In this post, we are going to see a powerful computing model: circuit, which is a ubiquitous non-uniform model in TCS. Many of its relations with other computing models are remained unknown. For example, people still cannot find a **NP** language which does not have a linear size circuit! (The best known lower bound for **NP** is around $6n$)

In the following, I will first quickly recall the definition of circuit and some related complexity classes. After that, several lower bound results will be presented. Finally, we will end up with a very cool result: Razborovs monotone lower bound.

1 Circuit: A non-uniform computing model

Circuit is a common element in engineering, people uses wires to connect everything together in order to build a block that has certain functionality. In theoretical computer science, circuit plays a similar roll as representing function by connecting inputs with some gadgets/connectives. Here, we consider the simplest circuit model: Boolean circuit. That is, everything in Boolean circuit only take value from 0 or 1.

However, we can soon find out circuit is non-uniform in the sense that a single circuit cannot accept any input size. In other words, if we want to have circuits to compute certain function $f : \{0,1\}^n \to \{0,1\}$ for any $n \in \mathbb{N}$, we need to have a **family** of circuits which accepting input length $1, 2, 3, \dots$. From this observation, we formally define this computing model as a family of circuits.

Definition 1 (Boolean circuit) *A T-size circuit family is a set of circuits $\{C_n\}$ where $T : \mathbb{N} \to \mathbb{N}$ is a function such that the size of C_n is at most $T(n)$ for any $n \in \mathbb{N}$.*

We say a language L is decided by a T-size circuit family $\{C_n\}$ if for any $n\mathbb{N}$ and $x \in \{0,1\}^n$, $C_n(x) = 1$ if and only if $x \in L$.

Remark: The size of a circuit is the number of gadgets/connectives inside.

To quantify the computing power of circuit, it's nature to limit the size of a circuit. As a result, we have a direct complexity class for circuit as follow.

Definition 2 ($SIZE$) *Let $SIZE(T)$ be the class of languages decided by T-size circuit family.*

An interesting fact about size circuits with fix size is that given a number s, there are about $2^{O(s \log s)}$ size s circuit.

However, we soon find out that this definition behaves strangely in some cases. For instance, one can see that a polynomial size circuit can decide unary language, which could be undecidable for Turing machine! As a result, to make the complexity class of circuit and Turing machine to be consistent, people came up with a better definition for circuit complexity.

Definition 3 (P/poly) *Let* **P/poly** *be the class of languages decided by a family of polynomial-size circuits being generated by a Turing machine in polynomial time.*

Similar complexity classes with different size can be defined in the same way. The interesting point here is that we can have an equivalent definition for **P/poly** without mentioning circuit at all!

Definition 4 (P/poly, advice) **P/poly** *is the class of languages decided by a Turing machine in polynomial time with polynomial length advice.*

Remark: Advice is a set of **fixed** strings $\{a_n\}$ that are fed into a Turing machine with the input. Every input having the same size will be fed into the Turing machine with the same advise. One can simply think of the **description** of a circuit family as a set of advice.

P/poly is a powerful complexity class. One can easily verify that $\mathbf{P} \subseteq \mathbf{P/poly}$. Moreover, Karp-Lipton theorem showed that it is unlikely to be the case where $\mathbf{NP} \subseteq \mathbf{P/poly}$. While **P/poly** captures the polynomial time complexity in some sense, there are another line of work dealing with circuit with low **depth**. Namely, the circuits that can be computed very fast.

Definition 5 (NC, AC) *For $k \in \mathbb{N}$, we define*

- $\mathbf{NC_k}$ *be the class of languages decided by $O(\log^k n)$ depth circuit family with bounded fan-in And and Or gates.*
- $\mathbf{AC_k}$ *be the class of languages decided by $O(\log^k n)$ depth circuit family with unlimited fan-in And and Or gates.*

Let $\mathbf{NC} = \cup_{k\in\mathbb{N}}\mathbf{NC_k}$ *and* $\mathbf{AC} = \cup_{k\in\mathbb{N}}\mathbf{AC_k}$.

