Today

- Non-Uniform Computation
  \( P_{\text{poly}} \) (Circuits, Branching Programs, Formulae)
  \( P_{\text{poly}} / \text{poly} = \text{Poly-size Circuits} \)

- Some easy counting based bounds
- Nepiporuk's lower bound on formula size

\[ \text{Notation for } L \subseteq \{0,1\}^* \]

Let \( L_n = L \cap \{0,1\}^n \)

Non-Uniform = Not uniform w.r.t. input length

ie. diff. alg. An for \( L_n \) (for n)
Various Models

Advice Turing Machines: Consider two right
Turing machine, $M$ with advice string
sequence $\bar{a} = (a_1, a_2, a_3, \ldots, a_n, \ldots)$

language accepted by $M$ with advice
sequence $\bar{a}$ is

$L(M, \bar{a}) = \{ x \mid M(x, a_{i_1}, \ldots, a_{i_k}) \text{ accepts} \}$

Resources: In addition to the usual ones, count
$L(n) = \log n$.

if $M, \bar{a}$ runs in time $t(n)$ on first right of
length $n$ then $L(M, \bar{a}) \in \text{TIME}(t(n)/L(n))$
\[ P_{\text{poly}} / \text{poly} = \bigcup_{p(n)} \text{DTIME}(p(n)) \bigcup_{p(n)} \text{poly} \]

Important class.

Similarly, one talks about \( \text{NP}_{\text{poly}} / \text{poly} \), \( \text{L}_{\text{poly}} \), etc.

Circuits (over basis of gates \( g_1, \ldots, g_k \))

(example: binary AND, OR, + unary NOT)

Circuit: DAG with

\[ \text{N sources: input gates: in-degree = 0:} \]

- labelled \( x_1, \ldots, x_n \)

\[ \text{distinct} \]
Remaining vertices: computing gates labeled with basis functions

Not - distinct

In-degree = in-deg of function

Out-degree = arbitrary

Some m of vertices also designated output gates

Computes function mapping \((x_1, \ldots, x_n) \rightarrow (y_1, \ldots, y_m)\)

by letting input vertices be labeled with input values; computing gates be labeled with value of function at input nodes
Circuit C with \( n \) inputs and 1 output. Decide \( L \) if \( \forall x \in L(x) = C(x) \).

**Example**

![Circuit Diagram]

**Measures:**
- **Principal**
- **Size** = \# edges in DAG

Circuit complexity of \( L \): Size of \( C_n \) as function of \( n \).
Depth = longest path in DAG.

1. Circuit complexity changes only by constant factors as basis changes from one finite (computable) set to another.

2. \( L \in \text{DTIME}(t(n)) \cap (l(n)) \)
   \[ \Rightarrow L \text{ has circuit complexity } O((t(n) + l(n)^2)) \]

(Turing Machine tableau, YADA YADA YADA)

3. \( L \) has circuit complexity \( s(n) \)
   \[ \Rightarrow L \in \text{DTIME}(s(n)^2) \cap s(n) \]

(olitto)
Conclusion: $\mathbb{P} / \text{poly} = \text{poly-size circuits}$.

Circuit complexity of function $f = \text{Nonuniform}$

---

What about Nonuniform Space?

Need another model

**Branching Programs** (only defining for decision problems)

BP also given by DAG

- One source = Start node
- Two sinks labelled 0 and 1
- Remaining nodes have out-degree 2 labelled by complementary literals
BP computes function naturally e.g.

Attempts if \( x_1 + x_2 + x_3 = 1 \pmod{3} \)

BP layered if vertices partitioned into layers \( L_1 \ldots L_{k-1} \)

Edges of \( L_i \) go to \( L_i \) or \( L_{i+1} \)

\[
\text{Width (BP)} = \max_i |L_i| \]
Principal measure

\[ \log(\text{width}) = \text{non-uniform space} \]

\[ \times \]

\[ \text{size (BP)} \leq \# \text{nodes} \]

\[ \cot \text{width} \times \]

Finally

\[ \text{Formula} \equiv \text{Circuit in which all } \]

\[ \text{nodes except input have } \]

\[ \text{out-degree } 1 \]

