Administrative

HW1 Graded?

HW2 out, Section 2 cycle ongoing, 1st 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
  \[ \forall f : \{0,1\}^n \rightarrow \{0,1\} : \text{Size}(f) \leq O\left(\frac{2^n}{n}\right) \]
- Claimed: \( \exists f, \text{\textbackslash Size}(f) \geq \Omega\left(\frac{2^n}{n}\right) \)
- 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: \(\text{SIZE}(s)\)

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

\begin{align*}
\text{Temp}[0] & \leftarrow \text{NAND}(X[0], X[1]) \\
\text{Temp}[1] & \leftarrow \text{NAND}(X[2], X[2]) \\
\text{...} \\
\text{Temp}[i] & \leftarrow \text{NAND}(\text{Temp}[j], X[k]) \\
\text{...} \\
\text{Y}[0] & \leftarrow \text{NAND}(X[0], X[1]) \\
\text{...} \\
\text{Y}[m-1] & \leftarrow \text{NAND}(X[0], X[1])
\end{align*}

Encode to \{0,1\}^* in class
Exercise Break 1:

1. Make Representation quantitative:
   - Give (prefix-free) $E: \text{Circuits} \rightarrow \{0,1\}^*$, such that $\forall C$ with $\leq s$ gates, $|E(C)| = O(s \log s)$

2. Show $|\text{SIZE}(s)| = 2^{O(s \log s)}$

3. Show $\exists f: \{0,1\}^n \rightarrow \{0,1\}$ s.t. $f \notin \text{SIZE} \left( o \left( \frac{2^n}{n} \right) \right)$ $ \iff \text{ALL}_n \notin \text{SIZE} \left( o \left( \frac{2^n}{n} \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: \( \exists E: \text{Circuits} \mapsto \text{binary string}, \ 1\text{-to}-1 \)

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

- 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?

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

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

• Recall: \( E(C) = ((i_0, j_0, k_0) ... (i_{s-1}, j_{s-1}, k_{s-1})) \) \( (s \leq m) \)

• Define: \( W_t \in \{0,1\}^{n+s} \): Values of \( n \) inputs, \( s \) TEMP\s after \( t \) execution steps

• Define:
  - EVAL − ITER: \((E(C), x, t) \mapsto W_t\)
  - EVALHELP: \((W_{t-1}, i_t, j_t, k_t) \mapsto W_t\)
  - EVAL − ITER\((E(C), x, t) = EVALHELP(EVAL − ITER(E(C), x, t - 1), i_t, j_t, k_t)\)
  - Suffices to show \( EVALHELP \in SIZE((m + n) \log m + n)\)
Sketch of EVALHELP

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

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

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

- **Exercise:**
  - Show $\text{UPDATE} \in \text{SIZE}(m^2)$ (even better $\text{SIZE}(m \log m)$)
    - Don’t have to work out details. Think of the high-level plan.
Circuits: What you need to know

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

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

SIZE Hierarchy Theorem: Book + Section/HW

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

* If \( f \) outputs \( m \) bits then add factor \( m \) to Thm I,II
Size Hierarchy Theorem (Sec 5.5)

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

Special case: \( \text{SIZE}_{n,1}(n) \subsetneq \text{SIZE}_{n,1}(n^2) \)

Proof: we know for every \( \ell \):

- \( \forall f: \{0,1\}^\ell \to \{0,1\}, \ f \in \text{SIZE}_{\ell,1}(c \cdot 2^\ell / \ell) \)
- \( \exists f^*: \{0,1\}^\ell \to \{0,1\}, \ f \notin \text{SIZE}_{\ell,1}(\delta \cdot 2^\ell / \ell) \)

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

\[ g^* \in \text{SIZE}_{n,1}(c2^\ell / \ell) \setminus \text{SIZE}_{n,1}(\delta \cdot 2^\ell / \ell) \]

\[ \text{SIZE}_{n,1} \left( \frac{\delta}{c} \cdot n^2 \right) \supseteq \text{SIZE}_{n,1}(n) \]
Based on best-known algorithms for these problems. We don’t know what is their true complexity. It could be that all are in \( \text{SIZE}_{n,n}(100n) \).

\[ \text{ALL}_{n,n} = \{ f \mid f : \{0,1\}^n \to \{0,1\}^n \} = \text{SIZE}_{n,n}(c \cdot 2^n) \]

Equality follows from Thm 4.15

\( \text{ALL}_n \setminus \text{SIZE}(2^{\sqrt{n}}) \) is not empty by the counting lower bound

Not empty by the size hierarchy theorem
Extended Church Turing Thesis (circuit version)

If $f: \{0,1\}^n \rightarrow \{0,1\}^m$ 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

STEP 1

STEP 2

STEP 3

STEP 4

STEP 5

Time (seconds)
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