Introduction

Power from an unexpected source?

- we know $\mathbf{P} \neq \mathbf{EXP}$, which implies no poly-time \textit{algorithm for Succinct CVAL}
- poly-size Boolean \textit{circuits for Succinct CVAL} ??

...and the depths of our ignorance:

- Does $\mathbf{NP}$ have linear-size, log-depth Boolean circuits ??

Outline

- Boolean circuits and formulae
- uniformity and advice
- the $\mathbf{NC}$ hierarchy and parallel computation
- the quest for circuit lower bounds
- a lower bound for formulae

Boolean circuits

- \textbf{circuit C}
  - directed acyclic graph
  - nodes: AND ($\land$); OR ($\lor$); NOT ($\neg$); variables $x_1, x_2, x_3, \ldots, x_n$

- $C$ computes function $f: \{0,1\}^n \rightarrow \{0,1\}$ in natural way
  - identify $C$ with fn. $f$ it computes
- \textbf{size} = # gates
- \textbf{depth} = longest path from input to output
- \textbf{formula (or expression)}: graph is a tree

Circuit families

- \textbf{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
- \textbf{circuit works for specific input length} \textbf{we're using} $f: \Sigma^* \rightarrow \{0,1\}$
- \textbf{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 associated with $f$
Connection to TMs

- 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.
- \( L \in P \) implies family of polynomial size circuits that decide \( L \)
- other direction?
- poly-size circuit family:
  - \( C_n = (x_1 \lor \neg x_2) \) if \( M_n \) halts
  - \( C_n = (x_1 \land \neg x_3) \) if \( M_n \) loops
- decides (unary version of) HALT!
- oops...

Uniformity

- Strange aspect of circuit family:
  - can “encode” (uncomputable) information in family specification
- solution: uniformity - require specification is simple to compute
  - circuit family \( \{C_n\} \) is logspace uniform iff TM \( M \) outputs \( C_n \) on input \( 1^n \) and runs in \( O(\log n) \) space
- 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

TM’s that take advice

- family \( \{C_n\} \) without uniformity constraint is 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 \( M(x, A(|x|)) \) accepts \( \iff x \in L \)
  - note: \( A(n) \) depends only on \( |x| \)
  - \( L \) in \( \text{TIME}(t(n))/f(n) \) iff
    - exists \( A(n) \) s.t. \( |A(n)| \leq f(n) \)
    - TM \( M \) decides \( L \) with advice \( A(n) \)
- most important such class:
  \( P/\text{poly} = \bigcup_k \text{TIME}(n^k)/n^k \)

TM’s that take advice

- Theorem: \( L \in P/\text{poly} \) iff \( L \) decided by family of polynomial size circuits.
- Proof:
  - \((\Leftarrow)\) \( C_n \) from CVAL construction; hardwire advice \( A(n) \)
  - \((\Rightarrow)\) define \( A(n) = \) description of \( C_n \); on input \( x \), TM simulates \( C_n(x) \)
- Believe \( \text{NP} \not\subset P \)
  - equivalent: “\( \text{NP} \) does not have uniform, polynomial-size circuits”
- Even believe \( \text{NP} \not\subset P/\text{poly} \)
  - equivalent: “\( \text{NP} \) does not have polynomial-size circuits”
  - implies \( P \neq \text{NP} \)
  - many believe: best hope for \( P \neq \text{NP} \)
Parallelism

- uniform circuits allow refinement of polynomial time:

\[ \text{circuit} \quad \text{depth} = \text{parallel time} \]

\[ \text{size} = \text{parallel work} \]

- the 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

\[ n \times n \quad \text{matrix} A \quad n \times n \quad \text{matrix} B = n \times n \quad \text{matrix} AB \]

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

Matrix multiplication

- two details
  - arithmetic matrix multiplication...
    \[ A = (a_{i,j}) \quad B = (b_{i,j}) \]
    \[ (AB)_{i,j} = \sum_k (a_{i,k} \times b_{k,j}) \]
  - vs. Boolean matrix multiplication:
    \[ A = (a_{i,j}) \quad B = (b_{i,j}) \]
    \[ (AB)_{i,j} = \vee_k (a_{i,k} \land b_{k,j}) \]
- single output bit: on input \( A, B, (i, j) \) output \( (AB)_{i,j} \)

- Boolean Matrix Mult. is in 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 NC^1

- Previous circuit is actually a formula. This is no accident:
  \[ \text{Theorem: } L \in \text{NC}^1 \text{ iff decidable by uniform family of Boolean formulas.} \]
  \[ \text{Proof:} \]
  - \( (\Rightarrow) \) convert NC^1 into formula
    \[ ^{\leftarrow} \]
  - recursively:
    \[ ^{\Rightarrow} \]
  - note: logspace transformation (stack depth \( \log n \), stack record 1 bit - "left" or "right")
**Boolean formulas and NC\(_1\)**

- \((\Leftarrow)\) convert formula of size \(n\) into formula of size \(O(\log n)\)
  - note: size \(\leq 2^{\text{depth}}\), so new formula has \(\text{poly}(n)\) size
  - key transformation:

```
C

D
```

- \(D\) any minimal subtree with size at least \(n/3\) (so size \(D\) \(\leq 2n/3\))
- \(T(n) = \max\) depth of size \(n\) formula
- \(C_2, D\) all size \(\leq 2n/3\)
- \(T(n) \leq T(2n/3) + 3 \Rightarrow T(n) \leq O(\log n)\)

