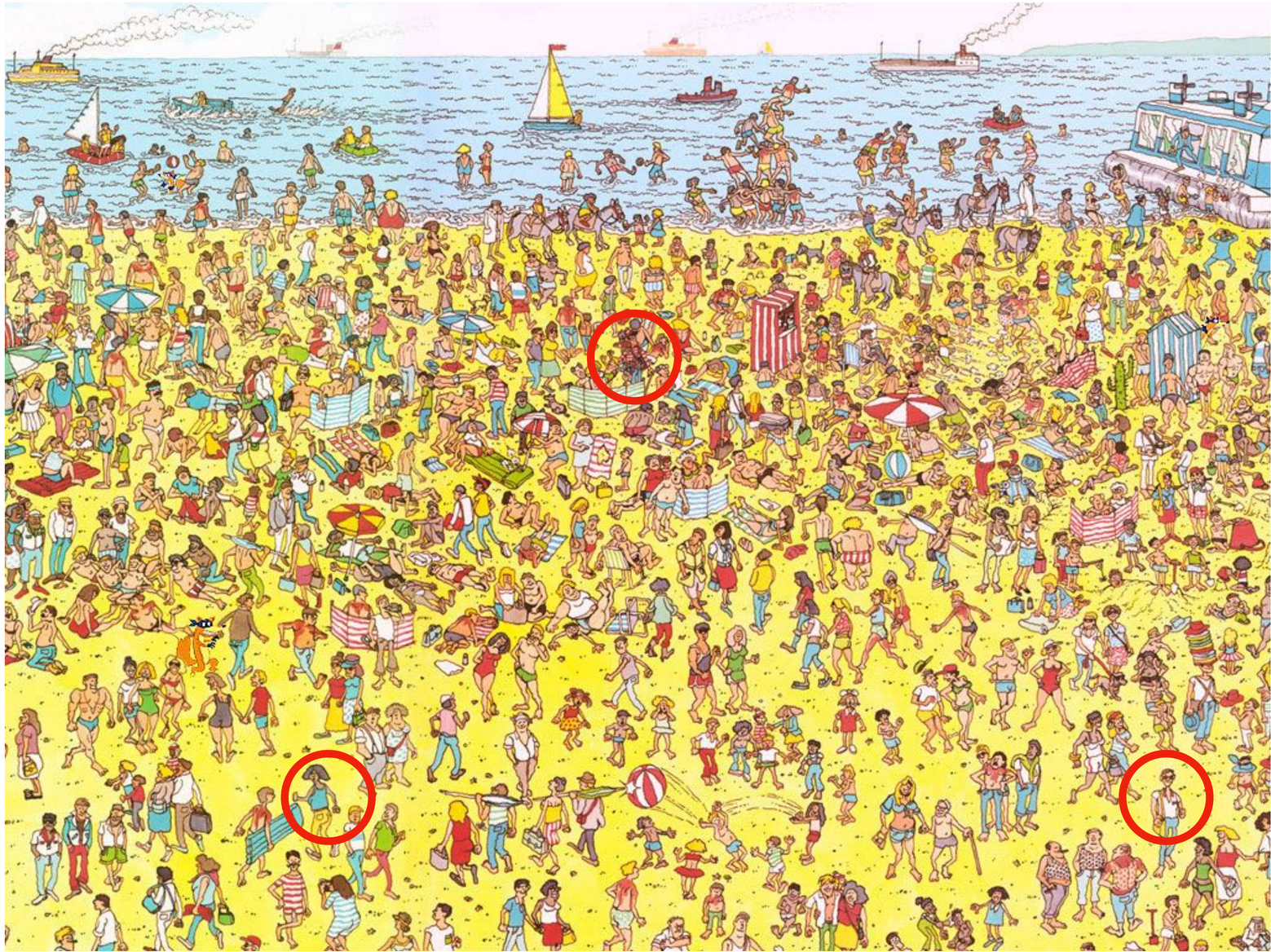
Coming up with new ideas

Nondeterminism *Mathematically modeled as* **Verifiable Guesses**

* More mathematical details in references, more on creativity in guest talk III.b and III.c, more on free will in lecture II.c.

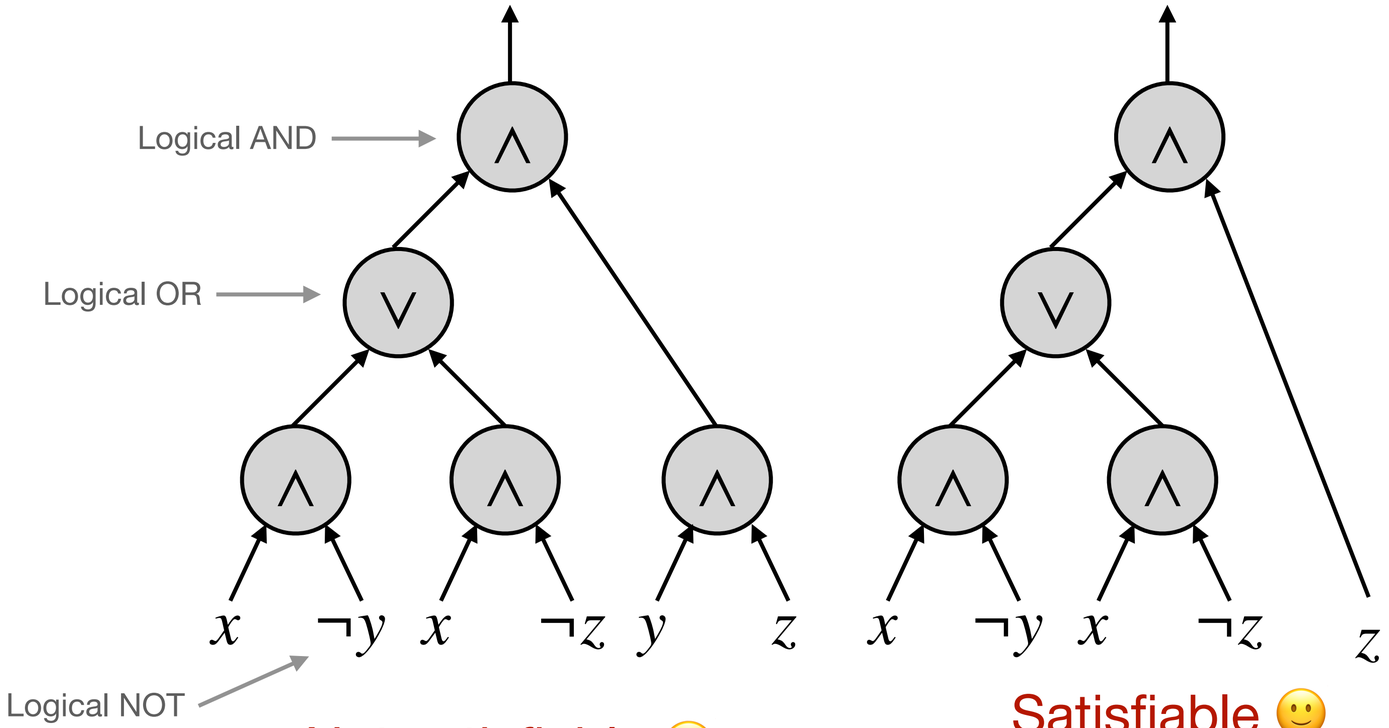
Nondeterminism → Verifiable Guesses

Where’s Wally?



