1 Definitions and Prerequisites

1.1 Review of Polynomial Hierarchy

$NP$ - languages with a polynomial-sized witness which can be easily verified.

$$L \in NP \iff \exists \text{ poly } p(\cdot), R \in P \text{ s.t. } (x \in L \iff \exists y R(x, y), |y| \leq p(|x|))$$

We can extend this idea to define the $k$th level of polynomial hierarchy. The intuitive way to describe it in terms of witnesses is that there are two players arguing whether the relation holds. The first person provides a witness (just like in $NP$), but now the second person is allowed to try and refute that witness (this would be the 2nd level), and we can continue and allow the first player to refute the refutation and so on to obtain the other levels. Finally we can think of a "referee" which checks whether the polynomial relation holds and thus "which player won." Formally,

$$L \in \Sigma_k^P \iff \exists \text{ poly } p(\cdot), R \in P \text{ s.t. } (x \in L \iff \exists y_1 \forall z_2 \ldots \exists y_k R(x, y_1, z_2, \ldots, z_k), |y_i|, |z_i| \leq p(|x|))$$

The union of all levels is known as the polynomial hierarchy:

$$\text{PH} = \bigcup_k \Sigma_k^P$$

Some known facts are that $P = NP$ implies that the polynomial hierarchy "collapses," i.e. $P = \text{PH}$. If for any $i$ it happens that $\Sigma_i^P = \Sigma_{i+1}^P$, then the polynomial hierarchy collapses down to the $i$th level, i.e. $\text{PH} = \Sigma_i^P$.

1.2 Exponential Hierarchy

We can define a similar type of hierarchy at the exponential level. We begin by defining the exponential equivalent of $NP$, non-deterministic exponential time ($\text{NEXP}$). $\text{NEXP}$ consists of those languages with a witness of exponential length, where we can decide (in polynomial time) whether it really is a witness. Notice that the relation is kept as a polynomial time one, but it operates on an input (witness) that is already exponential in the length of the original input. Formally,

$$L \in \text{NEXP} \iff \exists \text{ poly } p(\cdot), R \in P \text{ s.t. } (x \in L \iff \exists y R(x, y), |y| \leq 2^{p(|x|)})$$

Equivalently we can define the $k$th level of the exponential hierarchy as

$$L \in \Sigma_k^{\text{exp}} \iff \exists \text{ poly } p(\cdot), R \in P \text{ s.t. } (x \in L \iff \exists y_1 \forall z_2 \ldots \exists y_k R(x, y_1, z_2, \ldots, z_k), |y_i|, |z_i| \leq 2^{p(|x|)})$$

Even though the definition of the exponential hierarchy has so much resemblance with the definition of the polynomial hierarchy, we don't know that the exponential hierarchy is structured as nicely as the polynomial one. For instance, it is not known that if $\text{NEXP} = \text{EXP}$, the exponential hierarchy collapses (and even we know that there are oracles relative to which this statement is false).
In these definitions we bound the size of the witness by an exponential function of the input size. Similarly, we can define these classes with a bound on the witness size that is exponential but with only a linear exponent, resulting in classes $\Sigma_k^n$.

For any language $L$, we can define the language $1^L$, as $1^L = \{1^x \mid x \in L\}$, where by $1^x$ we denote the integer described by the string $x$. This helps us formalize some relation between the polynomial and the exponential hierarchy. By a simple padding argument we have

$$L \in \Sigma_k^n \iff 1^L \in \Sigma_k^n$$

Similar reasoning applies to EXP as well, so if $1^L_i = \{1^{x^i} \mid x \in L\}$, we have

$$L \in \Sigma_{k,\exp}^n \iff \exists i \text{ s.t. } 1^{L_i} \in \Sigma_k^n$$

From these we can also observe this (one-way) chain of implications:

$$\Sigma_i^n = \Sigma_j^n \Rightarrow \Sigma_i^e = \Sigma_j^e \Rightarrow \Sigma_i^{\exp} = \Sigma_j^{\exp}$$

## 2 Today’s results

**Theorem 2.1** (Kannan). $\exists f \in \Sigma_3^n$, such that $\text{SIZE}(f_n) \geq \Omega(2^n/n)$, where by $\text{SIZE}(f_n)$ we denote the minimum circuit size for $f$ on inputs restricted to length $n$.

