**Lower Bounds** **(CS294-152, Fall'18)**

# Functions With and Without Small Circuits

Notes for 9/24/18

*Lecturer: Ryan Williams, Scribe: Seri Khoury*

## 1 Introduction

In the previous lecture we saw that $\mathsf{EXP}^{\Sigma_2 P} \not\subseteq \mathsf{SIZE}(2^n/(cn))$ for a fixed $c \geq 1$[1]. Today we will consider lower bounds against $\mathsf{SIZE}(n^c)$ for a fixed $c \geq 1$, and lower bounds against $\mathsf{PSIZE} = \bigcup_c \mathsf{SIZE}(n^c)$. Our goal is to find the "weakest" functions which do not have small circuits, to put these hard functions in the smallest complexity classes possible. Why the smallest class possible? Although the question is arguably very mathematically interesting in itself, here are two concrete complexity-theoretic reasons:

1. As we mentioned in the previous lecture: As shown by [3], If $\mathsf{E} = \mathsf{TIME}[2^{O(n)}] \not\subseteq i.o\text{-}\mathsf{SIZE}(2^{\alpha \cdot n})$ for some $\alpha > 0$, then $\mathsf{P} = \mathsf{BPP}$.
2. If a class $C$ is not in $\mathsf{PSIZE}$, then $\mathsf{P} \neq C$ (Imagine how cool would it be to prove this for $C = \mathsf{NP}$, or $C = \mathsf{PSPACE}$). This is because $\mathsf{P} \subseteq \mathsf{PSIZE}$, as we are going to see in Section 3.1 in Corollary 3.1.

Here is one open question that, at least for me, emphasizes how little we know about the "actual" power of the model of circuit families:

**Open Question 1.** $\mathsf{E}^{\mathsf{NP}} \subseteq \mathsf{SIZE}(n^2)$*?*

At first sight, why would someone believe that such a powerful class as $\mathsf{EXP}^{\mathsf{NP}}$ has small circuits? Shouldn't it be possible to prove a lower bound, maybe by some hardcore combinatorics argument? The answer is: maybe, but the *infinite and non-uniform*[2] computational model of circuit families has some power that is easy to neglect at first sight, when we look at such open questions. For example, in the next section, we are going to see that there are undecidable problems in $\mathsf{SIZE}(O(1))$, and the proof is going to rely heavily on the non-uniformity of circuits. This might shed some light on why it is so difficult to prove circuit lower bounds.

[1] Recall that $\mathsf{SIZE}(s(n))$ is the class of functions $f : \{0,1\}^* \to \{0,1\}$ such that *for all but finitely many* $n$, $f_n$ ($f$ on inputs of size $n$) has a circuit of size at most $s(n)$. The class $i.o.\text{-}\mathsf{SIZE}(s(n))$ has the same definition but "all but finitely many" is replaced with "infinitely many".

[2] By infinite and non-uniform we mean that in a family of circuits $\mathcal{C} = \{C_n \mid n \in \mathbb{N}\}$ that computes a function $f$, we get one circuit $C_n$ for each input length $n$.

# 2 Undecidable Functions With Small Circuits

In this section we prove that there are undecidable functions that can be computed by small circuits.

**Theorem 2.1.** *There are undecidable functions in* $\mathsf{SIZE}(O(1))$.

*Proof.* Let $f : \{0,1\}^* \to \{0,1\}$ be an undecidable function, and let $\mathsf{bin} : \mathbb{N} \to \{0,1\}^*$ be an efficient $1:1$ encoding of all natural numbers in binary. Define $f' : \{0,1\}^* \to \{0,1\}$ as follows:

$$f'(x) = \begin{cases} 1 & \text{if } f(\mathsf{bin}(|x|)) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Hence, for each $x \in f$, $f'$ contains *all* the possible strings of length $int(x)$, where $int(x)$ is the integer value of the binary string $x$. First, observe that $f'$ is also undecidable, as if we could decide $f'$, then we could decide $f$ on input $x$ by computing $f'$ on $1^{int(x)}$, i.e., there is a simple exponential time reduction from $f$ to $f'$. Furthermore, observe that for any $n$, either $f'_n$ is all ones, or $f'_n$ is all zeros. Therefore, for any $n$, $f'_n$ can be implemented with one gate: either $x \vee \neg x$, or $x \wedge \neg x$. □

The key point here, as we discussed at the end of the introduction, is that the infinite and non-uniform computational model of circuit families has a very different behavior in general from the algorithmic model, where programs are assumed to be finite strings. This is one of the reasons for why it is so difficult to prove circuit lower bounds.