Boolean Satisfiability Problem

Circuit-SAT



Not satisfiable 😔

Satisfiable 🙂

(e.g., $x = z = 1, y = 0$)

3SAT

Logical OR Logical AND Logical NOT

$$(x \vee \neg y \vee z) \wedge (\neg x \vee z) \wedge (x \vee y \vee \neg z)$$

Satisfiable 🙂

(e.g., $x = y = z = 1$)

$$(x \vee \neg z) \wedge (\neg x \vee z) \wedge (\neg x \vee \neg z) \wedge (x \vee z)$$

Not satisfiable 😔

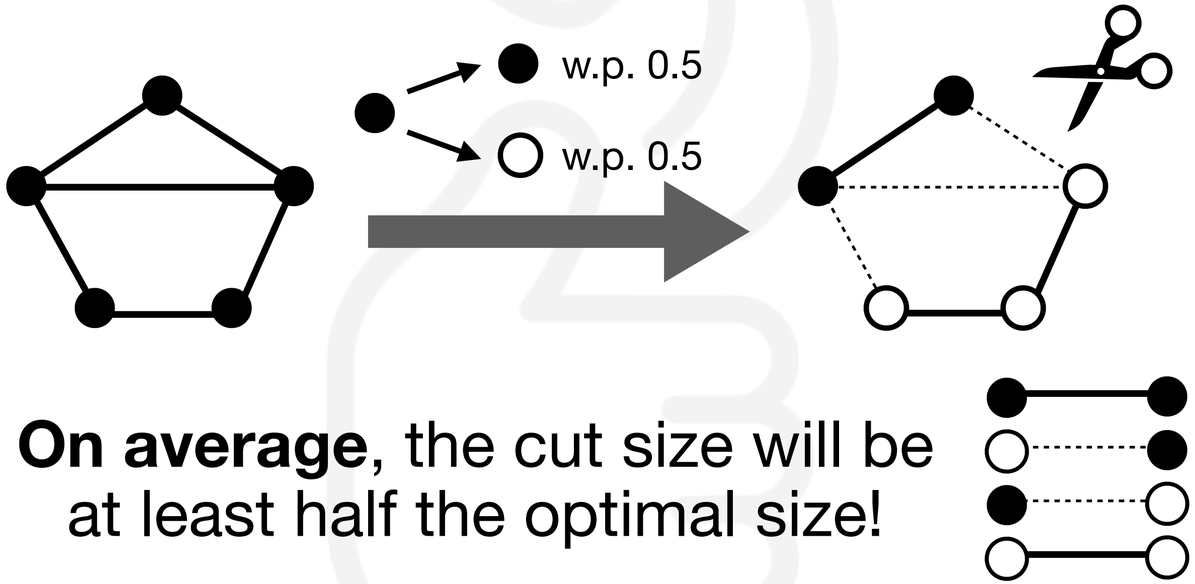
If the input is satisfiable ⇒ there's a simple way to verify!

* Note that for non-satisfiable input, it might not be easy to verify its unsatisfiability!

Example 2: Randomness & Quantumness

How does stochasticity play a role in computation?

Example: A simple randomized algorithm for Max-CUT



On average, the cut size will be at least half the optimal size!

Q: Can you remove the randomness?

Example: Shor's quantum factoring algorithm

124307

Some "quantum" operations

197×631

Q: Can you remove the quantumness?

* More details on quantum in Lectures II.b and Advanced section II.c, II.d.

Modern Study of Computation: Computability → Complexity

Efficiency and Complexity

All resources and models can still be *simulated* by a deterministic Turing machine!

But… the simulation could be inefficient!

Computational Complexity Conference (CCC)

- Take computational efficiency into account.
- The amount of resources needed ⇒ the **complexity** of a computational problem.

Q: How to mathematically model complexity?

Formulation of Computational Complexity

With a focus on time complexity

Example: n-digit Multiplication

- Algorithm design matters. ⇒ Best possible algorithm

  Schoolbook multiplication | Harvey-Hoeven algorithm

- Hardware/Software implementation matters. ⇒ Focus on “asymptotic”



$O(n \log n)$

- The difficulty of input matters. ⇒ Worst-case scenario

  $1010101010 \times 999999999$ | $6384910743 \times 1749380901$

* More mathematical details in advanced sections and philosophical discussion in lecture I.c and guest talk I.

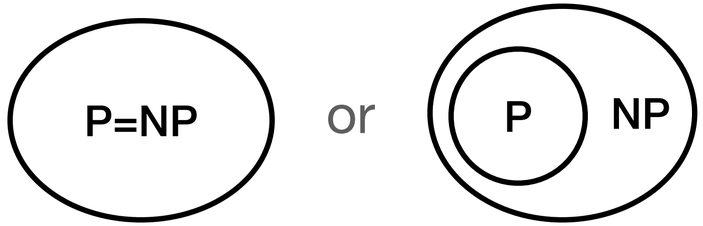
Complexity Classes

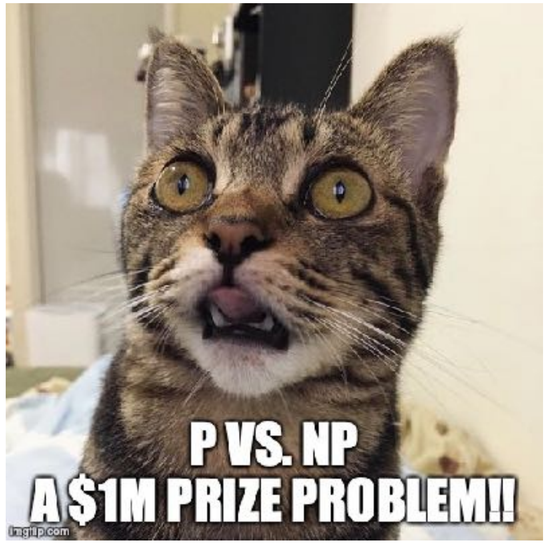
Informal Definition (Complexity class).

A complexity class is a collection of computational problems that cost similar computational resources.

Example:

- P = problems that don't cost too much time.
- NP = P + nondeterminism.
- BPP = P + randomness.
- BQP = P + quantumness.
- PSPACE = problems that don't cost too much space.



