1 Circuits

A circuit is a non-uniform model of computation, with a fixed number of bits. Formally, an $n$-bit circuit $C$ is given by a DAG with $n$ inputs, one output, and where nodes correspond to basic gates (say, AND, OR, NOT). We denote $C(x)$ the value that an input $x \in \{0,1\}^n$ evaluate to when run through $C$. The size of a circuit is the number of wires it has.

**Definition 1.1 (Size complexity).** A language $L \subset \{0,1\}^*$ is in $\text{SIZE}(S(n))$ if for any $n \in \mathbb{N}$ there exists a circuit $C_n$ of size $|C_n| \leq S(n)$ such that

$$\forall x \in \{0,1\}^n, \; x \in L \iff C_n(x) = 1.$$

For example, the language $L = \{1^n : n \in \mathbb{N}\}$ can be decided by a linear size circuit (computing the AND function).

**Definition 1.2 (P/poly).** $\text{P/poly} = \bigcup_{c \geq 1} \text{SIZE}(n^c)$.

2 Uniform vs nonuniform polynomial time

**Theorem 2.1.** $\text{P} \subset \text{P/poly}$.

**Proof.** This is very similar to the Cook-Levin theorem. Let $L \in \text{P}$ be computed by a Turing machine $M$. Assume $M$ runs in time $n^c$. For a fixed input length $n$, let $N = n^c$ be a bound on the space and time used by $M$. We can represent the computation of $M$ by $O(N^2)$ bits, representing the configuration of $M$ in every step. Note that the configuration in step $i + 1$ can be computed by a linear-size circuit from the configuration in step $i$. Hence, the entire computation can be computed by a circuit of size $O(N^2)$.

**Theorem 2.2.** There are undecidable languages in $\text{P/poly}$.

**Proof.** Let $L$ be any unary language, e.g. with only strings of the form $1^n$. Then clearly $L \in \text{P/poly}$ since for every input length there is at most one element in $L$. However, $L$ can be used to encode undecidable languages, for example $L = \{1^n : n = \langle M \rangle$ and $M$ halts on the empty input$\}$. 

1
3 Uniform circuits

Definition 3.1. A circuit class \( \{C_n : n \in \mathbb{N}\} \) is P-uniform if there exists a polynomial time TM which outputs \( C_n \) on input \( 1^n \).

Theorem 3.2. \( L \) is computable by a P-uniform circuit class iff \( L \in \mathcal{P} \).

Proof. If \( L \) is computable by a P-uniform circuit class, then there is a Turing machine \( M \) such that \( M(1^n) = C_n \) and \( C_n(x) = 1 \iff x \in L \) for \( x \in \{0,1\}^n \). Let \( U \) be a simulator so that \( U(C_n, x) = C_n(x) \). Then \( x \in L \iff U(M(|x|), x) = 1 \) and hence \( L \in \mathcal{P} \). For the other direction, if \( L \in \mathcal{P} \), then one can verify that the poly-size circuit one gets for inputs of length \( n \) in \( L \) (that we showed in the proof of \( \mathcal{P} \subset \mathcal{P}/\text{poly} \)) can be computed by a poly-time machine given as input the input length \( 1^n \).

4 Turing machines with advice

Definition 4.1. A language \( L \) is computed by a deterministic Turing machine in time \( T(n) \) with \( a(n) \) bits of advice, denoted \( L \in \text{TIME}(T(n))/a(n) \), if there exists a Turing machine \( M \) running in time \( T(n) \), and for every input length \( n \) there exists a string \( \alpha_n \in \{0,1\}^{a(n)} \), such that \( x \in L \iff M(x, \alpha_n) = 1 \).

Theorem 4.2. \( \mathcal{P}/\text{poly} = \bigcup_{c,d \geq 1} \text{TIME}(n^c)/n^d \).

Proof. If \( L \in \mathcal{P}/\text{poly} \), then the advice for length \( n \) is the circuit deciding the language, and the Turing machine evaluates an input in the circuit. If \( L \in \text{TIME}(n^c)/n^d \), then the circuit computes \( M(x, \alpha_n) \).

5 Boolean circuits and higher complexity classes

We believe that \( \mathcal{N} \not\subset \mathcal{P}/\text{poly} \). The following is a partial witness to that.

Theorem 5.1 (Karp-Lipton). If \( \mathcal{N} \subset \mathcal{P}/\text{poly} \) then \( \mathcal{PH} = \Sigma_2 \).

We would require the following claim first.

