Simons Institute Open Lecture	**Speaker: Ryan Williams**
Thinking Algorithmically About Impossibility	
Simons Insitute (online)	**Scribe: Chi-Ning Chou**

1 Algorithms and Lower bounds

We can formulate the abstract definitions of algorithms and lower bounds using quantifiers as follows. For a function $f : \Sigma^* \to \Sigma^*$,

- Algorithm: $\exists$ an *efficient* algorithm A such that $\forall x \in \Sigma^*$, A outputs $f(x)$.
- Lower bound: $\forall$ *efficient* algorithm A, $\exists x \in \Sigma^*$ such that A does not output $f(x)$.

Intuitively, with this view, designing algorithms seems to be inherently easier than designing lower bound. However, if we slightly modified the statement for lower bound to be the following.

- Lower bound*: $\exists$ a proof P such that $\forall$ efficient algorithm A, A cannot solve f.

Namely, we can hope to get lower bound result by the design and analysis of an algorithm! However, for now we are still on a very high-level stage. To achieve this goal, the *algorithm* for our lower bound construction requires several steps informally listed as follows.

(1) The input of the algorithm is algorithms/programs. A natural candidate is circuits.

(2) The algorithm can determine some interesting *properties* of the input, *e.g.,* cannot solve f.

Thus, a natural problems for us to apply in the scenario above is the circuit analysis problems.

1.1 Circuit analysis problems

A circuit analysis problem treat the input as a circuit. The general form is as follows.

- Input: A circuit C.
- Output: If C enjoys property P.

Formally, we can define the property P as a set of circuit that enjoys the abstract notion of the property. A canonical example is $P := \{C : \ C \text{ is a nonzero function}\}$, which is the Circuit Satisfiability problem (Circuit Satisfiability).

1.2 [Karp-Lipton-Meyer '80]

A classic circuit lower bound constructed by circuit analysis algorithm was from a series of work of Karp, Lipton, and Meyer around 1980.

Theorem 1 (Karp-Lipton-Meyer '80) *If* $P = \mathbf{NP}$*, then* $\mathbf{EXP} \not\subset \mathbf{P/poly}$.

Namely, a good circuit algorithm (SAT $\in \mathbf{P}$), tells us the limitation of circuits ($\mathbf{EXP} \not\subset \mathbf{P/poly}$). However, we do not believe that the hypothesis ($\mathbf{P} = \mathbf{NP}$) is true

1.3 NEXP vs. P/poly

A way to prove $\mathbf{P} \neq \mathbf{NP}$ is to show that $\mathbf{NP} \not\subset \mathbf{P/poly}$ since $\mathbf{P} \in \mathbf{P/poly}$. However, the fact is that we even don't know whether **NEXP** is in **P/poly** or not! Nevertheless, a recent breakthrough of Williams used the strategy mentioned above to prove the following theorem.

Theorem 2 *For all polynomial p, if the satisfiability of circuits with n inputs and $p(n)$ size is decidable in $O(2^n/n^{10})$ time, then* $\mathbf{NEXP} \not\subset \mathbf{P/poly}$.

There are some intuitions about the proof of the theorem.

- Faster Circuit Satisfiability algorithms uncover a weakness in small circuits. Concretely, small circuit cannot *obfuscate* the all-zeros function!
- Faster Circuit Satisfiability algorithms show the *strength* of faster than 2^n algorithms. Concretely, there is nice algorithm that can efficiently tell whether a given circuit is all-zeros.
- It's like a game between algorithms and circuits! Namely, circuits want to hide the information about the function while algorithms try to uncover it.