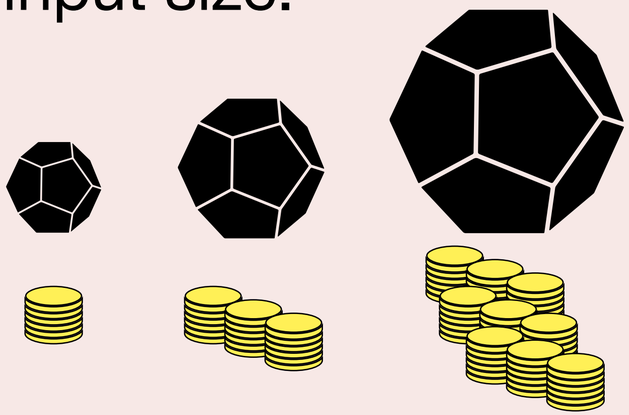


“Cost Similarly” & “Not Cost too Much”

Three concepts that save the days for complexity theorists

Asymptotic

Study how does the cost scale with the input size.



Big O notation

Ignore minor factor & constants not scaled with the input size.

Example:

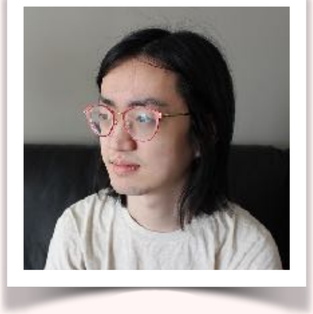
- $2n^2 + 100 = O(n^2)$.
- $\frac{2^n}{100} + n^{100} = O(2^n)$.

Polynomial time

Efficient = the scaling is polynomial in the input size.

Example:

- $2n^{1000} + 100$ is efficient.
- $\frac{2^n}{10000} + n$ is inefficient.



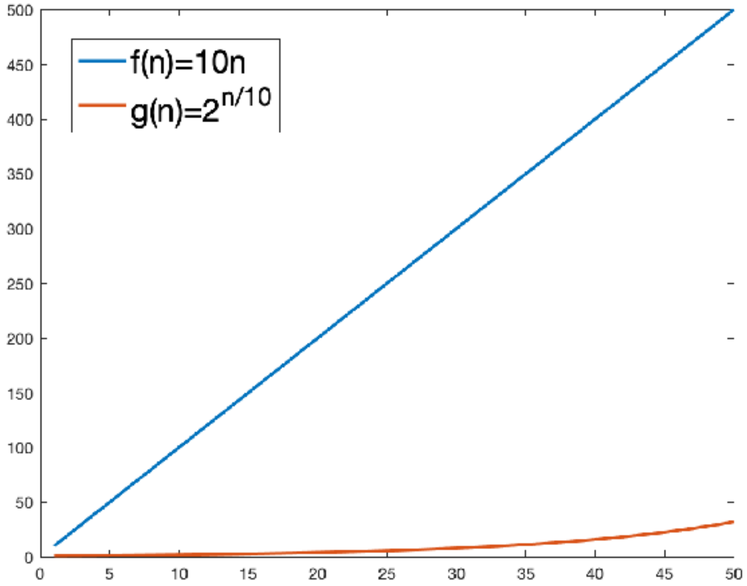
Lijie Chen
(Jan. 11 11am-12pm ET)

"Efficiency of Algorithms and the Sorites Paradox"