Claim 5.2. Assume that \( \mathcal{N} \subset \mathcal{P}/\text{poly} \). Then for every \( n \), there exists a circuit \( C_n \) as follows. It takes as input a 3-CNF formula \( \varphi \) on \( n \) variables, and if \( \varphi \) is satisfiable then it outputs a satisfying assignment to \( \varphi \). That is,

\[
\varphi \in 3-SAT \iff \varphi(C(\varphi)) = 1.
\]

Proof. If \( \mathcal{N} \subset \mathcal{P}/\text{poly} \) then there is a circuit \( C'_n \) such that \( C'_n(\phi) = 1 \iff \phi \) is satisfiable. We would use \( C'_n \) to discover a satisfying assignment. We find a satisfying assignment by
discovering its bits $a_1, \ldots, a_n$ iteratively. Let us denote by $\phi_k(a_1, \ldots, a_k, x_{k+1}, \ldots, x_n)$ the 3-CNF $\phi$ when we plug in $x_1 = a_1, \ldots, x_k = a_k$. Define
\[ a_1 = \begin{cases} 0 & \text{if } C'(\phi(0, x_2, \ldots, x_n)) = 1 \\ 1 & \text{otherwise} \end{cases} \]
and
\[ a_i = \begin{cases} 0 & \text{if } C'(\phi(a_1, \ldots, a_{i-1}, 0, x_2, \ldots, x_n)) = 1 \\ 1 & \text{otherwise} \end{cases} \]
By construction, if $\phi$ is satisfiable then $(a_1, \ldots, a_n)$ is a satisfying assignment. We can build a circuit computing $a_1, \ldots, a_n$ of size $n \cdot |C'|$.

Proof of Karp-Lipton theorem. To prove that $PH = \Sigma_2$ it suffices to prove that $\Pi_2 \subseteq \Sigma_2$. A complete problem form $\Pi_2$ is $\Pi_2 SAT = \{ \phi : \forall u \in \{0, 1\}^n \exists v \in \{0, 1\}^n \phi(u, v) = 1 \}$, where $\phi$ is a 3-CNF. By our assumption, there is a circuit $C_n$ such that for every $u$, $\phi(u, \cdot)$ is satisfiable iff $\phi(u, C_n(\phi, u)) = 1$. Hence
\[ \phi \in \Pi_2 SAT \iff \forall u \in \{0, 1\}^n \phi(u, C_n(\phi, u)) = 1. \]
The only problem is how do we know $C_n$? well, we can guess it.
\[ \phi \in \Pi_2 SAT \iff \exists C_n \forall u \in \{0, 1\}^n \phi(u, C_n(\phi, u)) = 1. \]

Theorem 5.3 (Meyer’s theorem). If $EXP \subseteq P/poly$ then $EXP = \Sigma_2$.

Proof. Let $L \in EXP$. Let $M$ a Turing machine running in time $N = 2^n$ computing $L$. Consider the following language
\[ L_M = \{(x, i, j) : i, j \leq 2^{|x|}, \text{the j-th bit of the i-th configuration of running M on x is 1 and M is in state} \}
\]
We also have that $L_M \in EXP$. Now, if $EXP \subset P/poly$ then for every input length $|x| = n$ there exists a circuit $C_n$ of size $n^{O(1)}$ such that
\[ (x, i, j) \in L_M \iff C_n(x, i, j). \]
In particular let $STATE(x, i)$ denote the bits representing the state of $M(x)$ in step $i$ (say, the first few bits of the configuration). The only question is how do we find $C_n$, and how do we verify it gives the correct values. The answer is that verification is local. That is, we can verify correctness by
\[ \forall i, j, VERIFY(C_n(x, i, j), \{C_n(x, i-1, j+a) : a = -1, 0, 1\}, STATE(x, i), STATE(x, i-1)). \]
So we get that
\[ x \in L_M \iff \exists C_n \forall i, j \in \{0, 1\}^n VERIFY(C_n, x, i, j). \]
6 Hard functions

It is not hard to show that hard functions exist. What we don’t know is how to prove for specific explicit functions that they are hard.

**Theorem 6.1.** For every \( n \geq 1 \) there exists a function \( f: \{0, 1\}^n \to \{0, 1\}^n \) which requires circuits of size \( \Omega(2^n / n) \).

**Proof.** This is by a counting argument. The number of all functions is \( 2^{2^n} \). To describe a circuit of size \( S \), we need to specify all the wires. Each wire takes \( O(\log S) \) bits, hence in total we need \( O(S \log S) \) bits. As long as \( 2^{O(S \log S)} < 2^{2^n} \) there are functions which require size \( S \). This holds as long as \( S < O(2^n / n) \).

In fact, if we choose a random function (by choosing \( f(x) \) uniformly and independently for any input) then we get that with very high probability it requires size \( \Omega(2^n / n) \). We can also deduce a size hierarchy theorem.

**Corollary 6.2.** If \( T(n) < O(2^n / n) \) then \( \text{SIZE}(T(n) / \log(T(n))) \subsetneq \text{SIZE}(T(n) \cdot \log(T(n))) \).

**Proof.** Set \( \ell \) so that \( 2^\ell = T(n) \). We shown that there is a function on \( \ell \) bits which cannot be computed in size \( O(2^\ell / \ell) \). However, any function on \( \ell \) bits can computed by a CNF of size \( O(2^\ell) \).