# 3 Some Upper Bounds

In this section we show that efficient algorithms, deterministic or randomized, do have small circuits.

## 3.1 Deterministic $\mathsf{P}$-Time Algorithms Have Small Circuits

We can relate algorithmic time to circuit size in the following way:

**Theorem 3.1.** *There is a constant* $c \geq 1$ *such that for any function* $f$*, it holds that* $f \in \mathsf{DTIME}(t(n)) \Rightarrow f \in \mathsf{SIZE}(t^c(n))$.

*Proof.* First, we point out that for any deterministic model, a time-$t(n)$ algorithm in that model can be converted into a one-tape Turing Machine $\mathsf{M}$ running in $t^b(n)$ time, for some constant $b$ (see also Claim 1.6 in Chapter 1 in [1]). We will show how to simulate $\mathsf{M}$ on inputs of length $n$ with a circuit $C_n$ of size $O(t^{2b}(n))$.

Consider the computation tableau $\mathsf{T}$ of $\mathsf{M}$ on input $x$ of length $n$. This is a $t^b(n) \times t^b(n)$ matrix. Each row encodes a configuration of $\mathsf{M}$, and each column corresponds to a particular entry of the tape, and how it changes over time. That it, the cell $\mathsf{T}_{(i,j)}$ contains the value of the $j$'th entry of the tape at time $i$ of the computations, and a bit indicating whether the head of the tape happens to be there. If this bit is on, then $\mathsf{T}_{(i,j)}$ contains also the state of $M$ at time $i$. In total, each cell contains $k = O(1)$ bits of information. The key observation here is that for all cells $\mathsf{T}_{(i,j)}$ below the top row, $\mathsf{T}_{(i,j)}$ is a function of at most 3 cells on the row above it: $\mathsf{T}_{(i-1,j-1)}$, $\mathsf{T}_{(i-1,j)}$, and $\mathsf{T}_{(i-1,j+1)}$. Indeed, at each cell we just need to implement a function $f : \{0,1\}^{3k} \to \{0,1\}^k$, which takes in $\mathsf{T}_{(i-1,j-1)}$, $T_{(i-1,j)}$, $T_{(i-1,j+1)}$ and outputs the value for $\mathsf{T}_{(i,j)}$.

As we saw in the previous lecture, each function on $n$ inputs can be implemented by a circuit of size $O(2^n/n)$. Therefore, we can replace each cell $T_{i,j}$ below the top row by a circuit of size $O(2^{3k})$: one circuit of size $O(2^{3k}/k)$ for each of the $k$ bits of $T_{i,j}$. To conclude the proof, on the bottom row we want to know whether $\mathsf{M}$ is in an accept state. We can take an OR over all cells in the last row to check if any are in an accept state. For inputs of length $n$, the total size of the circuit is $O(2^{3k} \cdot t^{2b}(n)) = O(t^{2b}(n))$. □

**Corollary 3.1.** $\mathsf{P} \subseteq \mathsf{PSIZE}$. *Furthermore, if a class $C$ is not in* $\mathsf{PSIZE}$*, then* $\mathsf{P} \neq C$.

Indeed, Theorem 3.1 is intuitive and not very hard to prove. But can we show that there is a fixed $k$ for which we can convert any $\mathsf{P}$-time algorithm to a circuit of size $O(n^k)$? Interestingly, this is still open.

**Open Question 2.** *Is there a fixed $k$ for which* $\mathsf{P} \subseteq \mathsf{SIZE}(n^k)$?

## 3.2 Randomized Algorithms Have Small Circuits

After proving that efficient deterministic algorithms have small circuits, let's prove a similar theorem for randomized algorithms. First, recall the definition of the class $\mathsf{BPP}$:

**Definition 1.** $f \in \mathsf{BPP} \iff$ *there is a polynomial $p(n)$ and a deterministic $p(n)$-time algorithm $\mathsf{A}(*,*)$, such that for all $x$, $Pr_{(r \in \{0,1\}^{p(|x|)})}[A(x,r) = f(x)] > 2/3$*