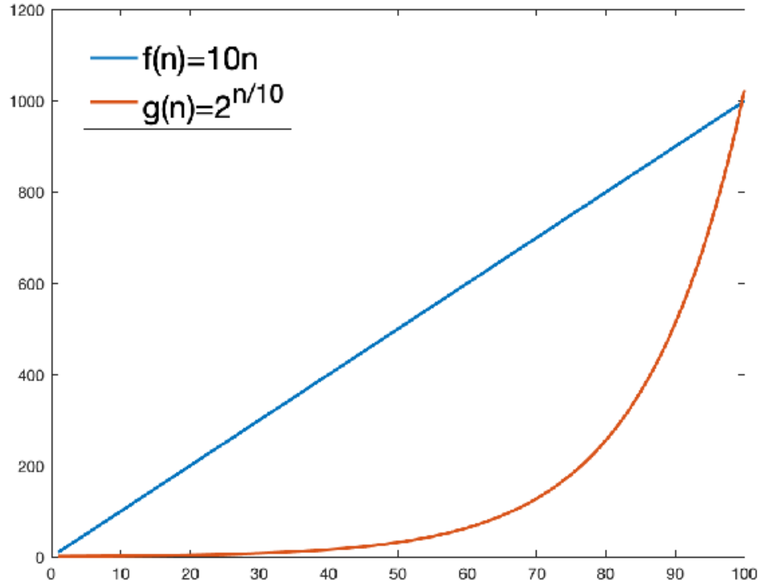
Example

$$f(n) = 10n$$

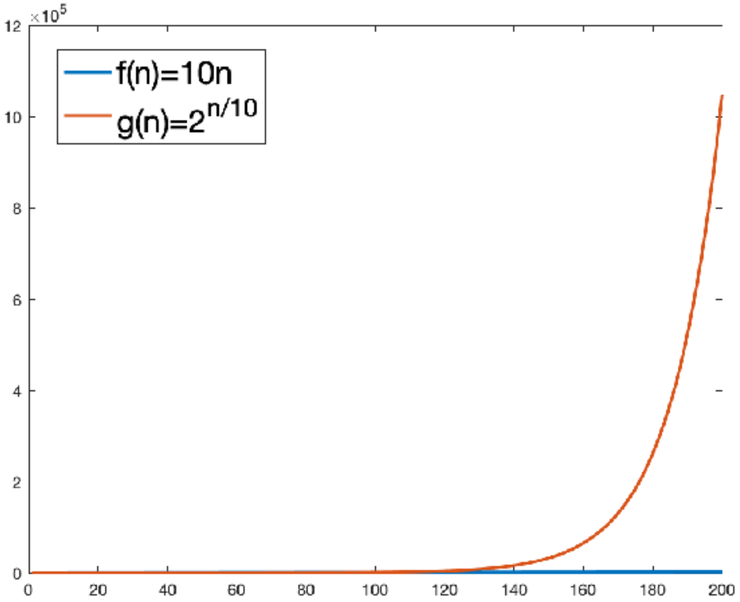
$$g(n) = 2^{n/10}$$



$n = 1,\ldots,50$

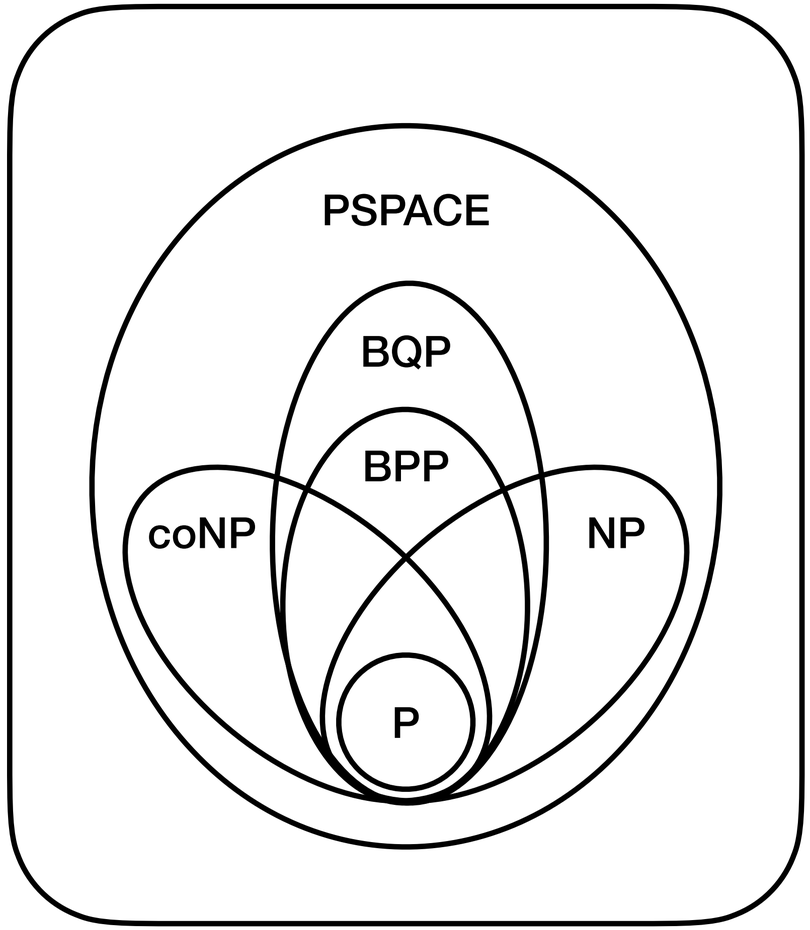


$n = 1,\ldots,100$



$n = 1,\ldots,200$

Questions Asked in Complexity Theory



Separation: showing two complexity classes are not the same.

Simulation: showing one complexity class contains the other.

Examples:

- P vs. NP: whether nondeterminism helps?
- P vs. BPP: whether randomness helps?
- P vs. BQP: whether quantumness helps?

The Gem of Theoretical CS

Reductions

The main sword in the theory of computation

Informal Definition (Reduction).

We say problem A can be reduced to problem B, denoted as

$$A \leq B$$

if one can “efficiently” compute A with the help of B.

A	B
Multiplication	Addition
Feasible computation	Turing machine
Kid	Parents

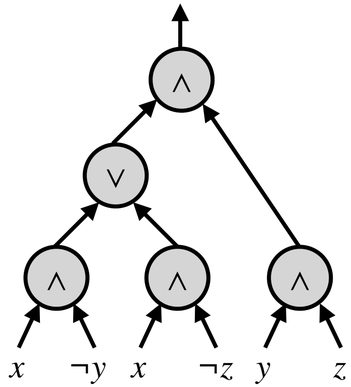
* The last example is just for illustration, I’m not aware of any mathematical proof for it and I doubt it’s correctness.

Various Ways to Think About Reductions

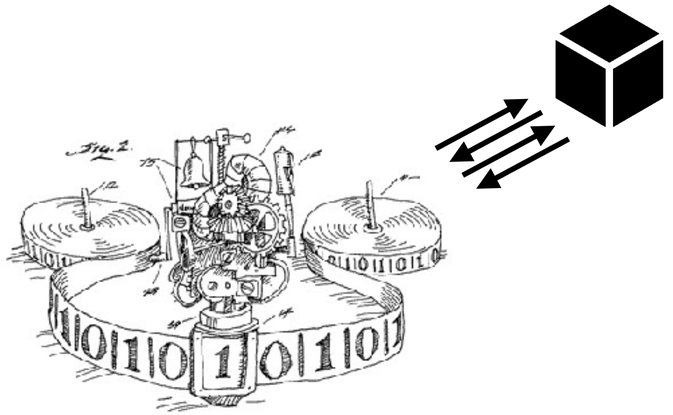
Blackbox/Oracle

Subroutine/Macro

$$A \leq B$$



$$(x \vee \neg y \vee z) \wedge (\neg x \vee z) \wedge (x \vee y \vee \neg z)$$



Karp Reduction

Turing Reduction

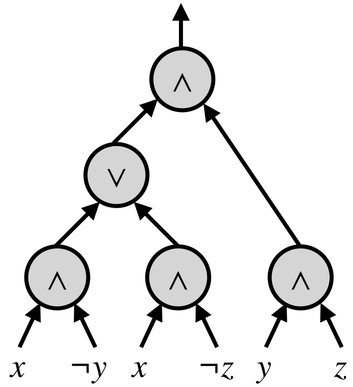
Example 1: Cook-Levin Theorem

Identify the most difficult problem in a complexity class!

Cook-Levin Theorem

Every problem in NP can be reduced to 3SAT in polynomial time. Namely, 3SAT is NP-hard.

Nondeterminism
=
Verifiable guesses



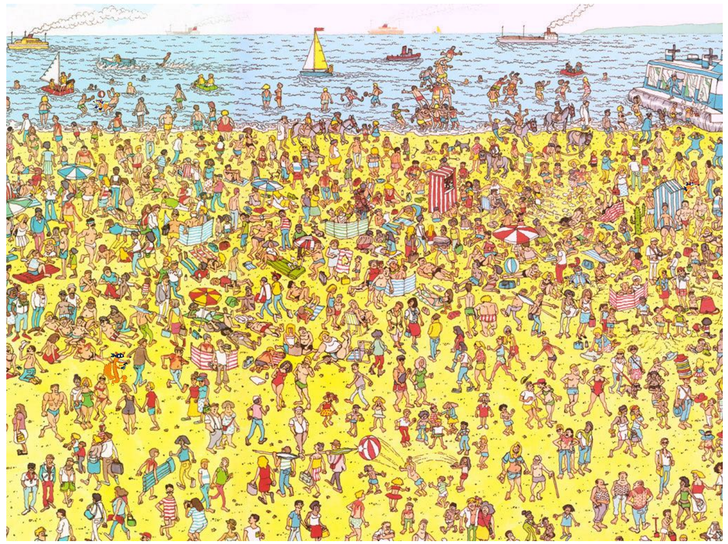
$$(x \vee \neg y \vee z) \wedge (\neg x \vee z) \wedge (x \vee y \vee \neg z)$$

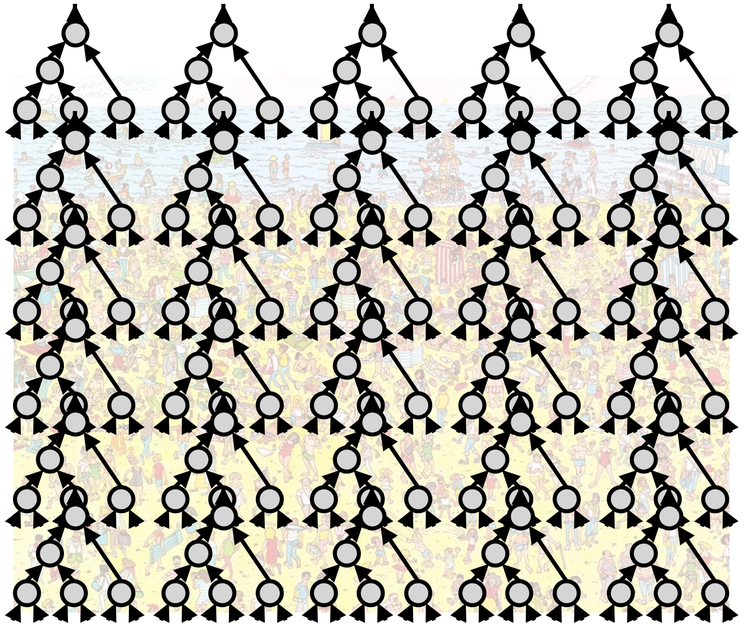
NP = problems that can be efficiently verified.