Remark: **NC** is a interesting computing model for parallel computing since it is a direct model for polylogarithmic time parallel computer with polynomially many processors.

2 Simple lower bound: $\Sigma_3 \not\subseteq SIZE(n^k)$

In this section, we are going to see that $\Sigma_3 \not\subseteq SIZE(n^k)$. Intuitively, this result provided a polynomial-size circuit lower bound for complexity class Σ_3. Note that **NP** $= \Sigma_1$ and we would really want to find such lower bound for it. (However, as we mentioned in the beginning, current best lower bound for **NP** is only linear size)

Let's formally state the theorem and prove it!

Theorem 6 $\Sigma_3 \not\subseteq SIZE(n^k)$

Proof: We will prove the theorem in two steps:

1. Given integer k, we will show that when n large enough, there's a function $f : \{0,1\}^n \to \{0,1\}$ has no n^k-size circuit.
2. The corresponding language of function in part 1. lies in Σ_3.

For the first step, we consider the first $k+1$ bits of the input and overlook the other. We can easily see that these $k+1$ can generate $2^{n^{k+1}}$ distinct functions. Now, consider circuit with size n^k, from certain counting method, we can show that there are at most $2^{n^k \log n^k}$ circuits with this size. When we carefully choose n such that $2^{n^{k+1}} > 2^{n^k \log n^k}$, then from pigeonhole principle, we know that there's some function has no size n^k circuit.

Next, we want to show that the language L decided by the above function can be decided by a Σ_3 machine. To show there's such machine, we know that it is equivalent to have a polynomial length Boolean formula with three alternating quantifiers starting from $\exists$.

The intuition of this Boolean formula is as follow:

- We want to find($\exists$) some function f such that $x \in L$ if and only if
- $f(x) = 1$
- For any($\forall$) n^k-size circuit C, there's($\exists$) an input y such that $C(y) \neq f(y)$.

We can simply write down the Boolean formula as follow:

$$x \in L \Leftrightarrow \exists f(\forall C \exists y(C(y) \neq f(y)) \wedge (f(x) = 1)$$

$\square$

3 Karp-Lipton Theorem

The golden ticket of theoretical computer science is to solve the **P** versus **NP** problem. And so far, almost everyone believes that it is the case $\mathbf{P} \neq \mathbf{NP}$. However, after decades of hardworking, people still cannot prove it. Since we know that **P** versus **NP** problem is highly connected to the *Polynomial Hierarchy*, it's nature to find some relations among the separation and collapse. Straightforward from the definition, it is known that when lower level of hierarchy collapses, then the upper levels will also collapse to that level. Thus, on the reverse direction, if we separate upper level of hierarchy, then the lower level must also be separated. Otherwise, it will lead to a contradiction.

With this structural properties in mind, after defining **P/poly**, people started to think about its relation toward complexity classes in the polynomial hierarchy. For instance, an ambitious goal could be showing that $\mathbf{NP} \not\subseteq \mathbf{P/poly}$. However, as we mentioned before, direct attack didn't work very well. Thus, researchers began to try some indirect method in order to locate the position of **P/poly**. It's tempting to show some results such as $\mathbf{NP} \not\subseteq \mathbf{P} \Rightarrow \mathbf{NP} \not\subseteq \mathbf{P/poly}$. But it turns out that we can only prove weaker results such as the following Karp-Lipton theorem.

Theorem 7 (Karp-Lipton) $\mathbf{PH} \neq \Sigma_2 \Rightarrow \mathbf{NP} \not\subseteq \mathbf{P/poly}$

Before we prove the theorem, I want to emphasize that theorems in this form often can be thought of in two directions: $A \Rightarrow B$ and $\neg B \Rightarrow \neg A$. Take Karp-Lipton as example, the theorem stated that $\mathbf{PH} \neq \Sigma_2 \Rightarrow \mathbf{NP} \not\subseteq \mathbf{P/poly}$. On the other direction, we have $\mathbf{NP} \subseteq \mathbf{P/poly} \Rightarrow \mathbf{PH} = \Sigma_2$. This is the two-side-of-a-coin: one is the containment/equality, the other is separation.