**Theorem 3.2.** $\mathsf{BPP} \subseteq \mathsf{PSIZE}$

*Proof.* Let $f \in \mathsf{BPP}$, and let $\mathsf{A}$ be the corresponding algorithms from Definition 1. Observe that we can amplify the success probability to be greater than $1 - 1/2^n$, by the following simple trick that is widely used. On input $x$ of length $n$, instead of running $\mathsf{A}$, pick at random $r_1, ..., r_{100n} \in \{0,1\}^{p(n)}$, and compute $\mathsf{Majority}(\mathsf{A}(x, r_1), ..., \mathsf{A}(x, r_{100n}))$. Let $Y_i$ be a random variable indicating whether $\mathsf{A}(x, r_i) = f(x)$, and let $Y = \sum_i Y_i$. By Chernoff's bounds (see also [10]), $Pr[\mathsf{Majority}(\mathsf{A}(x, r_1), ..., \mathsf{A}(x, r_{100})) \neq f(x)] = Pr[Y \leq 50n] = Pr[Y \leq \frac{3}{4}E[Y]] < e^{-E[Y]/32} < 1/2^n$.

Therefore, for a given $x$, the success probability is $1 - 1/2^n$. By the Union-Bound, the probability that there is some $x \in \{0,1\}^n$ for which the algorithm fails is strictly less than 1. Hence, there are $r_1^*, ...r_{100n}^*$ for which the algorithm does not fail on any $x$. This is also called sometimes the probabilistic method to prove the existence of an object with certain properties (in our example, the object is the set of strings $r_1^*, ..., r_{100n}^*$ that makes the algorithms outputs the correct answer for any input).

Given $r_1^*, ..., r_{100n}^*$, the algorithm $\mathsf{A}(*, \{r_1^*, ..., r_{100n}^*\})$ is deterministic. Hence, to conclude the proof, we can convert the deterministic polynomial time algorithm $\mathsf{A}(*, \{r_1^*, ..., r_{100n}^*\})$ to a $\mathsf{PSIZE}$ circuit by Theorem 3.1. □

As explained in the book of Arora and Barak (see chapter 19 and 20 in [1]), under widely believed complexity-theoretic conjectures, $\mathsf{P} = \mathsf{BPP}$. Hence, we suspect that $\mathsf{BPP} = \mathsf{P}$, and that by the Time Hierarchy Theorem (see also Chapter 3.1 in [1]), $\mathsf{BPP}$ is a proper subset of, say, $\mathsf{DTIME}(n^{\log n})$. Yet, currently we are not even able to show that $\mathsf{BPP}$ is a proper subset of $\mathsf{NEXP}$.

**Open Question 3.** $\mathsf{BPP} = \mathsf{NEXP}$*?*

# 4 Algorithms With Advice

Circuit families constitute one non-uniform model of computation. There is a more general way to characterize how "non-uniform" programs can help to solve problems. Consider the following situation: suppose that you are given a polynomial time to solve a problem, but for inputs of length $n$, you are allowed to write a program of $f(n)$ lines of code. The program must still run in, say, poly-time, but the code size is allowed to grow with the input. What problems can you then solve? It is a natural question! And it is inherently related to Boolean circuit complexity. This leads us to the following definition of algorithms with advice.

**Definition 2.** *Given an algorithm* $\mathsf{A}$ *and a boolean function* $f$*, we say that* $\mathsf{A}$ *decides* $f$ *with* $s(n)$ *advice if for all* $n > 0$*, there is a string* $a_n \in \{0,1\}^{s(n)}$*, such that for all* $x \in \{0,1\}^n$*,* $f(x) = \mathsf{A}(x, a_n)$.

Observe that it is very important to have the *"for all $x$"* quantifier after the *"for all $n$"* quantifier. Otherwise the definition would not make any sense, as any problem can be solved with one bit of advice in that case (in one step of computation..). In other words, it is very important that the advice is given prior to seeing the input.

**Definition 3.** *$\mathsf{P}/s(n)$ is the class of decision problems that can be solved by a polynomial time algorithm with $s(n)$ bits of advice.* $\mathsf{P}/\text{poly} = \bigcup_c \mathsf{P}/n^c$.

Next we see that the non-uniform model of circuits families is closely related to the non-uniform model of computation with advice.

**Theorem 4.1.** $\mathsf{P}/\text{poly} = \mathsf{PSIZE}$

*Proof.* $\Rightarrow$ To show that $\mathsf{P}/\text{poly} \subseteq \mathsf{PSIZE}$, we can simply take the deterministic algorithm that takes an additional input of $\text{poly}(n)$ bits of advice, convert it to a circuit by Theorem 3.1, and hard code the $\text{poly}(n)$ bits of advice into the input of this circuit.

