CS 121: Lecture 6
Code = Data

Madhu Sudan

https://madhu.seas.Harvard.edu/courses/Fall2020

Book: https://introtcs.org

How to contact us
- The whole staff (faster response): CS 121 Piazza
- Only the course heads (slower): cs121.fall2020.course.heads@gmail.com

Administrative

HW1 Graded?

HW2 out, Section 2 cycle ongoing, 1^{st} Midterm in 3 weeks

121.5: Ryan O'Donnell: Analysis of Boolean Functions

Where we are:

Part I: Circuits:
Finite computation, quantitative study

Part II: Automata:
Infinite restricted computation, quantitative study

Part III: Turing Machines:
Infinite computation, qualitative study

Part IV: Efficient Computation:
Infinite computation, quantitative study

Part V: Randomized computation:
Extending studies to non-classical algorithms

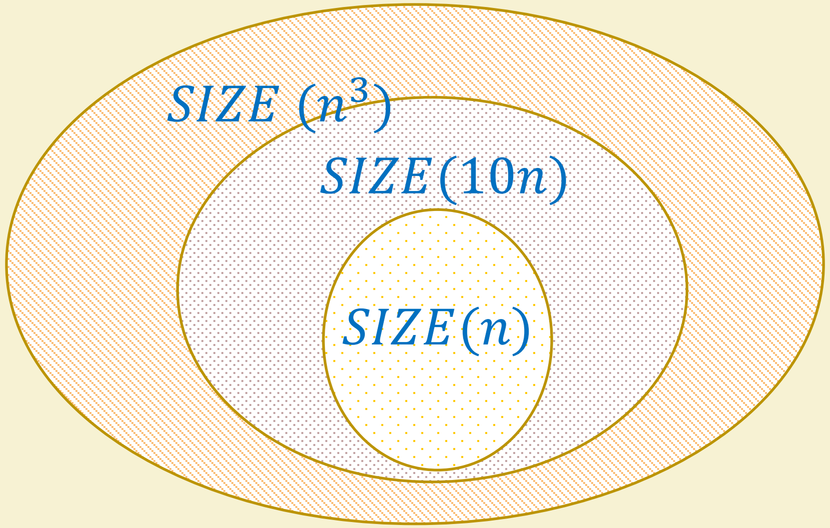
- Definition of Circuits
- Universality of NAND
- All functions can be computed
- $\forall f: \{0,1\}^n \to \{0,1\}$: Size(f) $\leq O\left(\frac{2^n}{n}\right)$
- Claimed: $\exists f, \backslash\mathrm{Size}(f) \geq \Omega\left(\frac{2^n}{n}\right)$
- Today: Will prove above.
- Show: Code = Data
- Show how to interpret Data as Code.

Today: Code as Data

- Circuits can be represented by bits
- Exercise break: Quantify above. Prove lower bound on Circuit size (for hardest function).
- Universality: Circuit Interpreter I(C,x) = C(x)
  - As immediate consequence of "Code as data" - Inefficient
- Efficient Circuit Interpreter (sketch)
- Exercise break: Some Ingredients

Notation: $\mathbf{SIZE}(s)$

- $\mathrm{SIZE}(s) = \{f: \{0,1\}^n \to \{0,1\} \mid \exists C \text{ with } \leq s \text{ NAND gates computing } f\}$
- $\mathrm{SIZE}(s)$ = Our first complexity class!
  - Always a set (aka “class”) of functions, not algorithms !
  - Is the following in $\mathrm{SIZE}(3)$? $\mathrm{SIZE}(10)$?
- $\mathrm{ALL}_n = \left\{f: \{0,1\}^n \to \{0,1\}\right\}$
- Thm: $\mathrm{ALL}_n \subseteq \mathrm{SIZE}\left(O\left(\frac{2^n}{n}\right)\right)$
- (Claimed) Thm: $\mathrm{ALL}_n \nsubseteq \mathrm{SIZE}\left(o\left(\frac{2^n}{n}\right)\right)$



Reminder: Circuit ≡ Straightline Program

Temp[0] ← NAND($X[0], X[1]$)

Temp[1] ← NAND($X[2], X[2]$)

…

Temp[i] ← NAND($\mathrm{Temp}[j], X[k]$)

…

Y[0] ← NAND($X[0], X[1]$)

…

Y[m-1] ← NAND($X[0], X[1]$)

Encode to $\{0,1\}^*$ in class

Exercise Break 1:

1. Make Representation quantitative:
   - Give (prefix-free) E: Circuits $\rightarrow \{0,1\}^*$, such that $\forall\ C$ with $\leq s$ gates, $|\boldsymbol{E}(\boldsymbol{C})| = \boldsymbol{O}(\boldsymbol{s} \log \boldsymbol{s})$
2. Show $|\boldsymbol{SIZE}(\boldsymbol{s})| = \ \boldsymbol{2}^{\boldsymbol{O}(\boldsymbol{s} \setminus \boldsymbol{log}\ \boldsymbol{s})}$
3. Show $\exists f: \{0,1\}^n \rightarrow \{0,1\}$ s.t. $f \notin \mathrm{SIZE}\left(o\left(\frac{2^n}{n}\right)\right)$ $\left(\Leftrightarrow\ \mathbf{ALL}_{\boldsymbol{n}} \not\subseteq \boldsymbol{SIZE}\left(\boldsymbol{o}\left(\frac{\mathbf{2}^{\boldsymbol{n}}}{\boldsymbol{n}}\right)\right)\right)$

Bonus question (0 points):
Why do Boaz Barak's slides associate this picture with 3rd exercise!

Interpreting Code

- Objective: Show Data representing Code can be interpreted as code.
- Have just shown: $\exists E$: Circuits $\mapsto$ binary string, 1-to-1

$$C \text{ has} \leq s \text{ NAND Gates} \quad \Rightarrow \quad |E(C)| = O(s \log s)$$

- $\mathrm{EVAL}: (E(C), x) \mapsto C(x),\ \forall C \text{ with } n \text{ inputs}, x \in \{0,1\}^n$
  - EVAL is a partial function – why?

- $\mathrm{EVAL}_{m,n}$ = restriction of EVAL to $E(C) \in \{0,1\}^m, x \in \{0,1\}^n$
  - Thm: $\mathrm{EVAL}_{m,n}$ computed by circuit of size $O(2^{m+n})$
  - Proof: Obvious!
  - Implication: Power of Code↔Data-duality!

Interpreting Circuits Efficiently.

- Goal: Show $\mathrm{EVAL}_{m,n} \in \mathrm{SIZE}\left(O\left((m+n)^2 \log^2 n\right)\right)$
  - Theorem 5.3 in Barak's IntroTCS.
  - Best bound in literature: close to $O\left((m+n)\log^2(m+n)\right)$
  - Great, but not "Meta-circular interpreter" (small interpreter that interprets bigger functions).

Sketch of EVAL

- Recall: $E(C) = \big((i_0, j_0, k_0) \dots (i_{s-1}, j_{s-1}, k_{s-1})\big) \quad (s \leq m)$
- Define: $W_t \in \{0,1\}^{n+s}$: Values of n inputs, s TEMPs after t execution steps
- Define:
  - $\mathrm{EVAL-ITER}: (E(C), x, t) \mapsto W_t$
  - $\mathrm{EVALHELP}: (W_{t-1}, i_t, j_t, k_t) \mapsto W_t$;
  - $\mathrm{EVAL-ITER}(E(C), x, t) = \mathrm{EVALHELP}(\mathrm{EVAL-ITER}(E(C), x, t-1), i_t, j_t, k_t)$
  - Suffices to show $\mathrm{EVALHELP} \in \mathrm{SIZE}((m+n)\log m + n)$

Sketch of $\mathbf{EVALHELP}$

- Key Ingredients:
  - $\mathrm{LOOKUP}(W, i) = W_i$ where $W = W_0 \ldots W_{m-1} \in \{0,1\}^m$, $i \in [m]$ represented in binary.
  - $\mathrm{UPDATE}(W, k, b) = \widehat{W}$ where $\widehat{W}_k = b$ and $\widehat{W}_\ell = W_\ell$ for $\ell \neq k$
  - Claims:
    - $\mathrm{LOOKUP} \in \mathrm{SIZE}(m)$
    - Exercise: $\mathrm{UPDATE} \in \mathrm{SIZE}(m^2)$ (even better $\mathrm{SIZE}(m \log m)$)
      - Don't have to work out details. Think of the high-level plan.
  - $\mathrm{EVALHELP}(W, i, j, k) = \mathrm{UPDATE}\Big(W, k, \mathrm{NAND}\big(\mathrm{LOOKUP}(W, i), \mathrm{LOOKUP}(W, j)\big)\Big)$

Exercise Break 2:

- $\mathrm{UPDATE}(W, k, b) = \widehat{W}$ where $\widehat{W}_k = b$ and $\widehat{W}_\ell = W_\ell$ for $\ell \neq k$

  $W, \widehat{W} \in \{0,1\}^m, k \in [m]$ represented in binary

- Exercise:
  - Show $\mathrm{UPDATE} \in \mathrm{SIZE}(m^2)$ (even better $\mathrm{SIZE}(m \log m)$)
    - Don't have to work out details. Think of the high-level plan.

Circuits: What you need to know

Theorem I: Every function $f: \{0,1\}^n \to \{0,1\}$ can be computed by circuit of size $O(2^n/n)$.