Now, we are going to prove the Karp-Lipton theorem in the equality direction.

PROOF: Suppose $\mathbf{NP} \in \mathbf{P/poly}$, then SAT has a family of polynomial-size circuits $\{C_n\}$ to generate satisfying assignment. To show that **PH** collapses to Σ_2, it suffices to show that $\Pi_2 \subseteq \Sigma_2$.

Consider $L \in \Pi_2$, there's a Boolean formula ϕ such that

$$x \in L \Leftrightarrow \forall y \exists z \phi(x, y, z) = 1$$

Now, as the circuit family for SAT is polynomial-size, it can be described by polynomial-length string. Thus, we can rewrite the above Π_2 formula into a Σ_2 formula with the help of the existence of such circuit.

$$x \in L \Leftrightarrow \exists C \forall y \phi(x, y, C(x, y)) = 1$$

The reason why we can pull out the existential quantifier is that for any input with the same length, the corresponding circuits are all the same. As a result, we can use the same circuit to find the satisfying assignment of ϕ no matter which y it is. As long as we know the existence of such circuit family, we can ensure that there will be no problem to pull out the existential quantifier. □

4 $\mathbf{AC_0}$ lower bound

As what we have defined above, $\mathbf{AC_k}$ is a class of language decided by polynomial-size log^k-depth circuit family with unlimited fan-in And and Or gate. Here, we consider $\mathbf{AC_0}$, namely, the depth of the circuit is **constant**. Since this computing model can finish its computation in constant time and some simple calculation such as integer addition/subtraction lie in $\mathbf{AC_0}$, we would like to understand its limit of it.

In 1986, Håstad showed a surprising fact that a seemingly simple problem: PARITY cannot be solved with $\mathbf{AC_0}$ circuit. As a result, $\mathbf{AC_0} \subsetneq \mathbf{NC_1}$ as the former contains PARITY. The proof beautifully used the *Switching lemma* to effectively shrink the depth of an $\mathbf{AC_0}$ circuit to 2. Then we can easily argue that a depth-2 circuit must not compute PARITY and thus PARITY$\notin \mathbf{AC_0}$.

Before we formally state the theorem, let's first see the definition of PARITY.

Problem (PARITY)
On input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, the output of PARITY problem is $x_1 \oplus x_2 \oplus \cdots \oplus x_n$.

Theorem 8 *PARITY$\notin \mathbf{AC_0}$. In particular, assume $n \geq 2^{O(k)}$ and $k > 2$. Computing PARITY of n bits with depth-k $\mathbf{AC}$ circuit requires size $S \geq 2^{\Omega(n^{1/(k-1)})}$*

Proof: For simplicity, here we only sketch the proof. First, let's see the *Switching lemma.*

Lemma 9 (Switching lemma (informal)) *We can switch any small DNF with small width to a CNF with small width with non-negligible probability.*

Now, let's suppose C_n is a circuit for computing n-bits PARITY function with size S.

1. We can first transform it into a circuit where the And and Or gates are **layered** with only polynomial blowup in size. Then, one can see that the button layer of the new circuit compute either a DNF or CNF function. WLOG, we assume it is a DNF.

2. By Switching lemma, we can switch the button layer into a CNF, which can be combined with the layer just above the original layer. That is, we can reduce the depth of the circuit by 1 without enlarging the size.

3. Repeatedly using Switching lemma and finally we have a depth-2 circuit for PARITY with non-negligible probability. Note that this resulting circuit is equivalent to a DNF or CNF.

4. We can show that any DNF/CNF computing PARITY of $n-$bits require size 2^{n-1}. Thus, we can conclude that the final 2-layer circuit must have size at least 2^{n-1}.

5. After tuning with the parameters in switching lemma, one can show that the original circuit must also has size at least $2^{\Omega(n^{1/(k-1)})}$

Since $\mathbf{AC_0}$ contains languages that can be solved with polynomial-size constant depth circuit, by the above argument, PARITY is clearly not in $\mathbf{AC_0}$.

$\square$

5 Razborovs monotone lower bound

Coming soon.