$\Leftarrow$ To show that $\mathsf{PSIZE} \subseteq \mathsf{P}/\text{poly}$, we can simply encode a poly-size circuit as a binary string and use it as an advice. Given an input $x$, a poly-time algorithm evaluates the circuit $C_{|x|}$ that is given by the advice and computes $C_{|x|}(x)$. □

# 5 $\Sigma_2\mathsf{P}$ and Circuit Lower Bounds

Sections 5 and 6 are dedicated to lower bounds. In this section we show a circuit lower bound for $\Sigma_2\mathsf{P}$. We start with a weaker circuit lower bound against $\mathsf{P}^{\Sigma_2\mathsf{P}}$. In the previous lecture we saw that $\mathsf{EXP}^{\Sigma_2 P} \not\subseteq \mathsf{SIZE}(2^n/(cn))$ for a fixed $c \geq 1$. Intuitively, one would imagine that we can "scale down" the argument for $\mathsf{EXP}^{\Sigma_2 P} \not\subseteq \mathsf{SIZE}(2^n/(cn))$ to show a *fixed* poly-size lower bound for functions in $\mathsf{P}^{\Sigma_2\mathsf{P}}$. Unsurprisingly, this is exactly the first lower bound we are going to see.

**Theorem 5.1.** *For all $k$, there is an $f_k \in \mathsf{P}^{\Sigma_2\mathsf{P}}$ such that $f_k \notin \mathsf{SIZE}(n^k)$.*

*Proof.* Let $m$ be a natural number. In the previous lecture, we showed how to find a function $g : \{0,1\}^m \to \{0,1\}$ with maximum circuit complexity (e.g., $2^m/(2m)$) in time $2^{O(m)}$ with an oracle to the function COMPLEX[3] which is in $\Sigma_2\mathsf{P}$.

For a given $k$, here is a $\mathsf{P}^{\Sigma_2\mathsf{P}}$ function $f_k$:

[3] We use COMPLEX here as a black box. See the previous lecture for the formal definition of COMPLEX.

Given an input $x$ of length $n$,

- Set $m := (k+1) \cdot \log(n)$.
- If $x$ is not of the form $0^{n-m}y$, reject.
- Otherwise, in $2^{O(m)} = n^{O(k)}$ time, use an oracle for COMPLEX to construct a function $g$ on $m$ inputs with maximum circuit complexity.
- Output $g(y)$.

Let $C_n$ be any circuit for $f_k$ on $n$-bit inputs. From $C_n$, we can easily get a circuit $C'_m$ for $g$ on $m$-bit inputs: $C'_m(y) := C_n(0^{n-m}y)$, i.e., hard-code $n-m$ inputs to be 0. Therefore, $\mathsf{SIZE}(C_n) \geq \mathsf{SIZE}(C'_m) \geq n^{k+1}/(2(k+1)\log n) > n^k$ for large enough $n$. Hence, $f_k$ is not in $\mathsf{SIZE}(n^k)$. □

Our goal next is to improve the result in Theorem 5.1 and to prove a lower bound against $\Sigma_2\mathsf{P}$. The following theorem was shown by Kannan [4].

**Theorem 5.2.** *For all $k$, there is an $f_k \in \Sigma_2\mathsf{P}$ such that $f_k \notin \mathsf{SIZE}(n^k)$.*

Before proving Theorem 5.2, let us go back to the P versus NP discussion, which appears to be related. As the strict majority of computer scientists believe, $P \neq \mathsf{NP}$. If indeed it happens to be the case, then one direction to prove this would be to show that $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. If indeed $\mathsf{NP} \not\subseteq \mathsf{P/poly}$, then obviously this would also prove Theorem 5.2. But what happens if $\mathsf{NP} \subseteq \mathsf{P/poly}$? Besides blocking one direction for proving that $\mathsf{P} \neq \mathsf{NP}$, it would give us some very interesting consequences. One example is the following theorem by Karp and Lipton [5].

**Theorem 5.3.** $\mathsf{NP} \subseteq \mathsf{P/poly} \Rightarrow \Sigma_3\mathsf{P} = \Sigma_2\mathsf{P}$.

In fact, Karp and Lipton showed a stronger result: $\mathsf{NP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{PH} = \Sigma_2\mathsf{P}$, where $\mathsf{PH} = \bigcup_c \Sigma_c\mathsf{P}$. We will shortly prove Theorem 5.3. But first, let us show how it immediately implies Theorem 5.2.