Theorem II: Some functions $f: \{0,1\}^n \to \{0,1\}$ cannot be computed by circuits of size $o(2^n/n)$.

SIZE Hierarchy Theorem: Book + Section/HW

Thm 5.11: $\exists C$ ($C = 1000$ will do) .s.t $\forall s < \frac{2^n}{Cn}$, $SIZE_{n,1}(s) \subsetneq SIZE_{n,1}(C \cdot s)$

* If f outputs m bits then add factor m to Thm I,II

Size Hierarchy Theorem (Sec 5.5)

Thm 5.11: $\exists C$ ($C = 1000$ will do) .s.t $\forall s < \frac{2^n}{Cn}$, $SIZE_{n,1}(s) \subsetneq SIZE_{n,1}(C \cdot s)$

Special case: $SIZE_{n,1}(n) \subsetneq SIZE_{n,1}(n^2)$

Proof: we know for every ℓ:

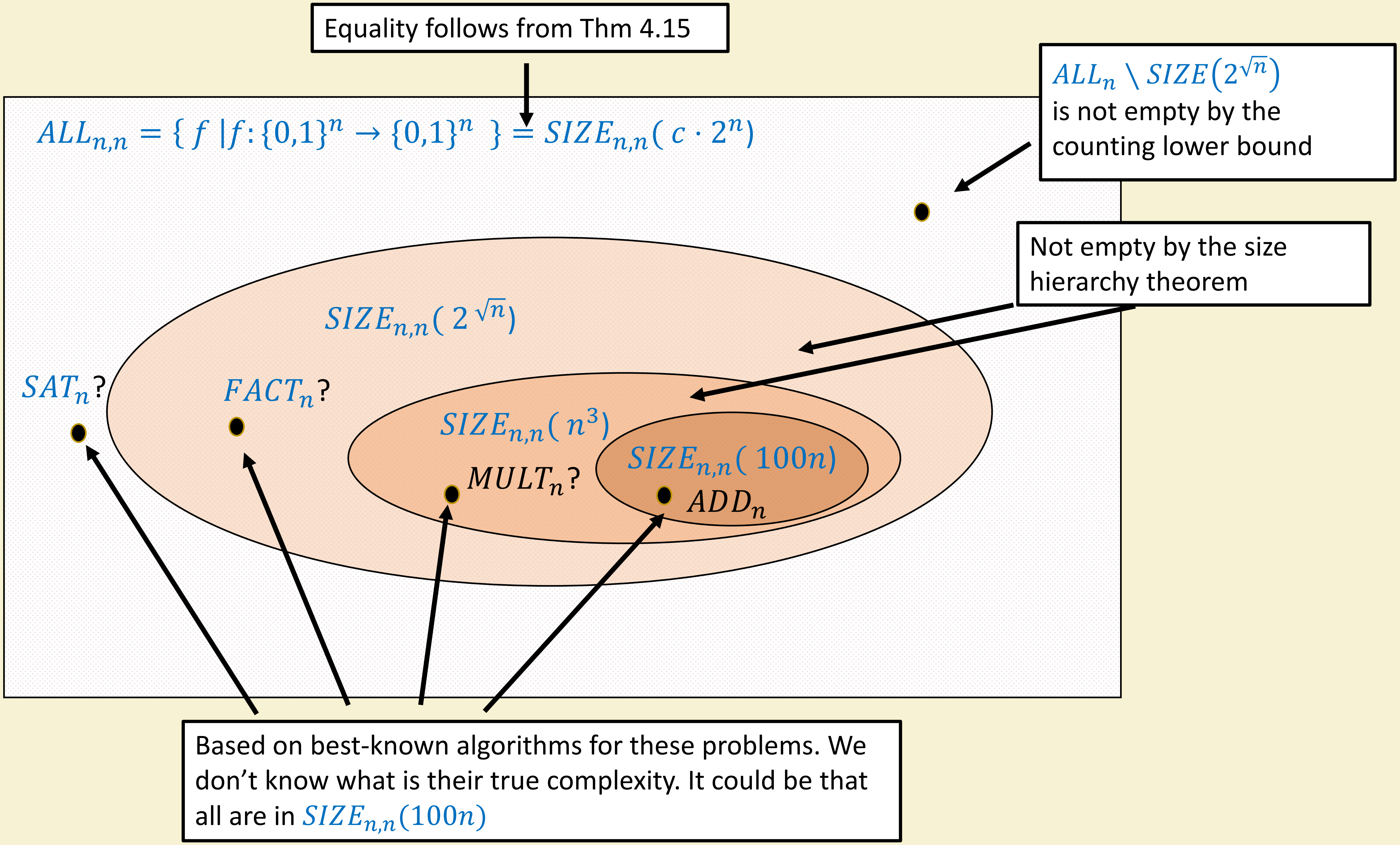
- $\forall f: \{0,1\}^\ell \to \{0,1\}$, $f \in SIZE_{\ell,1}\left(c \cdot 2^\ell/\ell\right)$
- $\exists f^*: \{0,1\}^\ell \to \{0,1\}$, f $\notin SIZE_{\ell,1}(\delta \cdot 2^\ell/\ell)$

