So far we have seen that circuits are quite powerful. In particular, $P = \text{poly}$ contains undecidable problems, and $\text{RP} \subseteq \text{BPP} \subseteq P/\text{poly}$. In this lecture, we will explore this relationship further, proving results that show circuits are very unlikely to be super-powerful compared to uniform complexity classes.

**Theorem 4.1.**

A. (Shannon, 1949) “Most” Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ have circuit complexity $\text{SIZE}(f) \geq \frac{2^n}{n} - \phi_n$, where $\phi_n$ is $o\left(\frac{2^n}{n}\right)$. (More precisely, for any $\epsilon > 0$ this holds for at least a $(1 - \epsilon)$ fraction of all Boolean functions.)

B. (Lupanov, 1965) Every Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed in $\text{SIZE}(f) \leq \frac{2^n}{n} + \theta_n$, where $\theta_n$ is $o\left(\frac{2^n}{n}\right)$.

**Proof.**  
A. The proof is a by a counting argument. Let $B_n = \{f : \{0, 1\}^n \rightarrow \{0, 1\}\}$, that is, the set of all Boolean functions on $n$ bits. $|B_n| = 2^{2^n}$. We will show that the number of circuits of size much smaller than $\frac{2^n}{n}$ is only a negligible fraction of $|B_n|$, proving the claim.

Let us compute the number of circuits of size at most $S \geq n+2$ over $\{\neg, \land, \lor\}$. Note that the argument we present works essentially unchanged for any complete basis of gates for Boolean circuits. What does it take to specify a given circuit? A gate labeled $i$ in the circuit is defined by the labels of its two inputs, $j$ and $k$ ($j = k$ for unary gates), and the operation $g$ the gate performs. The input labels $j$ and $k$ can be any of the $S$ gates or the $n$ inputs or the two constants, 0 and 1. The operation $g$ can be any one of the three Boolean operations in the basis $\{\neg, \land, \lor\}$. Adding to this the name $i$ of the gate, any circuit of size at most $S$ can be specified by a description of length at most $(S + n + 2)^{2S}3^S$. Note, however, that such descriptions are the same up to the $S!$ ways of naming the gates. Hence, the total number of gates of size at most $S$, noting that $S! \geq \frac{(S/\epsilon)^S}{S!}$, is at most

\[
\frac{(S + n + 2)^{2S}3^S}{S!} \leq \frac{(S + n + 2)^{2S}3^S}{\frac{(S + n + 2)^S}{S^S}} = \left(\frac{S + n + 2}{S}\right)^S (3e(S + n + 2))^S \leq \left(e^{\frac{n+2}{S}}3e(S + n + 2)\right)^S \leq (6e^2S)^{S+1}
\]

since $1 + x \leq e^x$. Thus, the number of circuits of size at most $S$ is at most $2^{2^n - o\left(\frac{2^n}{n}\right)}$.
To be able to compute at least an $\epsilon$ fraction of all functions in $\mathbb{B}_n$, we need

\[
(6e^2 S)^{S+1} \geq \epsilon 2^{2^n} \\
\Rightarrow (S + 1) \log_2 (6e^2 S) \geq 2^n - \log_2 (1/\epsilon) \\
\Rightarrow (S + 1)(5.5 + \log_2 S) \geq 2^n - \log_2 (1/\epsilon)
\]

Hence, we must have $S \geq 2^n/n - \phi_n$ where $\phi_n$ is $o(2^n/n)$ to compute at least an $\epsilon$ fraction of all functions in $\mathbb{B}_n$ as long as $\epsilon$ is $2^{-o(2^n)}$. This proves part A of the Theorem.

B. Proof of this part is left as an exercise (see Problem 3, Assignment 1). Note that a Boolean function over $n$ variables can be easily computed in $\text{SIZE}(n2^n)$ by using its canonical DNF or CNF representation. Bringing it down close to $\text{SIZE}(2^n/n)$ is a bit trickier.

\[\square\]

This gives a fairly tight bound on the size needed to compute most Boolean functions over $n$ variables. As a corollary, we get a circuit size hierarchy theorem which is even stronger than the time and space hierarchies we saw earlier; circuits can compute many more functions even when their size is only roughly doubled.

**Corollary 4.2 (Circuit-size Hierarchy).** For any $\epsilon > 0$ and $S_1, S_2 : \mathbb{N} \to \mathbb{N}$, if $n \leq (2 + \epsilon) S_1(n) \leq S_2(n) \ll 2^n/n$, then $\text{SIZE}(S_1(n)) \subset \text{SIZE}(S_2(n))$.