**Proof of Theorem 5.2.**
- If $\mathsf{NP} \not\subseteq \mathsf{P/poly}$, then SAT does not have an $n^k$ circuit for any $k$, and we are done.
- If $\mathsf{NP} \subseteq \mathsf{P/poly}$, by Theorem 5.3, it holds that $\Sigma_3\mathsf{P} = \Sigma_2\mathsf{P}$. Since $\mathsf{P}^{\Sigma_2\mathsf{P}} \subseteq \Sigma_3\mathsf{P}$ (see also Theorem 5.12 in Chapter 5 in [1]), then by Theorem 5.1 we are done. □

It remains of course to prove Theorem 5.3. Intuitively, we need to show that if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then we can somehow "give up" on the last existence quantifier in $\Sigma_3\mathsf{P}$.

**Proof of Theorem 5.3.** Let $f \in \Sigma_3\mathsf{P}$. By definition, there is a polynomial $q$ and a polynomial-time TM M such that:
$x \in f \iff \exists y_0 \in \{0,1\}^{q(|x|)}\ \forall y_1 \in \{0,1\}^{q(|x|)}\ \exists y_2 \in \{0,1\}^{q(|x|)}\ \ \mathsf{M}(x, y_0, y_1, y_2) = 1.$

Define $B$ to be a function that takes in inputs of the form $(x, y_0, y_1)$, and outputs 1 if and only if $\exists y_2 \in \{0,1\}^{q(|x|)}$ $\mathsf{M}(x, y_0, y_1, y_2) = 1$.

Observe that $B \in \mathsf{NP}$. Furthermore, if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then by a standard reduction from search to decision, there are polynomial size circuits $\{C_n\}$ which given an input of the form $(x, y_0, y_1) \in B$, output a witness $y_2$ for for which $\mathsf{M}(x, y_0, y_1, y_2) = 1$.

Now we use this family of circuits to get rid of one existence quantifier. The idea is to guess the circuit $C_{|x|}$, *before* the $\forall y_1$ quantifier. We will not need the $\exists y_2$ quantifier any more, because we can use the circuit $C_{|x|}$ to print a string $y_2$ for $\mathsf{M}(x, y_0, y_1, y_2)$. Therefore, in case $\mathsf{P/poly}$, for $f \in \Sigma_3\mathsf{P}$, it holds that:
$x \in f \iff \exists y_0 \in \{0,1\}^{q(|x)}$ and there is a poly-size circuit $C_{|x|}$ such that $\forall y_1 \in \{0,1\}^{q(|x|)}$ $\mathsf{M}(x, y_0, y_1, C_{|x|}(x, y_0, y_1)) = 1$.

Therefore, $f \in \Sigma_2\mathsf{P}$. □

**Corollary 5.1.** *(*of Theorem 5.3*)* $\Sigma_2$ EXP *is not in* P/poly.

*Proof.* If $\Sigma_2$ EXP is in P/poly, then NP is in P/poly. By Theorem 5.3, it holds that $\Sigma_3\mathsf{P} = \Sigma_2\mathsf{P}$. Furthermore, by a simple padding argument, it follows that $\Sigma_3$ EXP $=$ $\Sigma_2$ EXP. But $\mathsf{EXP}^{\Sigma_2\mathsf{P}}$ is in $\Sigma_3$ EXP, and as we saw in the previous lecture, $\mathsf{EXP}^{\Sigma_2\mathsf{P}}$ has functions of *maximum* circuit complexity, i.e., $2^n/cn$, for a fixed $c$. This is a contradiction. □

Observe that we can improve the P/poly in the above corollary to $\mathsf{SIZE}(s(n))$ for any $s(n)$ that is sub-exponential.

In general, improved Karp-Lipton theorems (Theorem 5.3) imply improved circuit lower bounds:

**Corollary 5.2.** *(*of Theorem 5.3*) If (*NP $\subseteq$ *P/poly* $\Rightarrow$ PH $=$ *C), then for all k, C is not in* $\mathsf{SIZE}(n^k)$.

Let us finish this section with the following interesting open question. As we showed in this section, for all $k$, $\mathsf{P}^{\Sigma_2\mathsf{P}}$, or even $\Sigma_2\mathsf{P}$ is not in $\mathsf{SIZE}(n^k)$. Can we say something about $\mathsf{P}^{\Sigma_1\mathsf{P}} = \mathsf{P}^{\mathsf{NP}}$?

**Open Question 4.** $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{SIZE}(O(n))$*?*

While this is still open, people believe that the answer is NO. Most of complexity-theoretic computer scientists even believe that the following much stronger statement is true.

**Conjecture 5.1.** $\mathsf{P} \not\subseteq \mathsf{SIZE}(O(n))$

# 6 Merlin-Arthur Protocols and Circuit Lower Bounds

In this section we prove a circuit lower bound against $\mathsf{MA}/1$. This is the Merlin-Artur complexity class with one bit of advice (see also Chapter 8 in [1], for a further explanation about the class $\mathsf{MA}$).