Set ℓ s.t. $n^2 = c2^\ell/\ell$, define: $g^*(x_0 \cdots x_{n-1}) = f^*(x_0 \cdots x_{\ell-1})$

$$g^* \in SIZE_{n,1}\left(c2^\ell/\ell\right) \setminus SIZE_{n,1}\left(\delta \cdot 2^\ell/\ell\right)$$

$$||$$

$$SIZE_{n,1}\left(\frac{\delta}{c} \cdot n^2\right) \supseteq SIZE_{n,1}(n) \quad \blacksquare$$


Equality follows from Thm 4.15
$ALL_{n,n} = \{ f \mid f: \{0,1\}^n \to \{0,1\}^n \} = SIZE_{n,n}(c \cdot 2^n)$
$ALL_n \setminus SIZE(2^{\sqrt{n}})$ is not empty by the counting lower bound
Not empty by the size hierarchy theorem
$SIZE_{n,n}(2^{\sqrt{n}})$
SAT_n?
$FACT_n$?
$SIZE_{n,n}(n^3)$
$MULT_n$?
$SIZE_{n,n}(100n)$
ADD_n
Based on best-known algorithms for these problems. We don't know what is their true complexity. It could be that all are in $SIZE_{n,n}(100n)$

Extended Church Turing Thesis (circuit version)

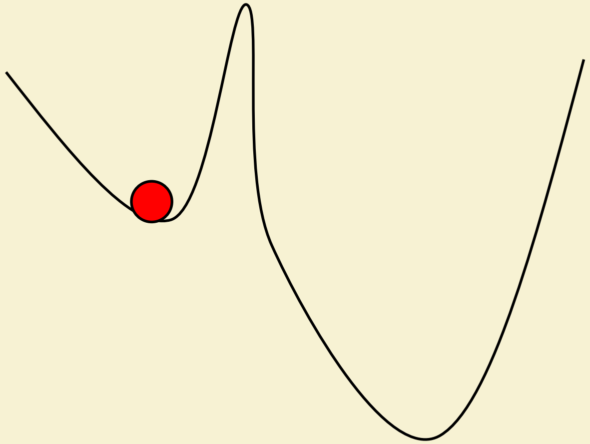
If $f: \{0,1\}^n \to \{0,1\}^m$ can be computed in the physical world using s resources then f can be computed by circuit of $\approx s$ (e.g. $O(s^2)$ or $O(s^3)$) gates.

(finite function version – we'll see unbounded function version soon)

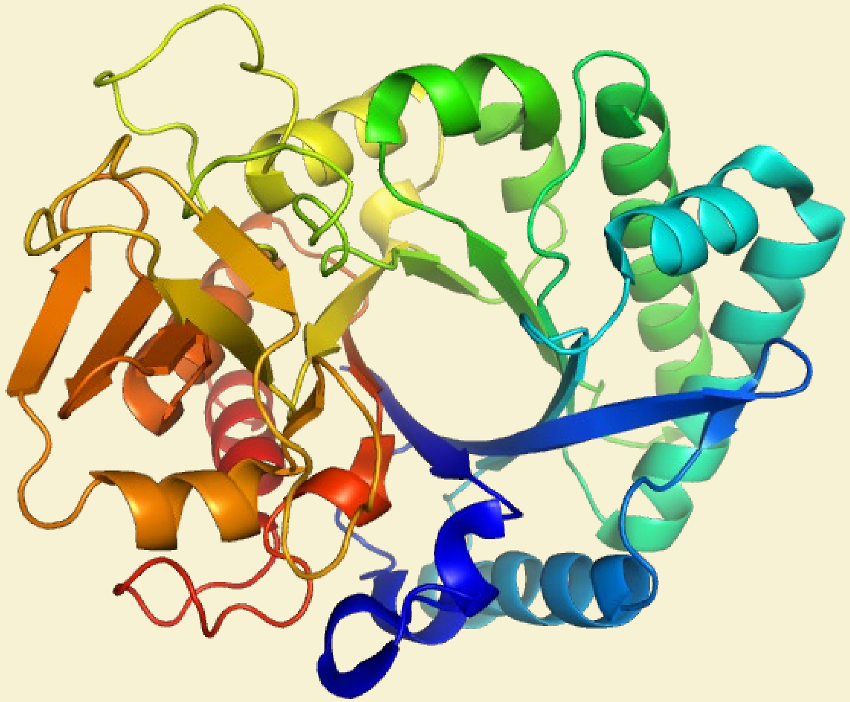
TL;DR: So far still stands. Only serious challenge is *quantum computing* which we'll see later.

