Introduction

Power from an unexpected source?

• we know \( P \neq EXP \), which implies no poly-time algorithm for Succinct CVAL

• poly-size Boolean circuits for Succinct CVAL ??

Does \( NP \) have linear-size, log-depth Boolean circuits ??
Outline

• Boolean circuits and formulas
• uniformity and advice
• the NC hierarchy and parallel computation
• the quest for circuit lower bounds
• a lower bound for formulas
Boolean circuits

• circuit $C$
  – directed acyclic graph
  – nodes: AND ($\land$); OR ($\lor$); NOT ($\neg$); variables $x_i$

• $C$ computes function $f: \{0,1\}^n \rightarrow \{0,1\}$ in natural way
  – identify $C$ with function $f$ it computes
Boolean circuits

- **size** = # gates
- **depth** = longest path from input to output
- **formula (or expression)**: graph is a tree

- every function \( f: \{0,1\}^n \rightarrow \{0,1\} \) computable by a circuit of size at most \( O(n2^n) \)
  - AND of \( n \) literals for each \( x \) such that \( f(x) = 1 \)
  - OR of up to \( 2^n \) such terms
Circuit families

• circuit works for specific input length
• we’re used to $f: \sum^* \rightarrow \{0,1\}$
• circuit **family** : a circuit for each input length $C_1, C_2, C_3, \ldots = \{C_n\}$
• $\{C_n\}$ computes $f$ iff for all $x$
  $$C_{|x|}(x) = f(x)$$
• $\{C_n\}$ decides $L$, where $L$ is the language associated with $f$
Connection to TMs

- given TM M running in time t(n) decides language L
- can build circuit family \( \{C_n\} \) that decides L
  - size of \( C_n = O(t(n)^2) \)
  - Proof: CVAL construction

- Conclude: \( L \in P \) implies family of polynomial-size circuits that decides L
Connection to TMs

• other direction?

• A poly-size circuit family:
  \[ C_n = (x_1 \lor \neg x_1) \text{ if } M_n \text{ halts} \]
  \[ C_n = (x_1 \land \neg x_1) \text{ if } M_n \text{ loops} \]

• decides (unary version of) HALT!

• oops…

April 17, 2017
Uniformity

• Strange aspect of circuit family:
  – can “encode” (potentially uncomputable) information in family specification

• solution: uniformity – require specification is simple to compute

Definition: circuit family \{C_n\} is \textcolor{red}{\text{logspace uniform}} iff TM M outputs \(C_n\) on input \(1^n\) and runs in \(O(\log n)\) space
Uniformity

**Theorem:** \( P = \) languages decidable by logspace uniform, polynomial-size circuit families \( \{C_n\} \).

- **Proof:**
  - already saw (\( \Rightarrow \))
  - (\( \Leftarrow \)) on input \( x \), generate \( C_{|x|} \), evaluate it and accept iff output = 1
TMs that take advice

- family \( \{ \mathcal{C}_n \} \) without uniformity constraint is called "non-uniform"

- regard "non-uniformity" as a limited resource just like time, space, as follows:
  - add read-only "advice" tape to TM \( M \)
  - \( M \) "decides \( L \) with advice \( A(n) \)" iff
    
    \[
    M(x, A(|x|)) \text{ accepts } \iff x \in L
    \]
  - note: \( A(n) \) depends only on \( |x| \)
TM that take advice

- Definition: $\text{TIME}(t(n))/f(n) = \{L | \text{exists } A(n) \text{ s.t. } |A(n)| \leq f(n) \text{ and TM M decides L with advice A(n) in time } t(n)\}$

- most important such class: $P/poly = \bigcup_k \text{TIME}(n^k)/n^k$
TMs that take advice

**Theorem:** \( L \in \text{P/poly} \) iff \( L \) decided by family of (non-uniform) polynomial size circuits.

• **Proof:**
  - \((\Rightarrow)\) \( C_n \) from CVAL construction; hardwire advice \( A(n) \)
  - \((\Leftarrow)\) define \( A(n) = \text{description of } C_n \); on input \( x \), TM simulates \( C_{|x|}(x) \)
Approach to P/NP

