We defined circuits in the last class and will spend the next two classes studying them. Circuits are an attractive area of study because they provide a level of structure which might be useful for proving that P and NP are separated. Recall the following definitions.

**Definition 1** \( \text{size}(T(n)) \) is the class of languages decidable by circuit families of size \( O(T(n)) \).

**Definition 2** \( \text{P/poly} = \bigcup_{c \in \mathbb{N}} \text{size}(n^c) \)

There are two interesting alternative definitions of \( \text{P/poly} \), as specified by the kinds of Turing machines that decide languages in the class:

**Definition 3** A nonuniform polynomial time Turing machine is a sequence \( \{M_n\}_{n \in \mathbb{N}} \) of Turing machines for which there exist a polynomial \( p \) such that \( |M_n| \leq p(n) \) and \( M_n \) runs in time \( p(n) \) on an input of length \( n \).

**Definition 4** A polynomial time Turing machine with advice is a Turing machine \( M \) and advice sequence \( \{a_n\}_{n \in \mathbb{N}} \) for which there exists a polynomial \( p \) such that \( M(x, a_n) \) runs in time \( p(n) \) on inputs \( x \) of length \( n \).

We are interested in \( \text{P/poly} \) because it is known that \( \text{P} \subseteq \text{P/poly} \) and believed that \( \text{P/poly} \subset \text{NP} \), and thus could potentially be used to find a separation between P and NP. We leave the former claim as an exercise and will provide evidence for the latter claim. Namely, if \( \text{NP} \subseteq \text{P/poly} \), then the polynomial hierarchy collapses, which would indeed by a surprising turn of events.

**Theorem 1** If \( \text{NP} \subseteq \text{P/poly} \), then \( \text{PH} = \Sigma_2 \).

**Proof.** In particular, we will show that \( \text{NP} \subseteq \text{P/poly} \implies \Pi_2 \subseteq \Sigma_2 \), which we know implies \( \Pi_2 = \Sigma_2 \) and in turn that \( \text{PH} = \Sigma_2 \). Consider a polynomial time relation \( R \) and a \( \Pi_2 \) language \( L \) defined as:

\[
x \in L \iff \forall y_1 \exists y_2 . R(x, y_1, y_2).
\]

7-1
We want to show that $L$ is in $\mathbf{\Sigma}_2$. Since we are assuming $\mathbf{NP} \subseteq \mathbf{P/poly}$, we can construct a polynomial size circuit that decides SAT problems. Since finding SAT assignments can be done in polynomial time given a SAT oracle, there exists a polynomial $q$ and $q(n)$-sized circuit family $\{C_n\}_{n \in \mathbb{N}}$ that finds satisfying assignments to SAT problems. The trick is then to replace the inner existential quantification with the use of this circuit. Consider the following language $L'$:

$$x \in L' \iff \exists C \forall y_1 : |C| \leq q(x) \land R(x, y_1, C(x, y_1)).$$

Since the inner predicate checks that the circuit $C$ is of polynomial size in $x$, it can be decided in polynomial time, so that the language $L'$ is in $\mathbf{\Sigma}_2$. Now we show $L = L'$.

Suppose $x \in L$. Since $R$ can be decided in time polynomial in $x$, it can be encoded into a polynomial size SAT problem (as in Carp’s theorem). Then choose $C$ to be a circuit as above that computes a satisfying assignment $y_2$ for $R(x, y_1, y_2)$ given $x$ and $y_1$. $C$ is size polynomial in $x$, so that the predicate in the expression defining $L'$ is satisfied.

Conversely, suppose $x \in L'$. The inner existential quantification in the definition of $L$ would be satisfied by choosing $y_2 = C(x, y_1)$, so that $x \in L$.

Thus, we have shown that every language $L \in \mathbf{\Pi}_2$ is in $\mathbf{\Sigma}_2$, so the polynomial hierarchy collapses.

This proof provides strong evidence that $\mathbf{P/poly}$ is a class that can distinguish $\mathbf{P}$ from $\mathbf{NP}$. To do this, we would need to, for example, demonstrate that $\mathbf{SAT} \notin \mathbf{P/poly}$, which means that we are interested in establishing lower bounds on the size of circuits necessary for solving various problems.

**Theorem 2** Consider the class of functions $\{T^k_n\}$ on binary tuples $x$ defined by $T^k_n(x_1, ..., x_n) = 1$ iff $\sum x_i \geq k$. For $2 \leq k \leq n - 1$, a boolean circuit of fan-in at most 2 that computes $T^k_n$ must be of size at least $2n - 4$.