Non-serious challenges: *(Following slides stolen from Boaz Barak who stole it from Scott Aaronson)*

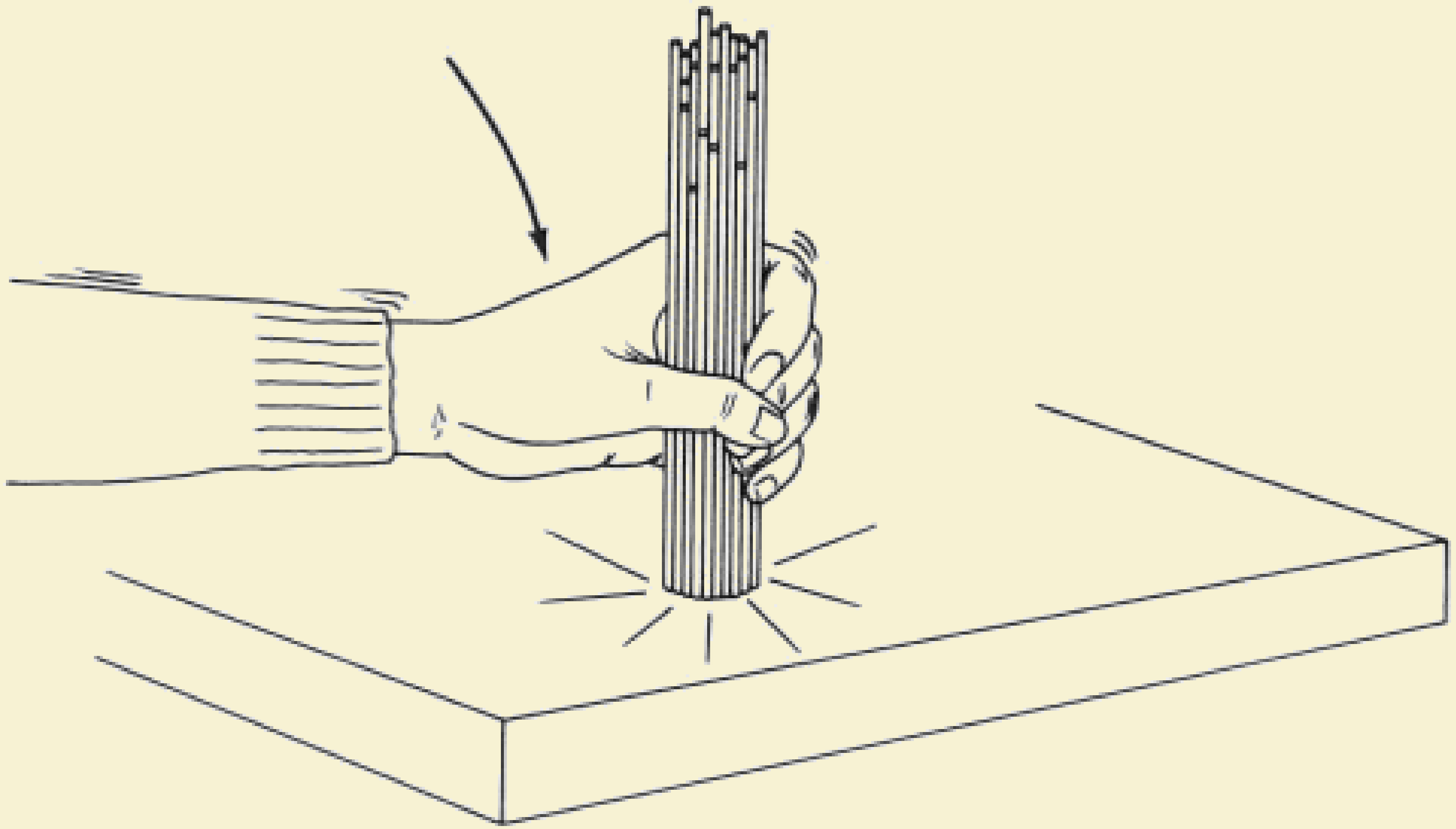
Soap Bubble Computer



