In circuit lower bounds, our goal is to explicitly exhibit a function that requires large circuits. Even though we believe that NP requires super polynomial size circuits, we do not even know how to show that NP does not have $n^2$ size circuits. In the last lecture, we showed that a larger class $\Sigma_3$ does not have $n^2$-size circuits. Today we will improve upon this result.

We will first show that $\Sigma_2$ does not have $n^2$-size circuits. Finally, we show that $ZPP^{NP}$ does not have a $n^2$ size circuit.

**Theorem 1** (Kannan). There is a language in $\Sigma_2$ that does not have a $n^2$-size circuit.

**Proof.** First consider the case when NP does not have polynomial size circuits. This implies that $\forall k$, SAT does not have $n^k$-size circuits. And as we know that NP $\subseteq \Sigma_2$, so $\Sigma_2$ also does not have any $n^2$-size circuits.

Now consider the case where NP has polynomial size circuits. We will show that in this case $\Sigma_2$ and $\Sigma_3$ are same. Since we already showed that $\Sigma_3$ does not have $n^2$ size circuits, we’ll be done. Now $\Sigma_2 \subseteq \Sigma_3$ from hierarchy. So we need to just show that $\Sigma_3$ is also a subset of $\Sigma_2$. We show this using Karp-Lipton theorem and Claim 1. □

**Theorem 2** (Karp-Lipton Theorem). If NP has polynomial size circuit then $\Sigma_2 = \Pi_2$.

**Proof.** Let $L$ be a language in $\Pi_2$. Then there exist polynomial P and a polynomial time computable relation $R$ such that

- $x \in L \iff \forall y, |y| \leq P(|x|), \exists z, |z| \leq P(|x|), R(x,y,z) = 1$
- $x \notin L \iff \exists y, |y| \leq P(|x|), \forall z, |z| \leq P(|x|), R(x,y,z) = 0$

Let us define a language $L'$ such that

$L' = \{ <x,y> | |y| \leq P(|x|), \exists z, |z| \leq P(|x|), R(x,y,z) = 1 \}$

It can be easily shown that $L' \in NP$. This implies

$x \in L \iff \forall y, |y| \leq P(|x|), <x,y> \in L'$.

Now because $L' \in NP$, it can be reduced to SAT in polynomial time. Let $f$ be a function such that

$f(<x,y>) \in SAT \iff <x,y> \in L'$.

So, we can write

$x \in L \iff \forall y, |y| \leq P(|x|), f(<x,y>) \in SAT$

If NP has a polynomial size circuits (as per the assumption), then there exists circuit family $C$ of size $n^k$ such that

- $\phi \in SAT \implies C(\phi)$ outputs a satisfying assignment
- $\phi \notin SAT \implies C(\phi)$ does not output a satisfying assignment

So, we can write

$x \in L \iff \exists$ Circuit $C, |C| \leq n^k \forall y, C[f(x,y)]$ outputs a satisfying assignment for f(x,y)
So, \( L \in \Sigma_2 \). Then \( \Pi_2 \subseteq \Sigma_2 \). Taking complements, \( \Sigma_2 \subseteq \Pi_2 \). So \( \Pi_2 = \Sigma_2 \). \( \square \)

Note that if we considered circuits which only decides satisfiability instead of outputting an assignment, the proof will not work. This is due to the fact that

**Claim 1.** If \( \Sigma_2 = \Pi_2 \), then \( \Sigma_3 = \Sigma_2 \).

**Proof.** Let language \( L \in \Sigma_3 \). Then \( \exists \) polynomial \( P \) and polynomial type relation \( R \) such that:

\[
x \in L \iff \exists y \ |y| \leq P(|x|), \ \forall z \ |z| \leq P(|x|), \ \exists u \ |u| \leq P(|x|) \ R(x, y, z, u) = 1
\]

Define a language \( L' \) such that

\[
L' = \{ <x, y> | |y| \leq P(|x|), \ \forall z \ |z| \leq P(|x|), \ \exists u \ |u| \leq P(|x|) \ R(x, y, z, u) = 1 \}
\]