We will also show some corollaries of this theorem,

**Theorem 2.2** (Karp-Lipton). If $\text{NP} \subseteq \text{P/poly}$, then $\text{PH} = \Sigma_2^n$.

**Corollary 2.3.** $\Sigma_2^n \not\subseteq \text{P/poly}$

**Corollary 2.4.** $\forall i, \Sigma_2^n \not\subseteq \text{SIZE}(n^i)$ proving this is left as a homework exercise

## 3 Proof of Kannan’s theorem

This is the main goal for today. Proving this result effectively puts "hard" functions in $\Sigma_3^n$. To see how "hard" they are, note that every function on $n$ inputs has a circuit of size $\text{SIZE}(2^n/n)$. Therefore, what Kannan’s theorem is saying is that in $\Sigma_3^n$ we can find a function that has, to within a gate, the maximum circuit complexity of any function.

To prove Kannan’s theorem, we will first show existence of a function $f$ that has the required high circuit complexity, and then we will place it in $\Sigma_3^n$. So, we show the following theorem first, due to Riordan and Shannon, which takes care of that first step, i.e. shows the existence of such $f$ without putting any constraints on it in terms of complexity classes:

**Theorem 3.1** (Riordan-Shannon). $\exists f, \text{ such that } \text{SIZE}(f_n) \geq \Omega(2^n/n)$

**Proof.** The proof is by a counting argument; we will show that the number of different functions is much greater than the number of different circuits of size smaller than $2^n/n$.

Let $g_1, g_2, \ldots, g_s$ be the gates in a circuit, where $s$ is the size of the circuit. Then, each $g_i$ can be written as some $\text{op}_i(\alpha, \beta)$, where $\text{op}_i$ is some boolean operation, and $\alpha, \beta$ are either $g_j, g_k$ where $j, k < i$ or one of the inputs $x_1, x_2, \ldots, x_n$. Let $c_2$ be the number of distinct boolean functions on 2 1-bit inputs (note that $c_2$ is a constant). Then, the number of possible different gates for each $g_i$ is upper bound by $\# g \leq c_2(s + n)^2$ ($c_2$ choices for the operation and $s + n$ choices for each operand). Since we are looking at circuits of size $s$, the total number of possible different circuits is then $\# C \leq (\# g)^s \leq (c_2(s + n)^2)^s$. Assuming $s \geq n \geq 4c_2$, we
get \( \#C \leq s^{3n} \).

On the other hand, the number of different boolean functions on \( n \) inputs is \( 2^{2^n} \). Then, if every boolean function can be described by a circuit of size \( s \) \((\forall f : \{0,1\}^n \rightarrow \{0,1\}, \text{SIZE}(f) \leq s)\), we have \( 2^{2^n} \leq s^{3n} \).

By quick case analysis: if \( s \geq 2^n \), then certainly \( s \geq 2^n/3n \). On the other hand, if \( s < 2^n \), then \( 2^{2^n} \leq 2^{3sn} \), so again \( s \geq 2^n/3n \). With a slightly sharper analysis, we can lose the 3 factor and get the tight bound \( 2^{n/3} \).

Let \( s_n = \max_f(\text{SIZE}(f)) \). From the previous result we know that \( s_n \geq 2^n/n \). Trying to find such function \( f \), we could write something like \( f \) has size \( s_n \Leftrightarrow \forall C, |C| < s_n, f_C \neq f \), where \( f_C \) is the function computed by \( C \). If we think of \( f \) as described by a \( 2^n \)-size truth table, then checking whether \( C \) computes \( f \) is "polynomial" in size of the input \( (f) \), as it would just check if for all inputs the circuit matches \( f \). In other words, saying that a function is hard is in \( \text{coNEXP} \), as it suffices to produce the \( C \) of size smaller than \( s_n \) which computes \( f \). However, this definition will define a class of functions, and to complete the proof we need to find a single hard function. So instead, we look at the lexicographically first hard function. We define it as follows:

\[
x \in L_n \Leftrightarrow \exists f \text{ s.t. } \forall C_1, |C_1| < s_n, \forall g \text{ before } f \text{ in the ordering, } \exists C_2, |C_2| < s_n \text{ s.t. } f_{C_2} = g, f_C \neq f, f(x) = 1
\]

Notice the structure of the RHS - it has the right quantifiers for \( \Sigma^p_3 \). You can think of it as the 2-player argument game as before - the first person guesses \( f \) and claims that it is the first lexicographically hard function on \( n \) inputs. The second player is trying to refute him. He can just perform a lookup and show that \( f(x) = 0 \). Otherwise, he can try showing that \( f \) is not actually a hard function by producing a \( C_1 \) such that \( f_{C_1} = f \), or finally he can show that it is not the lexicographically first hard function, by showing that there is a function \( g \) that comes before \( f \) in the lexicographic ordering that is hard, in which case the first player gets a chance to refute him by producing a \( C_2 \) such that \( f_{C_2} = g \). Therefore, we have constructed a language \( L_n \) such that \( L_n \in \Sigma^p_3 \) and \( \text{SIZE}(L_n) \geq \Omega(2^n/n) \), concluding the proof of Kannan’s theorem, i.e. that in \( \Sigma^p_3 \) there is a function with the maximum possible circuit complexity.