(Why consider: formalization of colloquial term formula)
Some Exercises

1. Formula size $= 2^\mathcal{O}(\text{formula depth})$
   (for finite basis) [trivial]

2. $F$-DEPTH $(f)$ in basis $B,
   \quad = \Theta( F$-DEPTH $(f)$ in basis $B_2 )$

3. For $f : \{0,1\}^n \rightarrow \{0,1\}$
   \[
   F$-depth $(f) = \mathcal{O}( \log (F$-SIZE $(f)) )
   \]
   [little harder]

4. $F$-SIZE $(f)$ in basis $B_1
   \quad \leq ( F$-SIZE $(f)$ in basis $B_2 )^{O(1) }$
Formulae vs. Branching Programs

- $B_{-}\text{size}(f) \leq O(\text{F-size}(f) \text{ in basis } \{\text{AND}, \text{OR}, \text{NOT}\})$  
  (for other bases... I'm not sure)

To Prove $NP \neq P$

suffices to give function in $NP - P/poly$  
(seems more combinatorial)
To prove $P \neq L$

suffices to give function in $P$ that
does not have poly-size B.P.

Unfortunately all the above open.

Best known Ckl- lower bound for function
in $NP$ (in basic AND, OR, NOR) is $\sim 4.5n$.

Best known BP- size lower bound ... in $o(n^2)$.
Counting + Elementary Arguments

1. Every function \( f \) has \( F\)-size in \((\text{AND, OR, NOT basis})\) \( O(2^n) \)

Proof: \( f(x_1, \ldots, x_n) \)

\[
= (x_1 = 0) \land f_0(x_2, \ldots, x_n) \\
\lor (x_1 = 1) \land f_1(x_2, \ldots, x_n)
\]

__________________________________________________________________________

2. \# functions with \( Ckt\)-size \( \leq S \) \( O(S) \)

\( (f\text{-size / BP-size}) \)

3. \( \exists \) functions with \( Ckt\)-size \( \geq \frac{2^n}{n} \)
4. \( \text{Ckt-Size}(S) \subseteq \text{Ckt-Size}(S \cdot \log S) \)

etc.

**Neciporuk's BP lower bound**

Then: \( \exists f \) (cl't distinctness) needing \( n^2 \) BP size.

Idea: Counting + Restriction \( \log^2 n \)

**Proof**

For \( S \subseteq [n] \)

\[ \text{SIZE}_S(BP) = \# \text{ gates with out-edges labelled by cl'ts of } S. \]

If \( S_1, \ldots, S_k = \text{partition of } [n] \)

Then \( \text{SIZE}(BP) \geq \sum_{i=1}^{k} \text{SIZE}_{S_i}(BP) \)
Will Create function

\[ f \left( \frac{x_1}{x_2}, \frac{x_2}{x_3}, \ldots, \frac{x_{k-1}}{x_k} \right) \]

\[ S_1, S_2, \ldots, S_k \]

\[ \text{sat. } H_i : \text{BP-size}_{S_i}(f) = \Omega(ek - kl \log k) \]

Theorem follows.

How to lower bound \( \text{BP-size}_{S_i}(f) \)?

distinguish

1. Count \# a function on \( S_i \) obtained by fixing bits of \( \frac{x_i}{x_j} \) for \( j \neq i \).

2. \( \log \) of above is a l.o.b. on \( \text{BP-size}_{S_i}(f) \)

Why? ----
... Because every such function has

BP of size \( \text{BP-size}\) \( S_i \) (f)

Let \( y(x'\ldots x') = f(x'\ldots x', 0, 1, 1, 0 \ldots) \)

then take \( \text{BP} (f) \)

If \( y = 0 \) drop \( y \) edge & collapse \( a \& c \)

Repeat; Only edges remaining are...
How to create $f$ s.t. $\forall S$:

function obtained by restricting other variables is large?

$f(x_1, \ldots, x_k)$

$= 1$ if $\exists i \neq j \ s.t. \ W_i = W_j$

Now by fixing $W_2, \ldots, W_k$ distinct in $[1, 2^k]$ get $(2^k) \choose k$ distinct functions

$log \left( \binom{2^k}{k} \right) \geq k \log_2 k$ \&