## 6.1 Motivation and Warm-Up

Before we jump into proving the lower bound, we prepare the ground with some motivation and easy claims that are going to be useful in the proof. Let us start with the following definition.

**Definition 4.** *(The Merlin-Arthur Class). A function $f$ is in $\mathsf{MA}$ if there is a polynomial time algorithm $\mathsf{V}$, and a polynomial $q$, such that:*

1. *If $f(x) = 1$, then there is $y \in \{0,1\}^{q(|x|)}$ such that $Pr_{z \in \{0,1\}^{q(|x|)}}[\mathsf{V}(x, y, z) = 1] = 1$.*
2. *If $f(x) = 0$, then for any polynomial size $y$, $Pr_{z \in \{0,1\}^{q(|x|)}}[\mathsf{V}(x, y, z) = 1] < 1/10$.*

Intuitively, the class $\mathsf{MA}$ defined above is the *probabilistic version of* $\mathsf{NP}$. Think about $\mathsf{V}$ as a verification algorithm; if $f(x) = 1$ and $\mathsf{V}$ is faced with a "good proof", then it always accepts. If $f(x) = 0$ and it is faced with a "bad proof", then it has some small chance of error (note: if $f(x) = 1$ but we give $\mathsf{V}$ a "bad proof", there is no guarantee on the success probability).

**Observation 1.** $\mathsf{MA} \subseteq \Sigma_2\mathsf{P}$.

The above observation follows immediately by the definitions of $\mathsf{MA}$ and $\Sigma_2\mathsf{P}$. Hence, proving circuit lower bounds for $\mathsf{MA}$ would be stronger than proving circuit lower bounds for $\Sigma_2\mathsf{P}$, which was our goal in the previous section. While this already gives a huge motivation to study the class $\mathsf{MA}$ and to try to prove lower bounds for it, there is a more fundamental complexity-theoretic reason. Under a widely believed assumptions, $\mathsf{MA} = \mathsf{NP}$[4]. Therefore, circuit lower bounds for $\mathsf{MA}$ could be used to make substantial progress towards proving circuit lower bounds for $\mathsf{NP}$, which in turn could make a huge progress towards answering the most important question in Computer Science, $\mathsf{P}$ versus $\mathsf{NP}$[5]. In 2007, Rahul Santhanam [7] achieved something very close to a lower bound for the class $\mathsf{MA}$. He showed an $n^k$ circuit lower bound for languages accepted by Merlin-Arthur machines running in polynomial time and using only a single bit of advice.

**Theorem 6.1.** *[7] For each $k$, $\mathsf{MA}/1 \not\subseteq \mathsf{SIZE}(n^k)$.*

[4] Recall that it is widely believed that $\mathsf{P} = \mathsf{BPP}$. Since $\mathsf{MA}$ is the probabilistic version of $\mathsf{NP}$, for the same widely believed derandomization assumptions, it is also believed that $\mathsf{MA} = \mathsf{NP}$.

[5] Recall that $\mathsf{P} \subseteq \mathsf{PSIZE}$.

Recall that in the Karp-Lipton proof of the lower bound for $\Sigma_2\mathsf{P}$ that we saw in the previous section, it was very useful to compare $\mathsf{NP}$ to $\mathsf{P/poly}$. The argument was separated into two cases. The first is in which $\mathsf{NP} \not\subseteq \mathsf{P/poly}$, which was the easier case, and the second is in which $\mathsf{NP} \subseteq \mathsf{P/poly}$, which implied that $\Sigma_3\mathsf{P} = \Sigma_2\mathsf{P}$. Here, we can't use the same argument, because we don't know whether $\mathsf{NP} \subseteq \mathsf{P/poly}$ implies that $\Sigma_3\mathsf{P} = \mathsf{MA}$. We are going to do a similar Karp-Lipton style argument, but instead of comparing $\mathsf{NP}$ to $\mathsf{P/poly}$, we compare $\mathsf{PSPACE}$ to $\mathsf{P/poly}$. However, here, the easier case is the one in which $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. This is because, as we are going to prove in Lemma 6.1, if $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$, then $\mathsf{PSPACE} = \mathsf{MA}$. Since $\Sigma_2\mathsf{P} \subseteq \mathsf{PSPACE}$, this concludes the proof of Theorem 6.1 for the case in which $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. What happens if $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$? This requires more work, and the next section is dedicated to this. To prove Lemma 6.1, we need the following Theorem that follows from work on interactive proofs [9, 6, 8, 2].