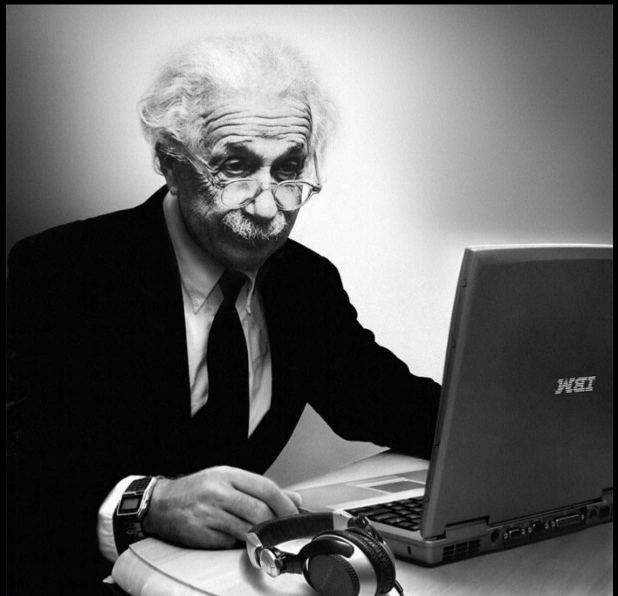
Protein Folding



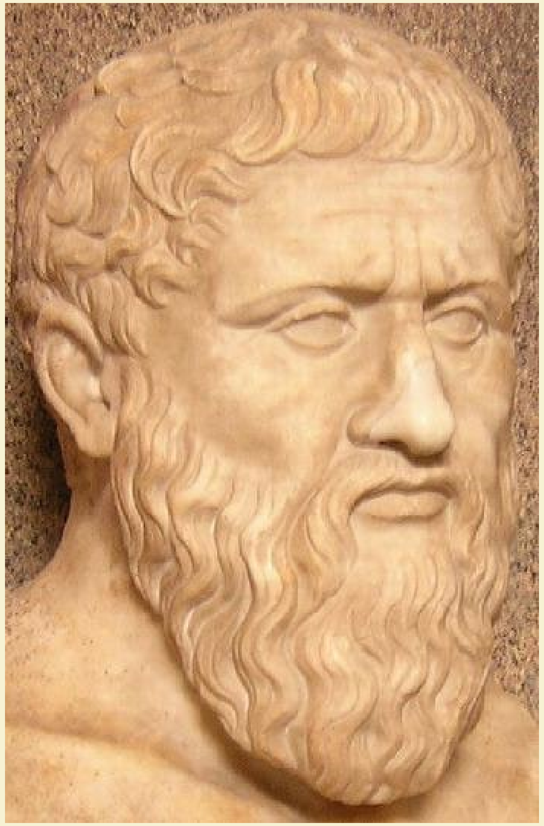
Spaghetti Sort

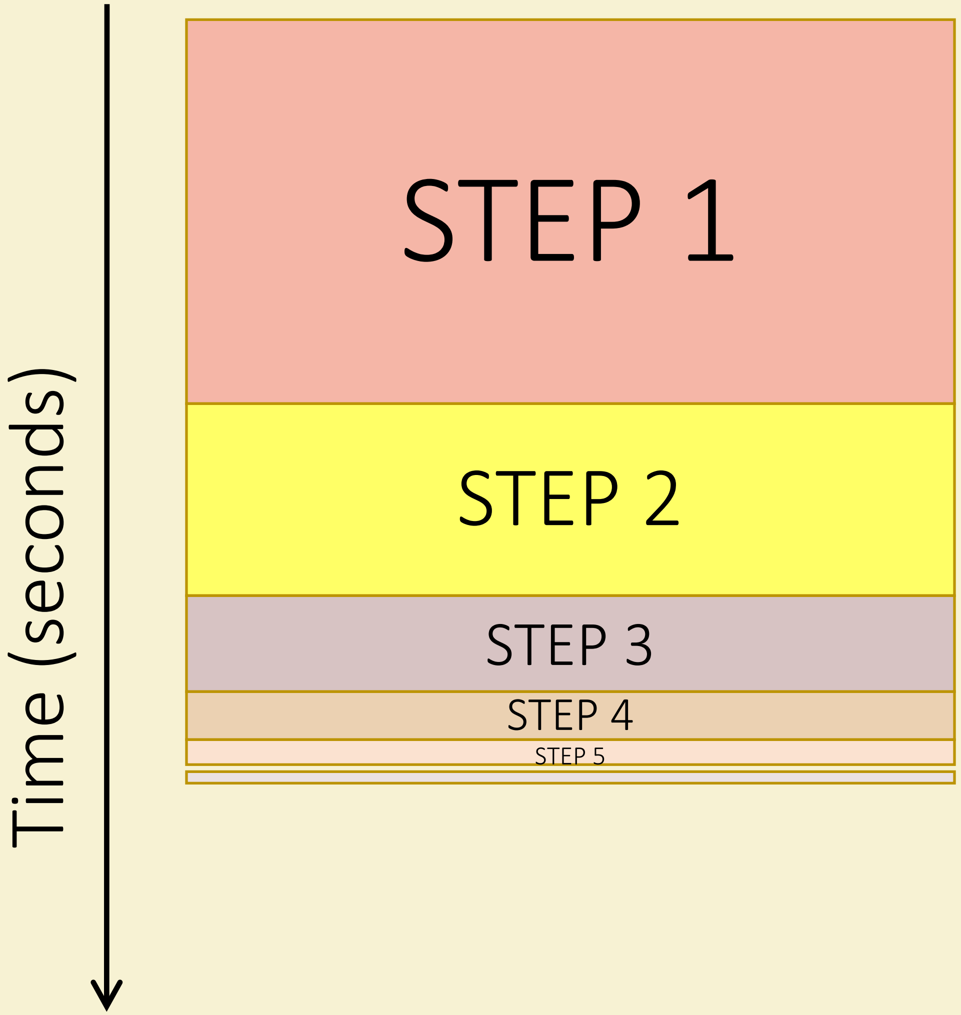


Relativity Computer
(cf. Malament and Hogarth)