**Proof.** We will in all cases consider, without loss of generality, the smallest circuit that computes this function. This implies that there would be no circuit elements of fan-in 1 (e.g., unary negation) because one could create a smaller circuit by embedding this gate’s functionality into downstream gates.

We proceed by induction on $n$. Consider the base case $n = 3 \implies k = 2$. Since every input must be examined to compute the function and each gate can examine only 2 inputs, there must be at least 2 gates in the network, which exactly meets the bound.

As our induction hypothesis, suppose that the theorem holds for

$$\{(n, k) \mid 3 \leq i < n \text{ and } 2 \leq k \leq i - 1\}.$$
Consider a circuit $C$ for computing $T_n^k$. Let $G$ be a gate that takes two variables $x_i$ and $x_j$ as inputs for $i \neq j$ (recall that we have excluded unary gates). After fixing values for $x_i$ and $x_j$, the function $T_n^k$ restricted to the remaining $n-2$ variables will take on at least 3 different forms.

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$x_j$</th>
<th>restricted form of $T_n^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$T_n^{k-2}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$T_n^{k-1}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$T_n^{k-1}$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$T_n^{k-2}$</td>
</tr>
</tbody>
</table>

Note that this is where the restriction $k < n$ becomes important. If $k = n$, then the restricted function takes only two forms, namely identically 0 or 1 if all its inputs are 1.

Since the restricted circuit assumes three different behaviors, however $G$ only outputs a single boolean value, there must be a connection from either $x_i$ or $x_j$ to another gate $G'$ in the circuit. Assume, without loss of generality, that the connection is from $x_i$. Now let $C_\alpha$ denote the circuit obtained by setting $x_i = \alpha$. Clearly, $C_\alpha$ computes $T_{n-1}^{k-\alpha}$. When constructing $C_\alpha$ we observe that $G$ and $G'$ are now unary gates, and so can be removed, so that $|C_\alpha| + 2 \leq |C|$. If $k = 2$, set $\alpha = 0$, other $\alpha = 1$ (so that $2 \leq k - \alpha \leq n - 2$). By the induction hypothesis, $|C_\alpha| \geq 2(n-1) - 4 \implies |C| \geq 2n - 4$.

Recall that a formula is a circuit where the fan-out of every node is 1. The circuit of a boolean formula is then just a binary tree.

**Theorem 3** Consider the following function $f_n$ defined on tuples $\vec{x}$ of bit-strings of length $2\log(n)$:

$$f_n(x_1, ..., x_n) = 1 \text{ iff } \exists i \neq j \text{ s.t. } x_i = x_j.$$  

Any formula computing $f_n$ must be $\Omega(n^2)$.

**Proof.**

Since the circuits for formulas are trees, some input variables must be read in multiple times if their values must be processed by multiple gates. Let $k$ be the total number of leaves (i.e., inputs) carrying bits from $x_i$, for some $i$, in a formula computing $f_n$. For some assignment $\vec{\alpha} = (\alpha_1, ..., \alpha_n)$ to all the bits in $\vec{x}$, consider $f_n^{\vec{\alpha}^{-1}}(x_i) = f_n(\alpha_1, ..., x_i, ..., \alpha_n)$.

After doing this restriction, the formula computing $f_n$ can be reduced to a formula with $k$ leaves that computes $f_n^{\vec{\alpha}^{-1}}$ by eliminating unary gates. Thus the number of functions that $f_n^{\vec{\alpha}^{-1}}$ can identically equal as $\alpha$ ranges over all possible values is bounded by the number of formulas of with $k$ leaves. We count these.
# formulas on $k$ leaves $\leq$ # formulas of size $2k$ \hspace{2cm} (1)
$\leq$ (# binary trees of size $2k$) · (# ways to assign gates) \hspace{2cm} (2)
$\leq$ $4^{2k} \cdot g^{2k}$ \hspace{2cm} (3)
$= 2^{O(k)}$ \hspace{2cm} (4)
(5)

Each $x_i$ takes on $2^{2\log(n)} = n^2$ values. This means that, for various choices of $\alpha$, the restricted function $f_{n, \vec{\alpha}}$ assumes at least $\binom{n^2}{n-1}$ functional forms (it is also sometimes identically 1). Therefore, it is necessary that
\[
\binom{n^2}{n-1} \leq 2^{O(k)} \implies 2^n \leq 2^{O(k)} \implies k = \Omega(n).
\]

Since the inputs in $x_i$ need to be replicated $\Omega(n)$ times and this holds for each $i$, any circuit computing $f_n$ must be of size $\Omega(n^2)$. 

7-4