---

**Relation to other classes**

- Clearly \(\text{NC} \subset \text{P}\)
  - recall \(\text{P} = \text{uniform poly-size circuits}\)
- \(\text{NC}_1 \subset \text{L}\)
  - on input \(x\), compose logspace algorithms for: generating \(C_{|x|}\); converting to formula; \(\text{FVAL}\)
- \(\text{NL} \subset \text{NC}_2\): \(\text{S-T-CONN} \in \text{NC}_2\)
  - given \(G = (V, E), s, t\)
  - \(A\) = adjacency matrix w/ self-loops
  - \((A^2)_{ij} = 1\) iff path of length \(\leq 2\) from node \(i\) to node \(j\)
  - \((A^n)_{ij} = 1\) 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?
  \(\text{NC} \not\subset \text{P}\)
- \(\text{P}\)-complete problems least-likely to be parallelizable
  - if \(\text{P}\)-complete problem is in \(\text{NC}\), then \(\text{P} \subseteq \text{NC}\)
  - Why? we use logspace reductions to show problem \(\text{P}\)-complete; \(\text{L} \in \text{NC}\)

- can every uniform, poly size Boolean circuit family be converted into a uniform, poly size Boolean formula family?
  \(\text{NC}_1 \not\subset \text{P}\)

---

**Lower bounds**

- Recall: "\(\text{NP}\) does not have polynomial-size circuits" (\(\text{NP} \not\subset \text{P/poly}\) implies \(\text{P} \neq \text{NP}\)
- major goal: prove lower bounds on (non-uniform) circuit size for problems in \(\text{NP}\)
  - believe exponential
  - super-polynomial enough for \(\text{P} \neq \text{NP}\)
  - best bound known: \(4.5n\)
  - don't even have super-polynomial bounds for problems in \(\text{NEXP}\)
- lots of work on lower bounds for restricted classes of circuits
  - we'll see two such lower bounds: for formulas + monotone circuits
Shannon's counting argument

- amazing/frustrating fact: almost all functions require huge circuits
- \( B(n) = 2^e \approx \# \text{fns. } f: \{0,1\}^n \to \{0,1\} \)
- \# functions computable by circuits with \( n \) inputs + size \( s \):
  - at most \( s \) circuits with \( n \) inputs + size \( s \), which is at most \( s^s \) gates
  \[ C(n, s) = ((n+3)s^2)^s \]
  \( n+3 \) gate types, 2 inputs per gate
- if \( s < 2^n/cn \), \( C(n, s) \ll B(n) \)
- most functions require circuits of size \( \Omega(2^n/n) \)
- same argument: most fns. require formulas of size \( \Omega(2^n/n) \)

Andreev function

- best lower bound for formulas:
- Theorem (Andreev, Hastad 93): the function described below requires \((\land, \lor, \neg)\) formulas of size at least \( \Omega(n^{3-o(1)}) \).

\[ \Omega(n^{3-o(1)}) \]

\[ y \]

\[ \text{selector} \]

\[ \text{XOR} \]

\[ \text{XOR} \]

n-bit string \( y \)

log \( n \) copies; \( n/\log n \) bits each

the Andreev function \( A \)

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
- \( R(n, e_n) = \text{set of restrictions that leave } e_n \text{ variables free} \)
- \( L(f) = \text{smallest } (\land, \lor, \neg) \text{ formula computing } f \) (leaf-size)
- observation:
  \[ E_{\rho \in R(n, e_n)} [L(f_\rho)] \leq \epsilon L(f) \]
  - each leaf survives with probability \( \epsilon \)
- may shrink more...
- propagates constants

Hastad's shrinkage result

- Lemma (Hastad 93): for all \( f \)
  \[ E_{\rho \in R(n, e_n)} [L(f_\rho)] \leq O(\epsilon^{2-o(1)} L(f)) \]

- Proof of theorem:
  - Recall: there exists a function \( h: \{0,1\}^{n \log n} \to \{0,1\} \) s.t. \( L(h) = n/\log n \)
  - hardwire truth table of that function into \( y \) to get \( A^*(x) \)
  - apply random restriction from \( R(n, m = 2\log n \log \log n) \) to \( A^*(x) \)
  - probability given XOR is killed by restriction:
  \[ (1 - 1/\log n)^m \leq 1/\log^2 n \]
  - probability even one of XORs is killed by restriction:
    \[ \log n (1/\log n) = 1/\log n < \frac{1}{2} \]
The lower bound

- Proof of theorem (continued):
  - probability even one of XORs is killed by restriction:
    \[ \log n(1/\log^2 n) = 1/\log n \cdot 1/2. \]
  - with probability at least \( \frac{1}{2} \),
    \[ L(A^*_p) \leq 2 E_{p \leftarrow R(n, m_0)}[L(A^*_p)] \]
  - for some restriction \( p \) all XORs survive and above inequality holds
  - if all XORs survive, can restrict formula further to compute hard function \( h \) (may need to add \( \neg \)'s)
    \[
    L(h) = \frac{n}{\log \log n} L(A^*_p)
    \leq 2E_{p \leftarrow R(n, m_0)}[L(A^*_p)] \leq O((m/n)^{2-o(1)} L(A^*))
    \leq O( (\log \log n/n)^{2-o(1)} L(A^*) )
    
    - implies \( \Omega(n^{3-o(1)}) \leq L(A^*) \leq L(A) \).