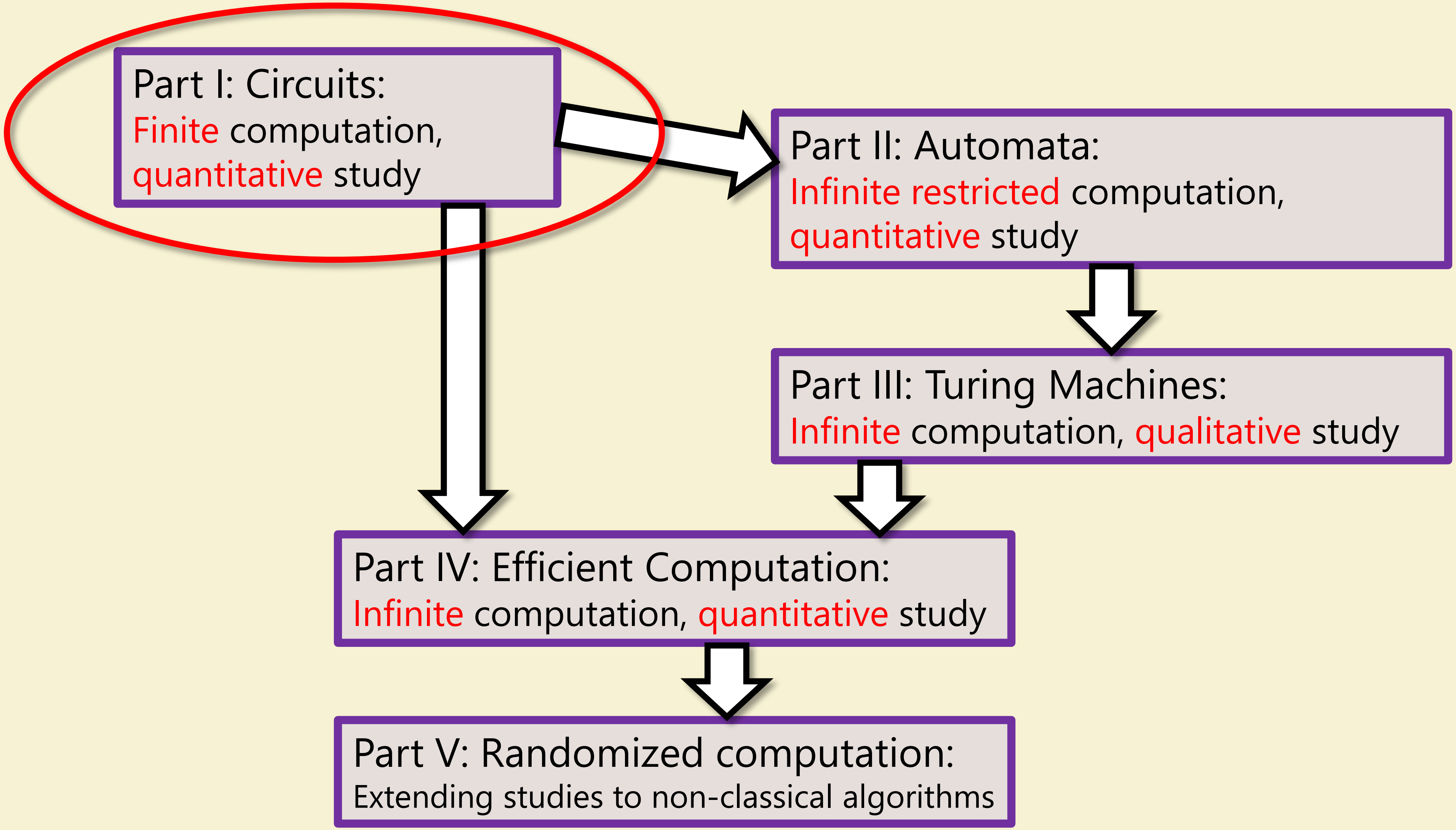


Zeno’s Computer

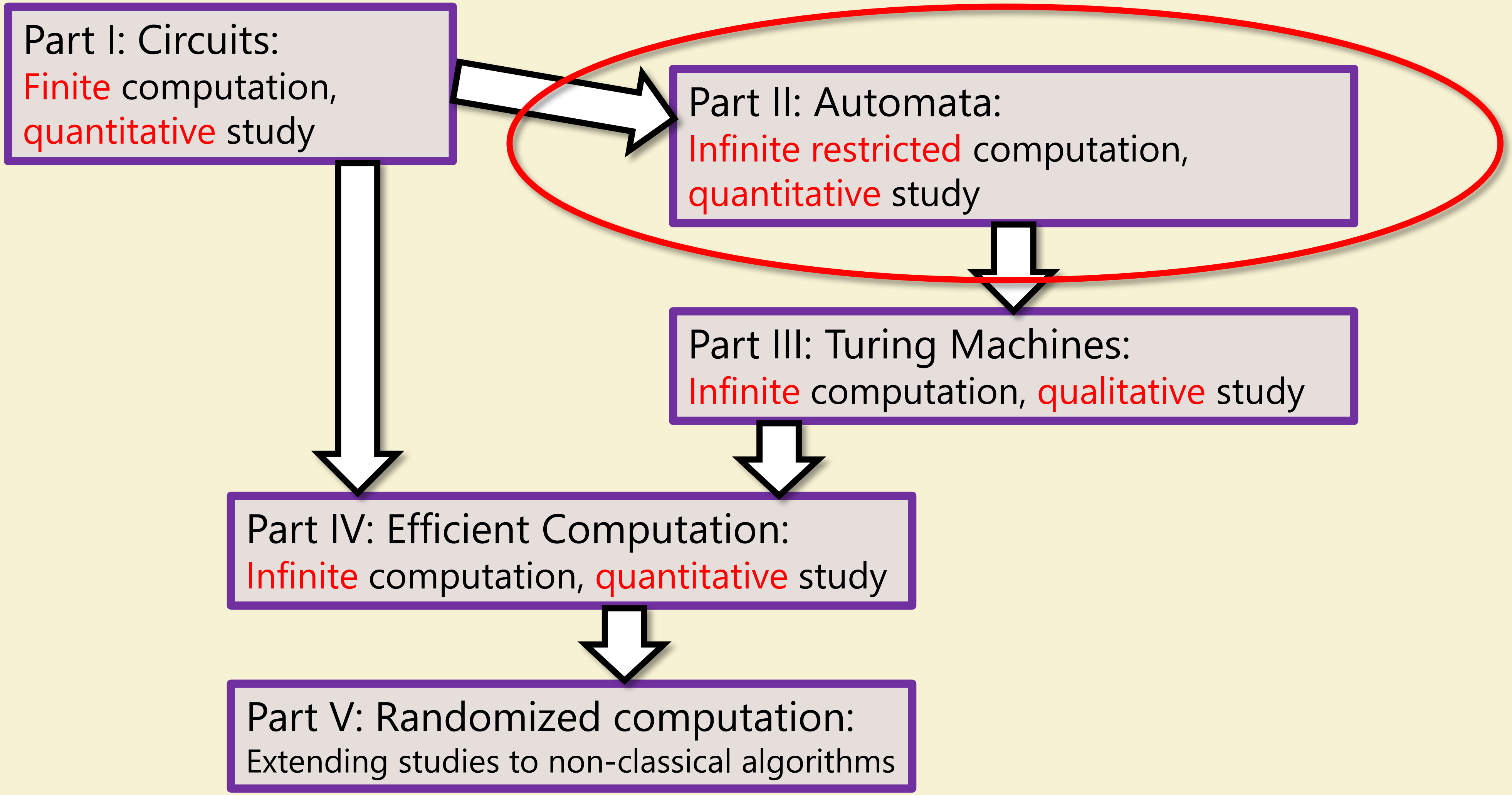




Where we are:


Part I: Circuits:
Finite computation,
quantitative study
Part II: Automata:
Infinite restricted computation,
quantitative study
Part III: Turing Machines:
Infinite computation, qualitative study
Part IV: Efficient Computation:
Infinite computation, quantitative study
Part V: Randomized computation:
Extending studies to non-classical algorithms

Where we are:


Part I: Circuits:
Finite computation, quantitative study
Part II: Automata:
Infinite restricted computation, quantitative study
Part III: Turing Machines:
Infinite computation, qualitative study
Part IV: Efficient Computation:
Infinite computation, quantitative study
Part V: Randomized computation:
Extending studies to non-classical algorithms