**Theorem 6.2.** *There is a* $\mathsf{PSPACE}$*-complete language* $L$ *and a probabilistic polynomial-time oracle Turing Machine* $\mathsf{M}$*, such that for any input* $x$*, it holds that:*

1. $\mathsf{M}$ *only asks the oracle queries of length* $|x|$.
2. *If* $\mathsf{M}$ *is given* $L$ *as oracle, and* $x \in L$*, then* $\mathsf{M}$ *accepts with probability 1.*
3. *if* $x \notin L$*, then irrespective of the oracle given to* $\mathsf{M}$*, it rejects with probability at least* $9/10$.

Here we are going to use Theorem 6.2 as a black-box without proving it. Now we are ready to prove the following lemma.

**Lemma 6.1.** $\mathsf{PSPACE} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{PSPACE} = \mathsf{MA}$.

*Proof.* First, observe that it is easy to see that $\mathsf{MA} \subseteq \mathsf{PSPACE}$. Now we show that if $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$, then $\mathsf{PSPACE} \subseteq \mathsf{MA}$. Assume that $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$, and let $L$ be the $\mathsf{PSPACE}$ complete language from Theorem 6.2. We show that $L \in \mathsf{MA}$, by showing a Merlin-Arthur protocol for $L$. On input $x$:

1. Merlin guesses a circuit $C_{|x|}$, intended to be a circuit for $L$, and sends it to Arthur.
2. Arthur runs $\mathsf{M}$ from Theorem 6.2, whenever $\mathsf{M}$ wants to ask the oracle $L$ a query, he can simulate $C_{|x|}$ instead.

This is a Merlin-Arthur protocol for $L$. This is because when Merlin sends a circuit to Arthur, he commits to a specific oracle. The rest follows by Theorem 6.2. If $L(x) = 1$, then there is a $C$ such that $Pr[\mathsf{V}(x, C_{|x|})\ accepts] = 1$. If $L(x) = 0$, then for all $C^*$, $Pr[\mathsf{V}(x, C^*_x)\ accepts] < 1/10$. □

In fact, if we remove the first condition in Theorem 6.2, then it holds for any PSPACE complete language. This "weaker" version of Theorem 6.2 is already sufficient to prove Lemma 6.1, as instead of guessing one circuit, the prover can guess polynomially many circuits corresponding to polynomially many queries that the verifier can make (since the verifier runs in polynomial time), which is in total a polynomial amount of information that the prover sends to the verifier. However, the first condition on the length of the queries is going to be crucial in the sequel, as we deal with the case in which $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$.

## 6.2 Proof of Theorem 6.1: what happens if $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$?

Let us now complete the proof of Theorem 6.1. It is remained to deal with the case in which $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$. Recall, we need to show that for any $k$ there is a function $f_k \in \mathsf{MA}/1$ such that $f_k \notin \mathsf{SIZE}(n^k)$. In the case we are dealing with here, we have that $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$. Therefore, $L$ from Theorem 6.2 is also not in $\mathsf{P/poly}$. That is, for any $k$, $L \notin \mathsf{SIZE}(n^k)$. Our approach is going to be as follows. For any $k$, we are going to design from $L$ another language $L'_k$ such that $L'_k \in \mathsf{MA}/1$, but $L'_k \notin \mathsf{SIZE}(n^k)$.

How do we design such a language? The standard approach would be to take inputs for $L$ and to "pad" them "enough" such that evaluating a circuit for $L$ becomes a polynomial-time task with respect to the length of the padded input. Then, $L'_k$ is defined to be the language that contains these padded inputs, and a Merlin-Arthur protocol for $L'_k$ would be as follows. Merlin guesses circuit for $L$ and sends it to Arthur. Arthur uses the circuit to decide inputs for $L'_k$. Our hope in this approach is to prove that the above argument is sufficient to show that $L'_k \in \mathsf{MA}$, but $L'_k \notin \mathsf{SIZE}(n^k)$.

However, there are two issues with this approach. First, let us clarify why the first condition in Theorem 6.2 is so important. If the circuit complexity of $L$ on inputs of length $n$ is $s(n)$ and the prover is allowed to ask the oracle queries of polynomial length, then the above approach only guarantees that on inputs of length $n$, the prover runs in $\mathrm{poly}(s(\mathrm{poly}(n)))$ time. But we don't have any guarantee on the behavior of $s$, namely, we don't know whether $s(\mathrm{poly}(n)) = \mathrm{poly}(s(n))$. This is exactly where the first condition in Theorem 6.2 comes into play, as it gives a constraint on the length of the oracle queries, which deals with this issue.