Use Circuit-SAT to efficiently verify

Circuit-SAT $\leq$ 3SAT

Apply Cook-Levin Theorem on Wally!





$$(x \vee \neg y \vee z) \wedge (\neg x \vee z) \wedge \cdots \wedge (x \vee y \vee \neg z)$$

Where's Wally?

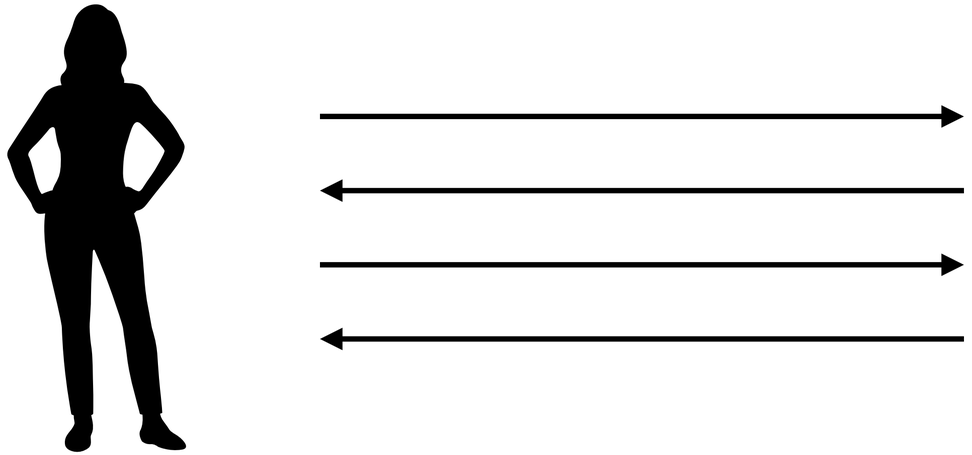
Use Circuit-SAT to efficiently verify

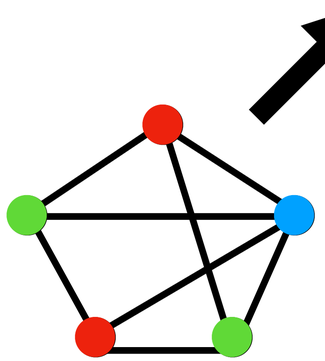
Circuit-SAT $\leq$ 3SAT

More on Cook-Levin Theorem in Lecture I.c and II.c!

Example 2: Communication Complexity

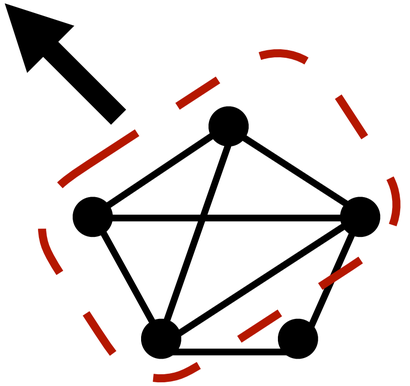
The Karchmer-Wigderson Games (KW games)





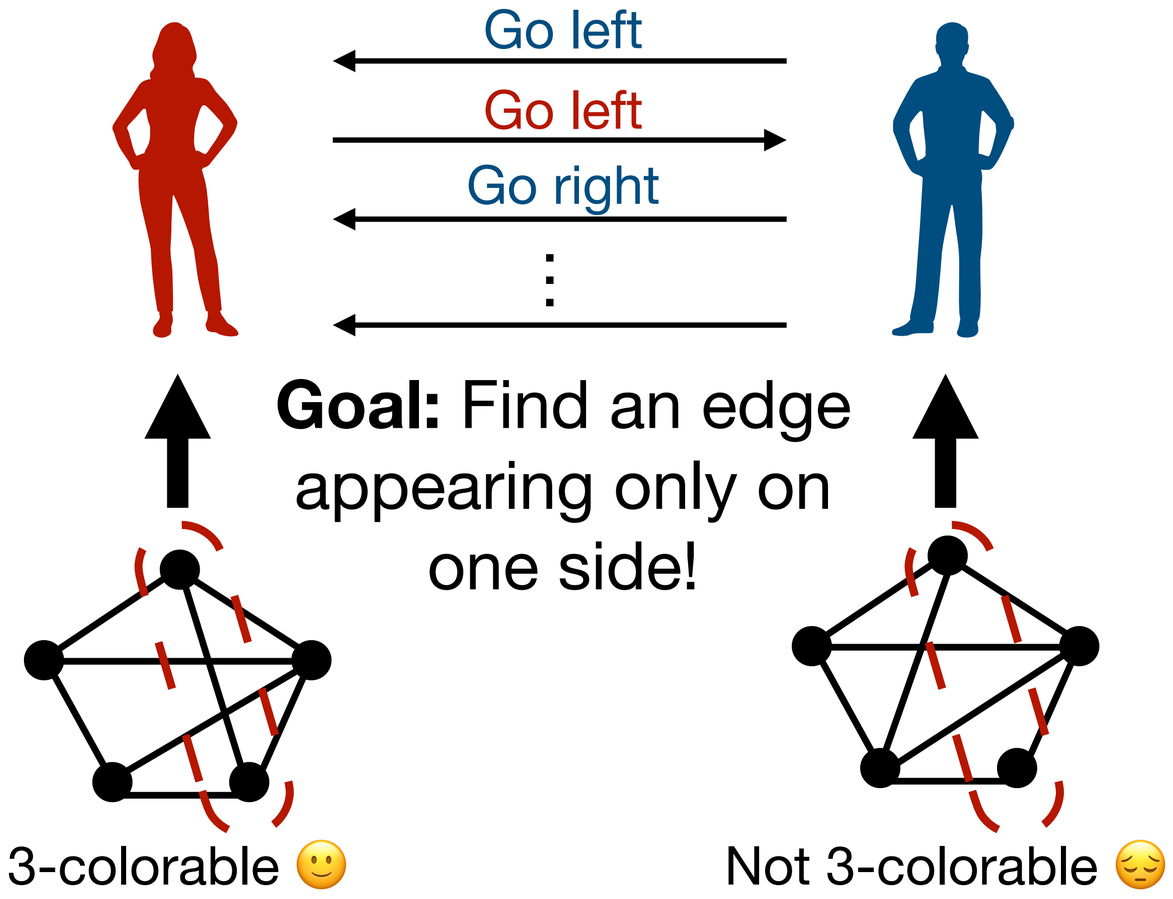
3-colorable 🙂

Goal: Use the least amount of communication to find an edge appearing on only one side!