**Proof.** Let $m = m(n)$ be the maximum integer such that $S_2(n) \geq (1 + \epsilon/2) 2^m/m$. By the preconditions of the Corollary, $S_1(n) \leq (1 - \epsilon/2) 2^m/m$ and $m \ll n$. Consider the set $\mathcal{F}$ of all Boolean functions on $n$ variables that depend only on $m$ bits of their inputs. By the previous Theorem, all functions in $\mathcal{F}$ can be computed by circuits of size $2^m/m + o(2^m/m)$ and are therefore in $\text{SIZE}(S_2(n))$. On the other hand, most of the functions in $\mathcal{F}$ cannot be computed by circuits of size $2^m/m - o(2^m/m)$ and are therefore not in $\text{SIZE}(S_1(n))$.  

The following theorem, whose proof we will postpone until the next lecture, shows that circuits can quite efficiently simulate uniform computation. Its corollaries will be useful in several contexts.

**Theorem 4.3 (Pippenger-Fischer, 1979).** If $T(n) \geq n$, then $\text{TIME}(T(n)) \subseteq \bigcup_c \text{SIZE}(cT(n) \log_2 T(n))$.

We now show that although $P/\text{poly}$ contains undecidable problems, it is unlikely to contain even all of $NP$. This implies that circuits, despite having the advantage of being non-uniform, may not be all that powerful. We start with a simple exercise:

**Theorem 4.4 (Karp-Lipton).** If $NP \subseteq P/\text{poly}$, then $\Sigma_2 P \cap \Pi_2 P$.

The original paper by Karp and Lipton credits Sipser with sharpening the result. The proof below which uses the same general ideas in a slightly different way is due to Wilson.

**Proof.** Suppose to the contrary that $NP \subseteq P/\text{poly}$. We'll show that this implies $\Sigma_2 P = \Pi_2 P$. From Lemma 2.6 this will prove the Theorem.

Let $L \in \Pi_2 P$. Therefore there exists a polynomial-time computable set $R$ and a polynomial $p$ such that $L = \{x | \exists y : p(|x|) y \exists z : p(|x|) z. (x, y, z) \in R\}$. The idea behind the proof is as follows. The inner relation in this definition, $\{(x, y) | \exists z : p(|x|) z. (x, y, z) \in R\}$, is an NP language. $NP \subseteq P/\text{poly}$ implies that there exists a
polynomial size circuit family \( \{ C_R \} \) computing this inner relation. We would like to simplify the definition of \( L \) using this circuit family, by
\[
\left\{ x \mid \exists (C_R) \forall y \text{, } C_R \text{ correctly computes } R \text{ on } (x, y) \text{ and } C_R(x, y) = 1 \right\}.
\]

This would put \( L \) in \( \Sigma_2 \text{P} \), except that it is unclear how to efficiently verify that \( C_R \) actually computes the correct inner relation corresponding to \( R \). (Moreover, the whole circuit family may not have a finite specification.)

To handle this issue, we modify the approach and use self-reduction for \( \text{NP} \) to verify correctness of the circuit involved. More precisely, we create a modified version of \( L \) that involves the composition of \( p \) and \( q \). We claim that there exists a polynomial-time algorithm \( M \) that given \( x, y \) and \( C_{\text{pref}}(x) \) as input, either

a. outputs a \( z \) such that \( (x, y, z) \in R \), in which case there exists a \( z \) satisfying this property, or

b. fails, in which case either \( C_{\text{pref}}(x) \) is not a prefix of \( \{ C_n \}_{n=0}^{\infty} \) for computing the \( \text{NP} \) set \( R' \), or no such \( z \) exists.

We prove the claim by describing an algorithm \( M \) that behaves as desired. It will be clear that \( M \) runs in polynomial time.

\[
\text{Algorithm } M: \text{ On input } x, y, C_{\text{pref}}(x), \text{ let } \quad z' \text{ be the empty string}.
\]

\[
\text{If } C_{\text{pref}}(x, y, z') = 0 \text{ then fail}
\]

\[
\text{While } (x, y, z') \notin R \text{ and } |z'| \leq p(|x|)
\]

\[
\text{If } C_{\text{pref}}(x, y, z') = 1
\]

\[
\text{then } z' \leftarrow z'0
\]

\[
\text{else } z' \leftarrow z'1
\]

\[
\text{EndIf}
\]

\[
\text{EndWhile}
\]

\[
\text{If } (x, y, z') \in R
\]

\[
\text{then output } z'
\]

\[
\text{else fail}
\]

\[
\text{EndIf}
\]

Given \( M \) satisfying the conditions of our claim above, we can characterize the language \( L \) as follows:
\[
x \in L \iff \exists z'(|x|) \langle C_{\text{pref}}(x) \rangle \forall y \text{, } M_{\text{decision}}(x, y, \langle C_{\text{pref}}(x) \rangle). \text{ Here } M_{\text{decision}} \text{ denotes the decision version of } M \text{ that outputs true or false rather than } z' \text{ or fail. Since } M \text{ is polynomial-time computable, this shows that } L \in \Sigma_2 \text{P}. \text{ Note that we were able to switch } \exists \text{ and } \forall \text{ quantifiers because } C_{\text{pref}}(x) \text{ doesn’t depend on } y.
This proves that $\Pi_2 P \subseteq \Sigma_2 P$. By the symmetry between $\Sigma_2 P$ and $\Pi_2 P$, this implies $\Sigma_2 P \subseteq \Pi_2 P$, making the two classes identical and finishing the proof.