Kabanets-Rackoff program: \( \text{aside} \)

\[
\forall f \in E, \text{SIZE}(f_n) \leq s_n - 1 \Rightarrow P \neq NP
\]

In other words, if we could show that there is a way to save just a single gate for all exponential functions, we would have proven \( P \neq NP \). To prove the above, we show the contrapositive:

\[
P = NP \Rightarrow PH = P \Rightarrow EH = E \Rightarrow L_n \in E, \text{ but we have already proven SIZE}(L_n) = s_n, \text{ which gives the contradiction}.
\]

4 Proof of Karp-Lipton theorem

Using the standard self-reduction techniques, we see that if there exists a small circuit \( C_n \) solving \( \text{SAT} \) on size \( n \) formulae, then there also exists a small circuit \( C'_n \) which also finds a satisfying assignment for the satisfiable formulae. Now we define the following language

\[
\text{SAT-SOLVE} = \{ (C, n) \mid C' \text{ solves SAT on formulae of size } n \}
\]

We want to show that \( \text{SAT-SOLVE} \) is in \( \text{coNP} \). The idea, in the two-player argument language, is that given \( C \) and \( n \), we want to construct \( C' \), and then we allow for the other player to try and refute by producing some formula \( \varphi \), where \( |\varphi| = n \) and an assignment to variables \( y \), such that \( C'(\varphi) = 0 \) and \( \varphi(y) = 1 \), which would refute that \( C' \) solves \( \text{SAT} \) and in turn refute that \( C \) solves \( \text{SAT} \).

Now assume \( NP \subseteq P_{/\text{poly}} \), and let \( L \) be any language in \( \Sigma^p_3 \), i.e.

\[
x \in L \Leftrightarrow \exists y_1 \forall z_2 \exists y_3 R(x, y_1, z_2, y_3),
\]

where of course \( |y_1|, |z_2|, |y_3| \leq \text{poly}(|x|) \). If we fix \( x, y_1, z_2 \), the \( NP \)-predicate \( R \) is just a boolean formula in \( y_3 \). Now, above we have seen that if there are small circuits for \( \text{SAT} \), recognizing them is in \( \text{coNP} \), so now
we will interleave the first two levels of quantifiers in the above definition of \( L \) with picking a circuit that solves SAT, which allows us to get rid of the final existential quantifier:

\[
x \in L \iff \exists y_1, C \forall z_2, \phi', y' \left[ C'(\phi_{x,y_1,z_2}) = 1 \land \left( C'(\phi') = 1 \lor \phi'(y') = 0 \right) \right]
\]

If \( C'(\phi_{x,y_1,z_2}) = 1 \), then the first player really could produce a \( y_3 \) for the third round if it were to be played (since \( C' \) also computes a satisfying assignment). So that first part of the predicate is what lets us lose the third existential quantifier. The second part of the predicate ensures that \( C' \), and therefore \( C \), really is a SAT solver. Thus, under the assumption that \( \text{NP} \subseteq P / \text{poly} \), we have managed to put an arbitrary problem in \( \Sigma_3^p \) into \( \Sigma_2^p \), which of course means that the polynomial hierarchy collapses to the \( 2^{nd} \) level, and so \( \text{PH} = \Sigma_2^p \).

### 4.1 Proof of Corollary 2.3

We can now show \( \Sigma_2^p \not\subseteq P / \text{poly} \). We distinguish two cases:

- **NP \not\subseteq P / \text{poly}**: If \( \text{NP} \) is not contained in \( P / \text{poly} \), then certainly \( \text{E} \not\subseteq P / \text{poly} \), and so \( \Sigma_2^p \not\subseteq P / \text{poly} \).

- **NP \subseteq P / \text{poly}**: By Karp-Lipton, \( \text{PH} = \Sigma_2^p \Rightarrow \text{EH} = \Sigma_3^p \Rightarrow L_n \in \Sigma_2^p \), and we know that \( L_n \not\subseteq P / \text{poly} \), so \( \Sigma_2^p \not\subseteq P / \text{poly} \).