Introduction

Power from an unexpected source?
• we know $P \neq \text{EXP}$, 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 $\text{NC}$ 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 $x_i$
  – $x_1, x_2, x_3, \ldots, x_n$

• $C$ computes function $f: \{0,1\}^n \to \{0,1\}$ 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 $f: \{0,1\}^n \to \{0,1\}$ 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 family : a circuit for each input length $C_1, C_2, C_3, \ldots = \{C_n\}$
• $\{C_n\}$ computes $f$ iff for all $x$
  $C_{|x|}(x) = f(x)$

• Definition: circuit family $\{C_n\}$ is logspace uniform iff TM $M$ outputs $C_n$ on input $1^n$ and runs in $O(\log n)$ space
TMs that take advice

- family \( \{C_n\} \) 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 \(|x|\)

Definition: \( \text{TIME}(t(n))/f(n) = \) the set of those languages \( L \) for which:
- there exists \( A(n) \) s.t. \(|A(n)| \leq f(n)\)
- TM \( M \) decides \( L \) with advice \( A(n) \) in time \( t(n) \)
- most important such class:
  \[ \text{P/poly} = \cup_k \text{TIME}(n^k)/n^k \]

Uniformity

**Theorem:** \( \text{P} \) = 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

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:

\[ \text{depth} = \text{parallel time} \]
\[ \text{size} = \text{parallel work} \]
Parallelism

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

Matrix Multiplication

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

Boolean formulas and $\textbf{NC}_1$

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

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

  Note: we measure formula size by leaf-size.
Boolean formulas and \textbf{NC}\textsubscript{1}

\begin{itemize}
  \item \((\leq)\) convert formula of size \(n\) into formula of depth \(O(\log n)\)
  \item note: size \(\leq 2^\text{depth}\), so new formula has \(\text{poly}(n)\) size
\end{itemize}

Relation to other classes

\begin{itemize}
  \item Clearly \(\text{NC} \subseteq \text{P}\)
    \begin{itemize}
      \item recall \(\text{P} = \text{uniform poly-size circuits}\)
    \end{itemize}
  \item \(\text{NC}_1 \subseteq \text{L}\)
    \begin{itemize}
      \item on input \(x\), compose \textit{logspace} algorithms for:
        \begin{itemize}
          \item generating \(C_{\text{val}}\)
          \item converting to formula
          \item FVAL
        \end{itemize}
    \end{itemize}
\end{itemize}

\textbf{NC} vs. \textbf{P}

\begin{itemize}
  \item can every \textit{efficient} algorithm be efficiently parallelized?
    \[\text{NC} \overset{?}{=} \text{P}\]
  \item \textbf{P}-complete problems least-likely to be parallelizable
    \begin{itemize}
      \item if \textbf{P}-complete problem is in \textbf{NC}, then \(\text{P} = \text{NC}\)
      \item Why?
        we use \textit{logspace} reductions to show \textbf{P}-complete; \textbf{L} in \textbf{NC}
    \end{itemize}
\end{itemize}
NC Hierarchy Collapse

$NC_1 \subseteq NC_2 \subseteq NC_3 \subseteq NC_4 \subseteq \ldots \subseteq NC$

**Exercise**

if $NC_i = NC_{i+1}$, then $NC = NC_i$

(prove for non-uniform versions of classes)

---

**Lower bounds**

- Recall: "**NP does not have polynomial-size circuits**" (NP $\not\subset$ 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

$f : \{0,1\}^n \rightarrow \{0,1\} \quad \Rightarrow \quad \text{ requires a circuit of size } \Omega(2^{\alpha n})$.

---

**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

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

- probability a random function has a circuit of size $s = (\frac{1}{2})2^{\alpha n}$ 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 $\Omega(2^{n/\log n})$.

Proof (counting):
- $B(n) = 2^{2^n} = \# \text{ functions } f : \{0,1\}^n \to \{0,1\}$
- $\# \text{ formulas with } n \text{ inputs + size } s$, is at most
  $$F(n, s) \leq 4^s 2^n \binom{s}{2} \cdot 2^n \cdot 2^{n/\log n}$$
  2 gate choices per internal node
  2n choices per leaf
  $4^s$ binary trees with $s$ internal nodes

- $F(n, s) \leq 4^s 2^n \binom{s}{2} \cdot 2^n \cdot 2^{n/\log n} \leq (16n)^{c 2^n/\log n}$
- $< 16 (c 2^n/\log n)^2 \left(1 + o(1)\right) 2^{(c/2)n}$
- $< o(1)$

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

Andreev function

- Best formula lower bound for language in NP:

  **Theorem** (Andreev, Hastad ’93): the Andreev function requires $\Omega(n^{3-o(1)})$-formulas of size at least $\Omega(n^{3-o(1)})$.