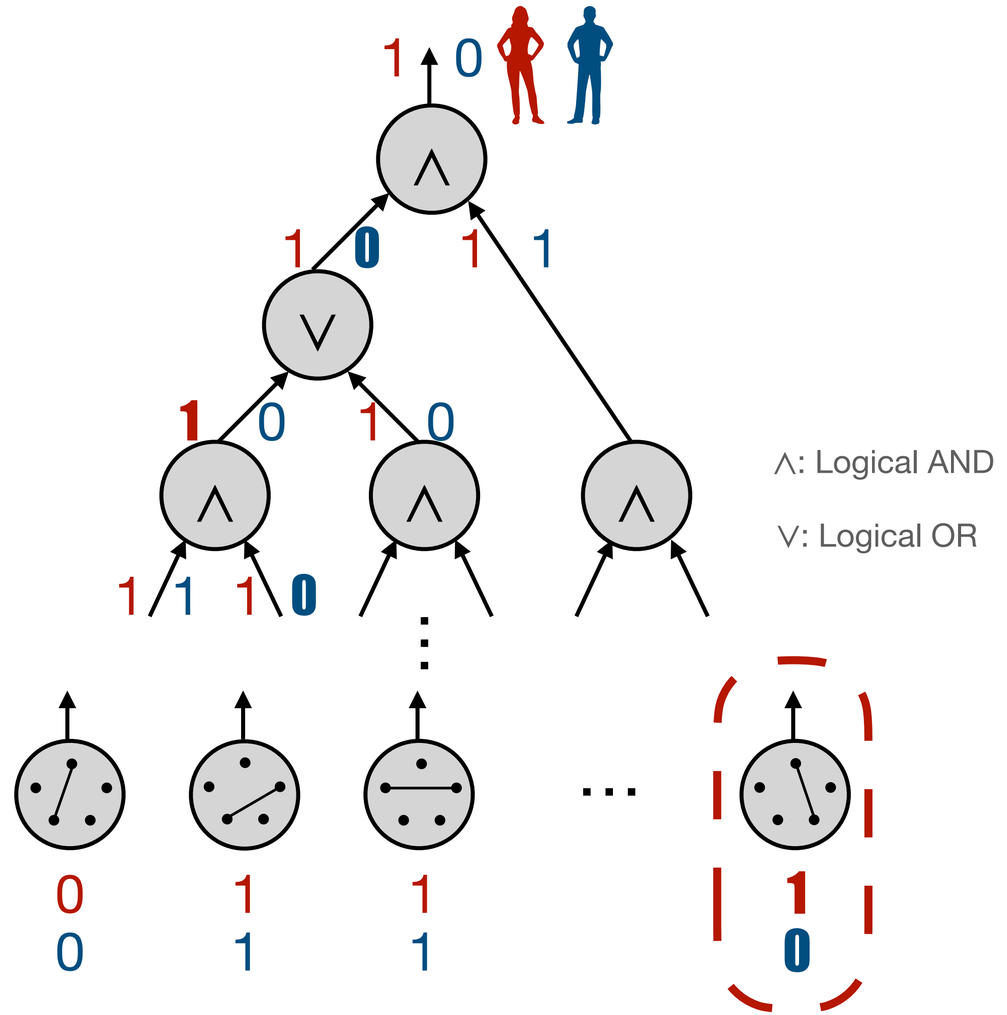


Not 3-colorable 😔

KW Games $\leq$ Boolean Circuits

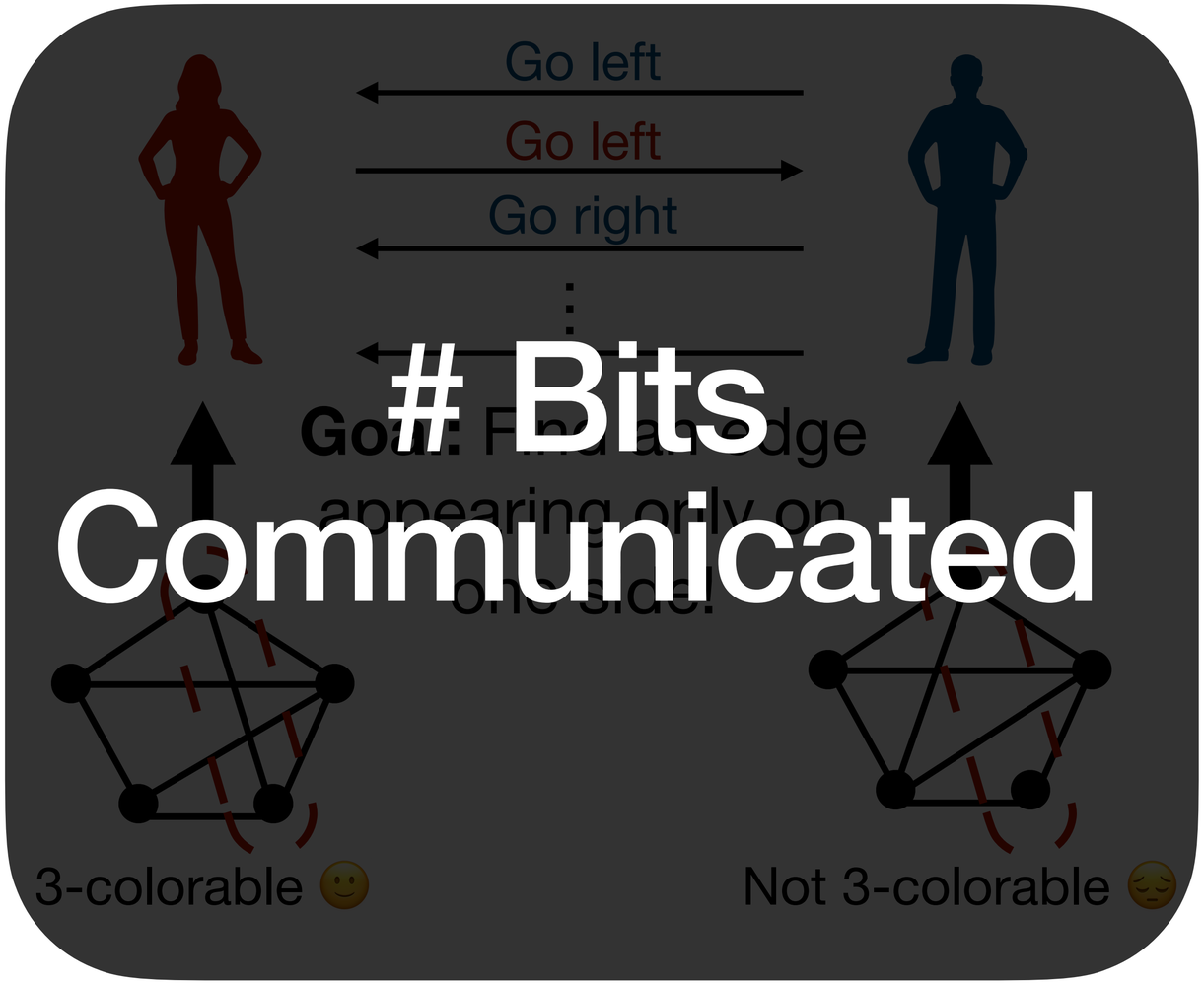


A KW game for 3-coloring



A Boolean circuit for 3-coloring

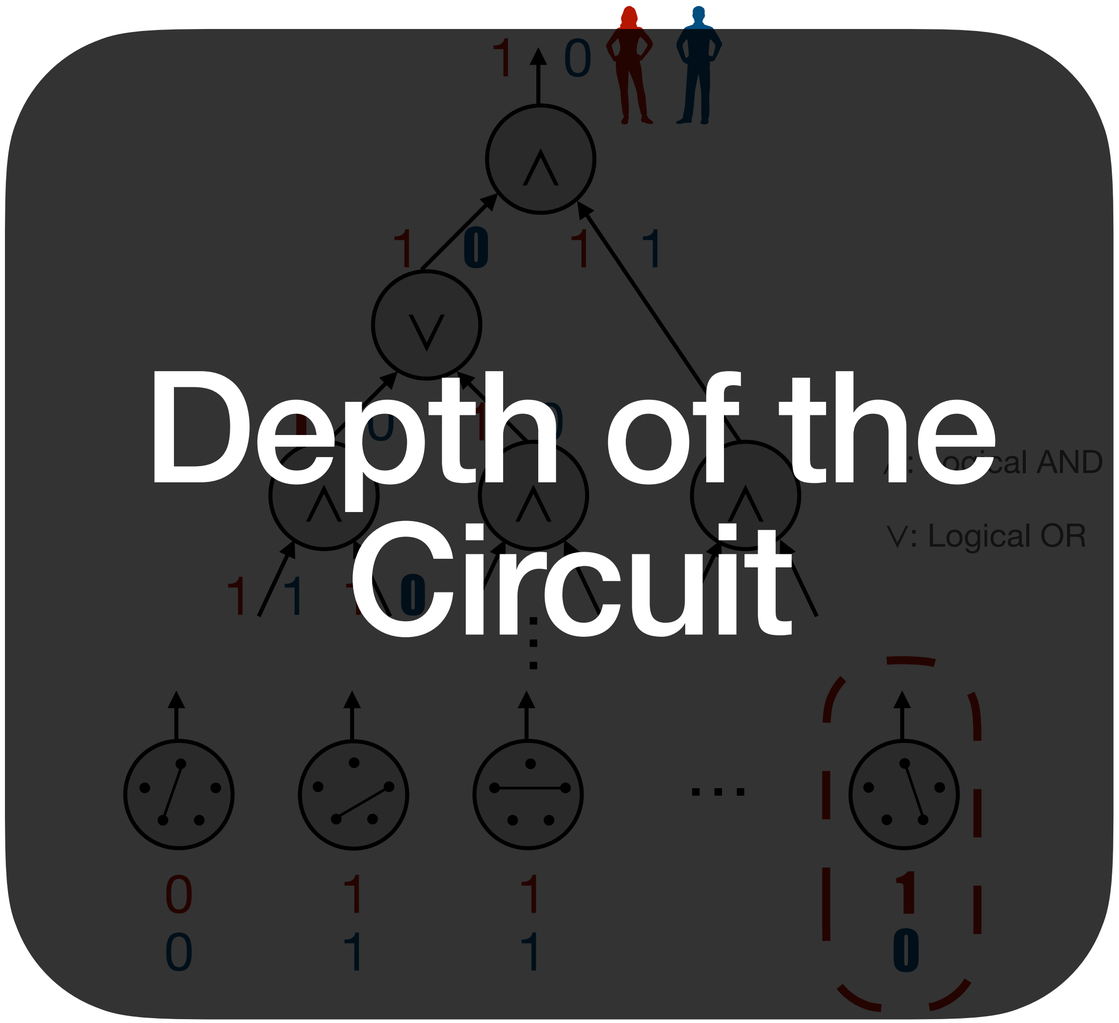