• Believe $\text{NP} \not\subset \text{P}$
  – equivalent: “$\text{NP}$ does not have uniform, polynomial-size circuits”

• *Even believe* $\text{NP} \not\subset \text{P/poly}$
  – equivalent: “$\text{NP}$ (or, e.g. SAT) does not have polynomial-size circuits”
  – implies $\text{P} \neq \text{NP}$
  – many believe: best hope for $\text{P} \neq \text{NP}$
Parallelism

• uniform circuits allow refinement of polynomial time:

\[ \text{circuit } C, \text{ depth } \equiv \text{ parallel time}, \text{ size } \equiv \text{ parallel work} \]
Parallelism

- the **NC** (“Nick’s Class”) Hierarchy (of logspace uniform circuits):
  \[ \text{NC}_k = O(\log^k n) \text{ depth, poly}(n) \text{ size} \]
  \[ \text{NC} = \bigcup_k \text{NC}_k \]
- captures “efficiently parallelizable problems”
- not realistic? overly generous
- OK for proving non-parallelizable
Matrix Multiplication

• what is the parallel complexity of this problem?
  – work = poly(n)
  – time = log^k(n)? (which k?)
Matrix Multiplication

• two details
  – arithmetic matrix multiplication…
    \[ A = (a_{i,k}) \quad B = (b_{k,j}) \quad (AB)_{i,j} = \sum_k (a_{i,k} \times b_{k,j}) \]
    
    … vs. Boolean matrix multiplication:
    \[ A = (a_{i,k}) \quad B = (b_{k,j}) \quad (AB)_{i,j} = \bigvee_k (a_{i,k} \wedge b_{k,j}) \]
    
  – single output bit: to make matrix multiplication a language: on input \( A, B, (i, j) \) output \( (AB)_{i,j} \)
Matrix Multiplication

- Boolean Matrix Multiplication is in $\mathbf{NC}_1$
  - level 1: compute $n$ ANDS: $a_{i,k} \land b_{k,j}$
  - next $\log n$ levels: tree of ORS
    - $n^2$ subtrees for all pairs ($i, j$)
    - select correct one and output
Boolean formulas and $\text{NC}_1$

- Previous circuit is actually a formula. This is no accident:

**Theorem:** $L \in \text{NC}_1$ iff decidable by polynomial-size uniform* family of Boolean formulas.

* $\text{DSPACE}(\log^2 n)$-uniform

Note: we measure formula size by leaf-size.
Boolean formulas and $\textbf{NC}_1$

- Proof:
  - $(\iff)$ convert $\textbf{NC}_1$ circuit into formula
    - recursively:

  ![Diagram of circuits]

  - note: logspace transformation (stack depth $\log n$, stack record 1 bit – “left” or “right”)
Boolean formulas and $\text{NC}_1$

$\neg (\iff)$ convert formula of size $n$ into formula of depth $O(\log n)$

- note: size $\leq 2^{\text{depth}}$, so new formula has poly(n) size

key transformation
Boolean formulas and $\mathbf{NC}_1$

– D any minimal subtree with size at least $n/3$
  • implies size(D) ≤ $2n/3$

– define $T(n) =$ maximum depth required for any size $n$ formula

– $C_1, C_0, D$ all size ≤ $2n/3$

$$T(n) \leq T(2n/3) + 3$$

implies $T(n) \leq O(\log n)$
Relation to other classes

• Clearly $\text{NC} \subseteq \text{P}$
  – recall $\text{P} \equiv$ uniform poly-size circuits

• $\text{NC}_1 \subseteq \text{L}$
  – on input $x$, compose logspace algorithms for:
    • generating $C_{|x|}$
    • converting to formula
    • FVAL
Relation to other classes

• $\textbf{NL} \subseteq \textbf{NC}_2$: $\text{S-T-CONN} \in \textbf{NC}_2$
  
  – given $G = (V, E)$, vertices $s, t$
  
  – $A =$ adjacency matrix (with self-loops)
  
  – $(A^2)_{i,j} = 1$ iff path of length $\leq 2$ from node $i$ to node $j$
  
  – $(A^n)_{i,j} = 1$ iff path of length $\leq n$ from node $i$ to node $j$
  
  – compute with depth $\log n$ tree of Boolean matrix multiplications, output entry $s, t$
  
  – $\log^2 n$ depth total
NC vs. P

• can every efficient algorithm be efficiently parallelized?