Clearly, \( L' \in \Pi_2 \). Since \( \Pi_2 = \Sigma_2 \), \( L' \in \Sigma_2 \). This implies,

\[
<x, y> \in L' \iff \exists z \ |z| \leq q(|<x, y>|) \ \forall u |u| \leq q(|<x, y>|), R'(x, y, z, u) = 1
\]

Finally writing \( L \) in terms of \( L' \)

\[
x \in L \iff \exists y \ |y| \leq P(|x|) <x, y> \in L'
\]

\[
x \in L \iff \exists y \ |y| \leq P(|x|), \ \exists z \ |z| \leq q(|x|), \ \forall u |u| \leq q(|x|) \ R'(x, y, z, u) = 1
\]

We can write

\[
x \in L \iff \exists <y, z> \ |y| \leq P(|x|), \ |z| \leq q(|x|), \ \forall u |u| \leq q(|x|) \ R'(x, y, z, u) = 1
\]

which is clearly in \( \Sigma_2 \) form. Hence proved. \( \square \)

In general, we have the following theorem:

**Theorem 3.** \( \forall k, \exists L \in \Sigma_2 \) such that \( L \) does not have \( n^k \)-size circuits.

Now the question is whether we can still find a class between \( \Sigma_2 \) and \( NP \) which satisfies this. We define below a class \( ZPP^{NP} \) which has this property.

So far we have defined \( \Sigma_2 \) using two ways:

- Quantifier characterization.
- Alternating turing machine.

Now let us define it using Oracle characterization. Let \( L \) be a language in \( NP \).

**Definition 1.** We say \( L' \in NP^L \) if there exists a non deterministic polynomial algorithm (say \( N \)) that accepts \( L' \). \( N \) during its computation can ask questions like \( Does \ y \in L. \)

**Theorem 4.** \( \Sigma_2 = NP^{NP} \)

**Proof.** We will prove that \( \Sigma_2 \subseteq NP^{NP} \). Consider a language \( L \in \Sigma_2 \), i.e., \( x \in L \iff \exists y \forall z R(x, y, z) = 1 \). Now define a language \( L' \) such that \( L' = \{ <x, y> | \exists z R(x, y, z) = 0 \} \). Clearly \( L' \in NP \). Then we can write \( L \) in term of \( L' \),

\[
x \in L \iff \exists <y, z> \notin L'
\]

Let us consider the following oracle machine for \( L \).

1. Input \( x \).
2. Guess \( y \).
3. Ask if \( <x, y> \in L' \)
(4) If NO, then accept. If YES, then reject.

Since $L' \in \text{NP}$ and above is a NP machine, $L \in NP^{NP}$, which implies $\Sigma_2 \subseteq NP^{NP}$. We omit the proof of $NP^{NP} \subseteq \Sigma_2$. □

Note that since any problem in NP is reducible to SAT we can use SAT as oracle in place of NP. As a side note, we know that $\overline{SAT} \in \text{NP}^{NP}$, in fact we know that $\overline{SAT} \in \text{P}^{NP}$. But we do not know if it belongs to NP.

**Definition 2.** A language $L \in \text{ZPP}$ if the following holds:

\[
\begin{align*}
x \in L & \implies Pr[A(x) \text{ accepts}] \geq 2/3 \\
Pr[A(x) \text{ rejects}] & = 0 \\
x \notin L & \implies Pr[A(x) \text{ rejects}] \geq 2/3 \\
Pr[A(x) \text{ accepts}] & = 0
\end{align*}
\]

We know that $\text{P} \subseteq \text{ZPP} \subseteq \text{NP}$ and we can show easily $\text{P}^{NP} \subseteq \text{ZPP}^{NP} \subseteq \text{NP}^{NP}$. We have the following improvement in circuit lower bound.