KW Games $\leq$ Boolean Circuits



Bits Communicated

$\geq$

Depth of the Circuit



A KW game for 3-coloring

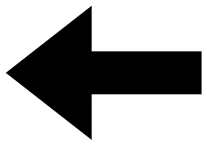
A Boolean circuit for 3-coloring

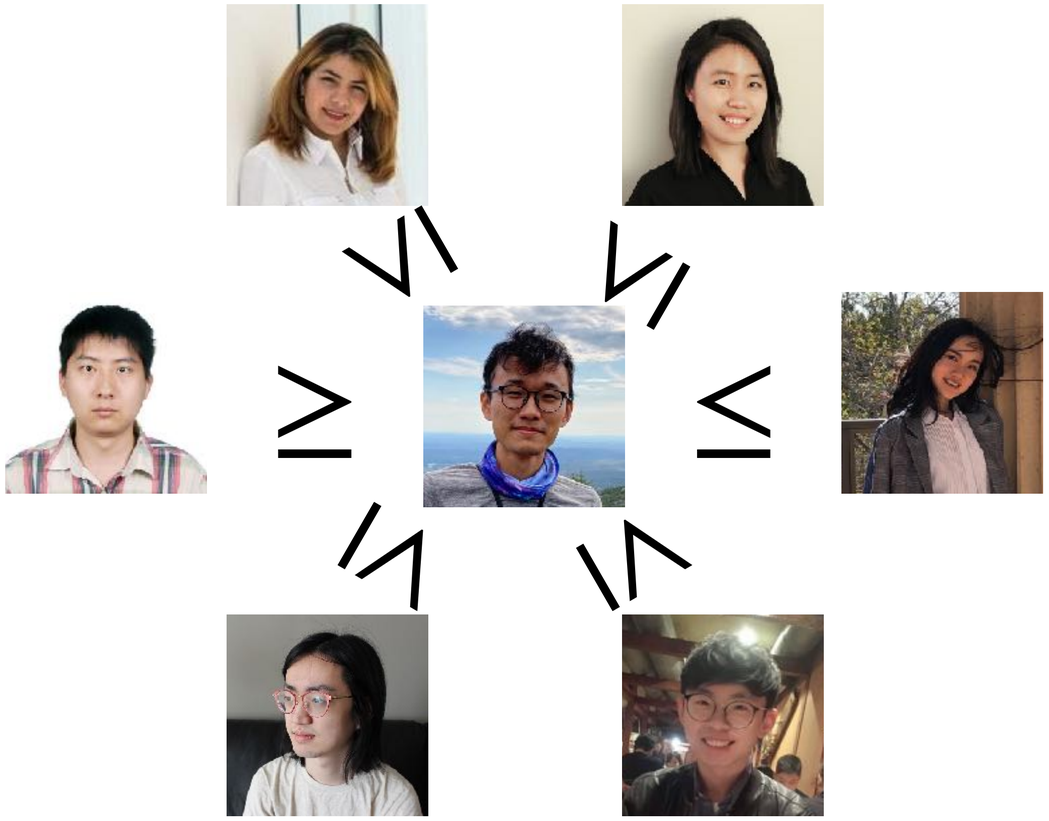
Reductions are Ubiquitous!