The second issue is that Arthur doesn't know $s$. Therefore, he doesn't know whether the input is padded "enough" such that the circuit evaluation indeed takes only poly-time. Our extra bit of advice will take care of this, telling Arthur when he has enough padding. Now we are ready to formulate the above intuitions and deduce the following lemma, which concludes Theorem 6.1.

**Lemma 6.2.** *If* $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$*, then for any* $k > 0$*, it holds that* $\mathsf{MA}/1 \not\subseteq \mathsf{SIZE}(n^k)$.

*Proof.* Let $L$ be the language in Theorem 6.2. Since $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$, for any $k > 0$, $L \notin \mathsf{SIZE}(n^k)$. Define $L'_k$ as follows:

$L'_k = \{x1^y \mid x \in L, y \geq |x| > 0$ is a power of 2, and the minimum circuit size of $L_{|x|}$ is between $(y + |x|)^{k+1}$ and $(2y + |x|)^{k+1}\}$

We show that $L'_k \in \mathsf{MA}/1$ but $L'_k \notin \mathsf{SIZE}(m^k)$, where $m$ here is the input's length for $L'_k$:

1. $L'_k \in \mathsf{MA}/1$: here is a Merlin-Arthur protocol with a single bit of advice for $L'_k$. Let $x'$ be an input of length $m$. The single bit of advice is set to 1 if $m$ is of the form $y + n$, where $y \geq n > 0$ is a power of 2, and the minimum circuit size for $L_n$ is between $(y + n)^{k+1}$ and $(2y + n)^{k+1}$. Otherwise, it is set to 0. Observe that if $m$ is of the above form then $y$ and $n$ are determined solely by $m$, the input's length[6].

   If the bit of advice is 0, or $x'$ is not of the from $x1^y$ (i.e., it doesn't end with number of ones that is at least half of the input's length), then Arthur rejects immediately. Otherwise Merlin guesses an $s$ between $(y + n)^{k+1}$ and $(2y + n)^{k+1}$, and a circuit $C_n$ of size $s$, intended to be a circuit for $L_n$, and sends it to Arthur. Arthur runs $\mathsf{M}$ from Theorem 6.2, on $x$, whenever $\mathsf{M}$ wants to ask the oracle $L$ a query, Arthur can simulate $C_n$ instead (observe that the simulation takes a polynomial time with respect to the length of the input $x'$). The correctness of the protocol follows form Theorem 6.2.

2. $L'_k \notin \mathsf{SIZE}(m^k)$: suppose for the sake of contradiction that $L'_k \in \mathsf{SIZE}(m^k)$, and denote by $\{C'_m\}_{m=1}^{\infty}$ the family of circuits for $L'_k$. Recall that we denote by $s(n)$ the minimum circuit size for $L_n$. Since $L \notin \mathsf{P/poly}$, there is an infinite sequence of numbers $I$, such that for any $n \in I$, it holds that $s(n)$ is super-polynomial. Hence, for any $n \in I$, there is an $m$ of the form $m = y + n$ such that $y$ is a power of 2, $y \geq n > 0$, and the minimum circuit size for $L_n$ is between $(y + n)^{k+1}$ and $(2y + n)^{k+1}$. Therefore, for any $n \in I$ and the corresponding $m$ and $y$, it is easy to see that we can use $C'_m$ as a circuit for $L_n$, as for any $x$ of length $n$, we can simply hardcore $y$ ones into $C'_m$. This implies that the minimum circuit size for $L_n$ is at most $|C'_m| \leq (y + n)^k$, which contradict the fact that $s(n) \geq (y + n)^{k+1}$.

□

# References


[1] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.


[6] This is because by having $m$ in this form, it holds that $y$ is just the highest order bit in the binary representation of $m$. The rest of the bits in the binary representation give $n$.

[2] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 316–324, 2004.

[3] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if $E$ requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229, 1997.

[4] Ravi Kannan. A circuit-size lower bound. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 304–309, 1981.

[5] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309, 1980.

[6] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[7] Rahul Santhanam. Circuit lower bounds for merlin-arthur classes. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 275–283, 2007.

[8] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[9] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, pages 129–138, 2002.

[10] Wikipedia contributors. Chernoff bound — Wikipedia, the free encyclopedia, 2018. [Online; accessed 17-October-2018].