**Theorem 5.** $\text{ZPP}^{NP}$ does not have $n^2$-size circuits.

**Proof.** First, if NP does not have polynomial size circuits, we are done as $\text{NP} \subseteq \text{ZPP}^{NP}$.

Now consider the case that NP has polynomial size circuits, which implies, SAT has polynomial size circuits, i.e., $\exists k$ such that $\overline{SAT} \in \text{SIZE}(n^k)$. Again we can assume that the circuit for SAT outputs a satisfying assignment whenever the input formula is satisfiable. From now we will assume that all our circuits have this property: When they say a formula is satisfiable, they output a satisfying assignment. From this it follows that the following language $L_1$ is in NP.

\[
\{ C \mid C \text{ is a wrong circuit for SAT} \}.
\]

Given a circuit $C$ let $S_C$ be the unique string that encode the circuit $C$. Consider a language $L_2$ that consists of tuples of following form: $(x, i, \phi_1, \ldots, \phi_\ell)$. Such a tuple belongs to the set, if 1) All formulas are satisfiable, 2) there is a circuit $C$ of size at most $n^k$ such that $S_C$ is one bit extension of $x$ and the $i$the bit of $S_C$ is one, and 3) $C$ is correct all formulas $\phi_i$, $1 \leq i \leq \ell$. Observe that this language is also in NP.

**Claim 2.** There is a $\text{ZPP}^{NP}$ algorithm $A$ such that $A(1^n)$ outputs a circuit that correctly solves SAT on the inputs of length $n$.

Fix $n$. $T_0 = \text{all satisfiable formulas of length } n$. Then consider the following ZPP$^{NP}$ algorithm $A$:

1. Input $1^n$.
2. For $i = 0$ to $n$
   (a) Uniformly at random pick $\phi_1, \phi_2 \ldots, \phi_i$ from $T_i$.
   (b) By making queries to $L_2$ obtain a circuit of size $\leq n^k$ that is correct on all of $\phi_1, \phi_2 \ldots, \phi_i$. Let us call it $C_i$.
   (c) $T_{i+1} = \text{All satisfiable formulas on which all of the } C_0, C_1, C_2 \ldots C_i \text{ are wrong}$.
3. Set $C = C_0 \lor C_1 \lor C_2 \lor \ldots \lor C_n$.
4. Ask the NP language $L_1$, if $C$ is a correct circuit for SAT. If so, output $C$, if not output “?”. 

3
Observe that the above algorithm outputs a circuit $C$, only when it correctly solves SAT. Thus it never outputs a wrong answer.

How does the sizes of $T_0$ and $T_1$ compare?

**Claim 3.**

$$\Pr[|T_0| \geq \frac{1}{2}|T_1|] \leq \text{small.}$$

Call a circuit $C$ bad if it is wrong on more than $1/2$ fraction of inputs from $T_0$. Consider the iteration when $i$ equals zero. Here, we uniformly at random pick $\ell$ formulas. What is the probability there is a bad circuit $D$ that is correct on all of these formulas? This is at most $\sum_{C \text{ is bad}} \Pr[D = C]$. If $C$ is bad, then with probability at least $1 - 1/2^\ell$, we would have picked a formula such that $C$ is wrong on that formula. Thus the probability that there is a bad circuit is at most $2^{n^k^2}/2^\ell$, as there are at most $2^{n^k^2}$ bad circuits of size $n^k$. If we choose $\ell$ to $n^k^3$, this probability is very small. Thus the probability that $C_1$ is a bad circuit is very small. Thus the size $T_1$ is at most half the size of $T_0$ with high probability.

Similarly, we can show that for every $i$, with high probability the size of $T_{i+1}$ is at most half the size of $T_i$. Thus for every formula, there exist a circuit $C_i$, $1 \leq i \leq n$, such that $C_i$ is correct on that formula. Recall that this means, $C_i$ outputs a satisfying assignment, whenever it says a formula is satisfiable. Thus the final circuit $C$ correctly solves SAT.

Finally, to show that the above algorithm can be implemented by a ZPP$_{NP}$-machine, we have to show that one can pick formulas uniformly at random from $T_i$. Observe that each $T_i$ is a set in NP. It is known that, there is a probabilistic polynomial time algorithm, with an NP-oracle, that can (almost) uniformly generate elements from a set NP. We omit the proof.

□