968105639
(Factoring problem)

$\leq$

13147*73637





Cryptography

Real Life!

Next

Lijie Chen
(Jan. 11 11am-12pm ET)

“Efficiency of Algorithms and the Sorites Paradox”

Abstract: Why do theoretical computer scientists think of an algorithm with running time $100000n$ being efficient while another with running time $2^{n/10000}$ being inefficient? In this talk, I'm going to introduce you to a philosophical foundation of this central formalism in theoretical CS. I will guide you to rethink about the definition of "efficient algorithm" through the lens of Sorites paradox. Furthermore, we will touch on connections to Moore's law, cosmology, and beyond. Hope that after the talk, you won't feel that you are old when you wake up tomorrow! (if you didn't get this joke, you will after attending my talk!)

More!

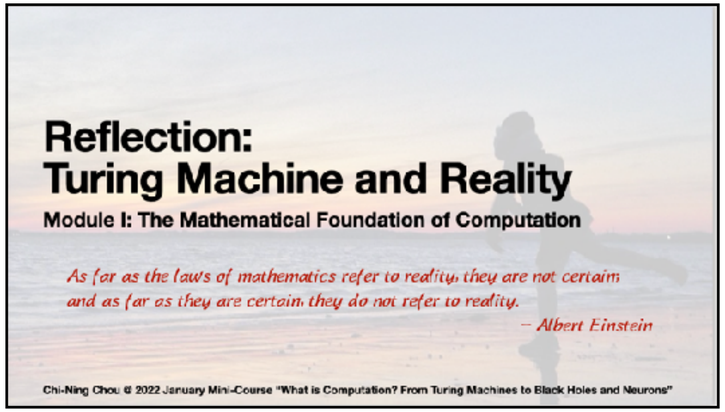
Reijo
(Jan. 11
2pm-3pm ET)

"*Undecidability of the Halting Problem and Gödel's incompleteness Theorem*"

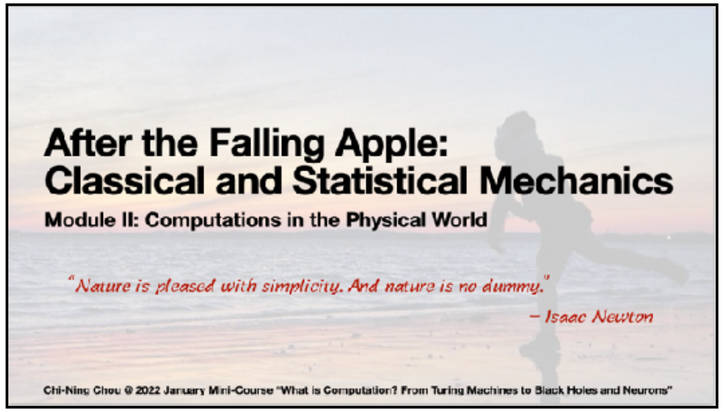
Prahlad
(Jan. 12
9am-10am ET)

"The Four Color Theorem"

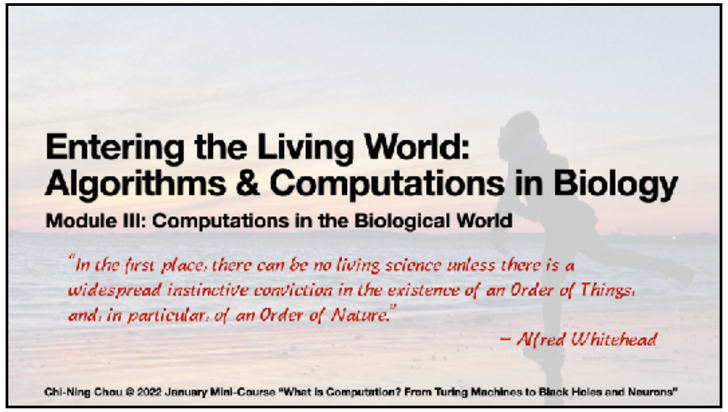
See the abstract for their talks on the course website!



Lecture I.c
(Jan. 17 10am-10:50am ET)



Lecture II.a
(Jan. 12 10am-10:50am ET)



Lecture III.a
(Jan. 12 11am-11:50am ET)

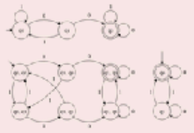
Summary

Key Concepts

Turing Machine is not Alone!

Different computational models reveal different aspects of computation

Weaker Models



Finite automata

[\w._%+-]+@[\w.-]+\.[a-zA-Z]{2,4}

Regular expression

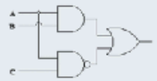
"Turing-Complete" Models

λ-calculus

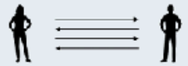
Programming language

Video games

Other Models



Circuit



Communication model

Different Computational Resources

Toward a better understanding of the essence of computation

Time

Space

Nondeterminism

Randomness

Quantumness

Interaction

Complexity Classes

Informal Definition (Complexity class).

A complexity class is a collection of computational problems that cost similar computational resources.

Example:

- P = problems that don't cost too much time.
- NP = P + nondeterminism.
- BPP = P + randomness.
- BQP = P + quantumness.
- PSPACE = problems that don't cost too much space.

Reductions

The main sword in the theory of computation

Informal Definition (Reduction).

We say problem A can be reduced to problem B, denoted as

$$A \leq B$$

if one can "efficiently" compute A with the help of B.

A	B
Multiplication	Addition
Feasible computation	Turing machine
Circuit-SAT	3SAT

Food for Thought

Q: Which do you find being more intuitive? Circuits or Turing machines? Why?

Q: Do you think here we elaborate all the possible computational resources? If no, can you name more?

Q: Suppose something is easily verifiable (e.g., Where's Wally), is it also easy to prove it wrong (i.e., is it easy to prove that Wally is not in the picture?)?

Exercise

- Think about how to explain the P vs. NP problem to your friends or families! In particular, try to use both an intuitive example and a slightly mathematical formulation.
- Can you come up with some examples of reductions? Either in CS, other fields, or even in daily life.

References

Articles:

- Horswill, Ian. “What is computation?”, 2008, link.
- Wigderson, Avi. “The Godel Phenomenon in Mathematics: A Modern View.” Kurt Gödel and the Foundations of Mathematics (2011): 475, link.

Introductory Books:

- Moore, Cristopher, and Stephan Mertens. The nature of computation. OUP Oxford, 2011, link.
- Barak, Boaz. Introduction to Theoretical Computer Science. Online, link.
- Rautenberg, Wolfgang. A concise introduction to mathematical logic. New York, NY: Springer, 2006, link.

Advanced Books:

- Wigderson, Avi. Mathematics and computation. Princeton University Press, 2019, link.
- Arora, Sanjeev, and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009, link.

Fun reads:

- Aaronson, Scott. “Why philosophers should care about computational complexity.” Computability: Turing, Gödel, Church, and Beyond 261 (2013): 327, link.

* Many icons in the slides were made by Freepik from www.flaticon.com