Random restrictions

- Key idea: given function $f : \{0,1\}^n \to \{0,1\}$ restrict by $\rho$ to get $f_\rho$
  - $\rho$ sets some variables to 0/1, others remain free

  - $R(n, \rho) = \text{ set of restrictions that leave } \rho \text{ variables free}$

  - Definition: $L(f)$ = smallest $(\land, \lor, \neg)$ formula computing $f$ (measured as leaf-size)
Random restrictions

- observation:
  
  \[ E_{R(n, m)}[L(f)] \leq \varepsilon L(f) \]

  - each leaf survives with probability \( \varepsilon \)

- may shrink more...
  
  - propagate constants

**Lemma** (Hastad 93): for all \( f \)

  \[ E_{R(n, m)}[L(f)] \leq O(\varepsilon^{2-o(1)}L(f)) \]

Hastad’s shrinkage result

- Proof of theorem:
  
  - Recall: there exists a function
    
    \[ h: (0,1)^{\log n} \rightarrow (0,1) \]

    for which \( L(h) > n/2\log \log n \).
  
  - hardwire truth table of that function into \( y \) to get \( A(x) \)
  
  - apply random restriction from \( R(n, m = 2(\log n)(\ln \log n)) \)

The lower bound

- Proof of theorem (continued):
  
  - probability given XOR is killed by restriction is probability that we “miss it” \( m \) times:
    
    \[ (1 - (n/\log n)/n)^m \leq (1 - 1/\log n)^m \leq (1/e)^{m/\log n} \leq 1/\log^2 n \]

    - probability even one of XORs is killed by restriction is at most:
      
      \[ \log n(1/\log^2 n) = 1/\log n < 1/2. \]

The lower bound

- (1): probability even one of XORs is killed by restriction is at most:
  
  \[ \log n(1/\log^2 n) = 1/\log n < 1/2. \]

- (2): by Markov:
  
  \[ \Pr[ L(A_{\rho}) > 2 E_{R(n, m)}[L(A_{\rho})] ] < 1/2. \]

- Conclude: for some restriction \( \rho \)
  
  - all XORs survive, and
  
  \[ L(A_{\rho}) \leq 2 E_{R(n, m)}[L(A_{\rho})] \leq O( (\log n)(\ln \log n)/\log^2 n) \]

  - implies \( \Omega(n^{3-o(1)}) \leq L(A) \leq L(A) \).

Clique

**CLIQUE** = \( \{ (G, k) | G \text{ is a graph with a clique of size} \geq k \} \)

- \( \text{clique} = \text{set of vertices every pair of which are connected by an edge} \)

  - **CLIQUE** is **NP-complete**.
Circuit lower bounds

- We think that $\textbf{NP}$ requires exponential-size circuits.
- Where should we look for a problem to attempt to prove this?
- Intuition: “hardest problems” – i.e., $\textbf{NP}$-complete problems

Circuit lower bounds

- Formally:
  - if any problem in $\textbf{NP}$ requires super-polynomial size circuits
  - then every $\textbf{NP}$-complete problem requires super-polynomial size circuits
- Proof idea: poly-time reductions can be performed by poly-size circuits using a variant of CVAL construction

Monotone problems

- Definition: monotone language = language $L \subseteq \{0,1\}^*$ such that $x \in L$ implies $x' \in L$ for all $x \leq x'$.
  - flipping a bit of the input from 0 to 1 can only change the output from “no” to “yes” (or not at all)

Monotone problems

- some $\textbf{NP}$-complete languages are monotone
  - e.g. CLIQUE (given as adjacency matrix):
  - others: HAMILTON CYCLE, SET COVER…
  - but not SAT, KNAPSACK…

Monotone circuits

A restricted class of circuits:

- Definition: monotone circuit = circuit whose gates are ANDs ($\wedge$), ORs ($\vee$), but no NOTs
- can compute exactly the monotone fnns.
  - monotone functions closed under AND, OR

Monotone circuits

- A question:
  Do all poly-time computable monotone functions have poly-size monotone circuits?
  - recall: true in non-monotone case
Monotone circuits

A monotone circuit for $\text{CLIQUE}_{n,k}$

- Input: graph $G = (V, E)$ as adj. matrix, $|V|=n$
  - variable $x_{ij}$ for each possible edge $(i,j)$
- $\text{ISCLIQUE}(S)$ = monotone circuit that = 1 if $S \subseteq V$ is a clique: $\land_{ij \in S} x_{ij}$
- $\text{CLIQUE}_{n,k}$ computed by monotone circuit: $\lor_{S \subseteq V, |S|=k} \text{ISCLIQUE}(S)$

- Size of this monotone circuit for $\text{CLIQUE}_{n,k}$:
  - when $k = n^{1/4}$, size is approximately:

$\left(\frac{n}{\binom{n}{k}}\right)^{k^2/2} \approx n^{\Omega(n^{1/4})}$

- Theorem (Razborov 85): monotone circuits for $\text{CLIQUE}_{n,k}$ with $k = n^{1/4}$ must have size at least $2^{\Omega(n^{1/8})}$.

- Proof:
  - next lecture

April 13, 2015