NC = P

• P-complete problems least-likely to be parallelizable
  – if P-complete problem is in NC, then P = NC
  – Why?
    we use logspace reductions to show problem P-complete; L in NC
**NC vs. P**

- can every uniform, poly-size Boolean circuit family be converted into a uniform, poly-size Boolean formula family? 

\[ \text{NC}_1 = \mathbf{P} \]
NC Hierarchy Collapse

\[ \text{NC}_1 \subseteq \text{NC}_2 \subseteq \text{NC}_3 \subseteq \text{NC}_4 \subseteq \ldots \subseteq \text{NC} \]

**Exercise**

if \( \text{NC}_i = \text{NC}_{i+1} \), then \( \text{NC} = \text{NC}_i \)

(prove for non-uniform versions of classes)

April 17, 2017
Lower bounds

• Recall: “NP does not have polynomial-size circuits” \((NP \not\subseteq P/poly)\) implies \(P \neq NP\)

• major goal: prove lower bounds on (non-uniform) circuit size for problems in \(NP\)
  – believe exponential
  – super-polynomial enough for \(P \neq NP\)
  – best bound known: 4.5n
  – don’t even have super-polynomial bounds for problems in \(NEXP\)
Lower bounds

• lots of work on lower bounds for restricted classes of circuits

  – we’ll see two such lower bounds:
    • formulas
    • monotone circuits
Shannon’s counting argument

• frustrating fact: *almost all* functions require huge circuits

**Theorem** (Shannon): With probability at least $1 – o(1)$, a random function $f: \{0,1\}^n \rightarrow \{0,1\}$ requires a circuit of size $\Omega(2^n/n)$. 
Shannon’s counting argument

• Proof (counting):
  – \( B(n) = 2^{2^n} \) = # functions \( f: \{0,1\}^n \rightarrow \{0,1\} \)
  – # circuits with n inputs + size s, is at most

\[
C(n, s) \leq ((n+3)s^2)^s
\]
Shannon’s counting argument

\[ C(n, s) \leq ((n+3)s^2)^s \]

\[ -C(n, c2^n/n) < ((2n)c^2 2^{2n}/n^2)(c^{2n}/n) \]
\[ < o(1)2^{2c2^n} \]
\[ < o(1)2^{2^n} \quad \text{(if } c \leq \frac{1}{2} \text{)} \]

- Probability a random function has a circuit of size \( s = (\frac{1}{2})2^n/n \) is at most
\[ C(n, s)/B(n) < o(1) \]
Shannon’s counting argument

• frustrating fact: *almost all* functions require *huge formulas*

**Theorem** (Shannon): With probability at least $1 - o(1)$, a random function $f: \{0,1\}^n \to \{0,1\}$ requires a *formula* of size $\Omega(2^n / \log n)$. 

April 17, 2017
Shannon’s counting argument

• Proof (counting):
  – $B(n) = 2^{2^n} = \# \text{ functions } f: \{0,1\}^n \rightarrow \{0,1\}$
  – $\# \text{ formulas with } n \text{ inputs } + \text{ size } s$, is at most

$$F(n, s) \leq 4^s 2^s (2n)^s$$

- $4^s$ binary trees with $s$ internal nodes
- 2 gate choices per internal node
- 2n choices per leaf
Shannon’s counting argument

\[ F(n, s) \leq 4^s 2^s (2n)^s \]

- \[ F(n, c2^n / \log n) < (16n)^{(c2^n / \log n)} \]
  \[ < 16^{(c2^n / \log n)} 2^{(c2^n)} = (1 + o(1)) 2^{(c2^n)} \]
  \[ < o(1) 2^{2n} \quad \text{if } c \leq \frac{1}{2} \]

- probability a random function has a \textbf{formula} of size \( s = (\frac{1}{2}) 2^n / \log n \) is at most \( F(n, s) / B(n) < o(1) \)