The following exercise uses the same kind of self reduction that we employed in the above argument:

**Exercise 4.1.** Prove that $NP \subseteq BPP$ implies $NP = RP$.

We now prove that even very low levels of the polynomial time hierarchy cannot be computed by circuits of size $n^k$ for any fixed $k$. This result, unlike our previous Theorem, is unconditional; it does not depend upon our belief that the polynomial hierarchy is unlikely to collapse.

**Theorem 4.5 (Kannan).** For all $k$, $\Sigma_2 P \cap \Pi_2 P \nsubseteq SIZE(n^k)$.

**Proof.** We know that $SIZE(n^k) \subseteq SIZE(n^{k+1})$ by the circuit hierarchy theorem. To prove this Theorem, we will give a problem in $SIZE(n^{k+1})$ and $\Sigma_2 P \cap \Pi_2 P$ that is not in $SIZE(n^k)$.

For each $n$, let $C_n$ be the lexically first circuit on $n$ inputs such that $size(C_n) \geq n^{k+1}$ and $C_n$ is minimal; i.e., $C_n$ is not equivalent to a smaller circuit. (For lexical ordering on circuit encodings, we’ll use $\prec$.) Let $\{C_n\}_{n=0}^\infty$ be the corresponding circuit family and let $A$ be the language decided by this family. By our choice of $C_n$, $A \notin SIZE(n^k)$. Also, by the circuit hierarchy theorem, $size(A)$ is a polynomial $\leq (2 + \epsilon)n^{k+1}$ and the size of its encoding $|A| \leq n^{k+3}$, say. Note that the factor of $(2 + \epsilon)$ is present because there may not be a circuit of size exactly $n^{k+1}$ that computes $A$, but there must be one of size at most roughly twice this much.

**Claim:** $A \in \Sigma_4 P$.

The proof of this claim involves characterizing the set $S$ using a small number of quantifiers. By definition, $x \in A$ if and only if

\[
\exists p(|z|) (C_{|z|}) \cdot \left(\begin{array}{ll}
\forall p(|z|) (C_{|z|}) & size(C_{|z|}) \geq |z|^{k+1} \\
\land & \forall p(|z|) (D_{|z|}) : [size(D_{|z|}) < size(C_{|z|}) \to \exists |z| y. D_{|z|}(y) \neq C_{|z|}(y)] \\
\land & \forall p(|z|) (D_{|z|}) : [((D_{|z|}) \prec (C_{|z|})) \land (size(D_{|z|}) \geq |z|^{k+1})] \to \\
\exists p(|z|) (E_{|z|}) : [size(E_{|z|}) < size(D_{|z|}) \land \forall |z| z. D_{|z|}(z) = E_{|z|}(z)]
\end{array}\right)
\]

The second condition states that the circuit is minimal, i.e., no smaller circuit $D_{|z|}$ computes the same function as $C_{|z|}$. The third condition enforces the lexically-first requirement; i.e., if there is a lexically-earlier circuit $D_{|z|}$ of size at least $|z|^{k+1}$, then $D_{|z|}$ itself is not minimal as evidenced by a smaller circuit $E_{|z|}$. When we convert this formula into prenex form, all quantifiers, being in positive form, do not flip. This gives us that $x \in A$ iff $\exists (C_{|z|}) \forall (D_{|z|}) \exists |z| y \exists (E_{|z|}) \forall |z| z. \phi$ for a certain quantifier free polynomially decidable formula $\phi$. Hence $A \in \Sigma_4 P$.

This proves the claim and implies that $\Sigma_4 P \nsubseteq SIZE(n^k)$. We finish the proof of the Theorem by analyzing two possible scenarios:

a. $NP \subseteq P/poly$. In this case, by the Karp-Lipton Theorem, $A \in \Sigma_4 P \subseteq PH = \Sigma_2 P \cap \Pi_2 P$ because the polynomial time hierarchy collapses, and we are done.

b. $NP \nsubseteq P/poly$. In this simpler case, for some $B \in NP$, $B \notin P/poly$. This implies $B \notin SIZE(n^k)$ and proves, in particular, that $\Sigma_2 P \cap \Pi_2 P \nsubseteq SIZE(n^k)$.

This finishes the proof of the Theorem. We note that unlike the existential argument (the witness is either the language $A$ or the language $B$), one can also define a single language $A'$ witnessing it where $A'$ is a hybrid language between $A$ and a diagonal language in NP.