CS 710: Complexity Theory 9/28/2017

Advice

Instructor: Dieter van Melkebeek Scribe: Shuo Yang, Yuxin Sun

DRAFT

In this lecture we talk about the concept of advice which is an extra input to a Turing machine that is allowed to only depend on the length n of the input, but not the input itself. There are a variety of classes such as P/poly, L/poly and NL/poly with some interesting conclusions. And then we move on the Karp-Lipton theorem which involves NP and P/poly. Finally, we introduce the notion of selector function of a language [1].

1 Advice and Nonuniform Computation Models

Definition 1 (Advice)**.** *Given complexity class $\mathcal{C}$ and advice size bound $a : \mathbb{N} \to \mathbb{N}$, define*

$$\mathcal{C}/a(n) = \left\{ L \mid (\exists L' \in \mathcal{C})(\exists \alpha_0, \alpha_1, \alpha_2, \ldots \in \Sigma^*) \text{ such that } x \in L \iff \langle x, \alpha_{|x|} \rangle \in L' \right\},$$

where $|\alpha_n| \leq a(n)$.

Based on the length n of the input, the advice function give an auxiliary information of each n. The bound on the length of the advice string only depends on the input size in the class $\mathcal{C}$. P/poly is the class of all languages that can be computed in polynomial time with polynomial-length advice. We may view that a Turing machine is given a single advice polynomial length string $\alpha(n)$ for all inputs of length n, or a infinite series of Turing machine and the nth Turing machine computes of input of length n.

P/poly has the relations with *boolean circuits*, which can used as an general and simplified abstract model of the basic part in modern computers.

Definition 2 (Circuit families)**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a function. A $T(n)$-size circuit family is a sequence $\{C_n\}_{n \in \mathbb{N}}$ of Boolean circuits, where C_n has n inputs and a single output, and its size $|C_n| \leq T(n)$ for every n.*

The boolean circuit is such kind of nonuniform model that allows a different algorithm to be used for each input size. While there is always the same Turing machine solving problems with infinite input sizes in the standard Turing machine model. The following result demonstrates the connection between the class P/poly with boolean circuits.

Proposition 1. *A language $L \in$ P/poly if and only if there exists a family of circuits $\{C_n\}$, where circuit C_n has n inputs and one output, and there exists a polynomial $p(\cdot)$ such that for all n, $|C_n| \leq p(n)$ and C_n decides language L on $x \in \{0,1\}^n$. In symbol,*

$$\text{P/poly} = \{L \mid \exists c \, \forall n \; C_L(n) < n^c\}$$

where $C_L(n)$ is the size of a smallest Boolean circuit that decides L at length n.

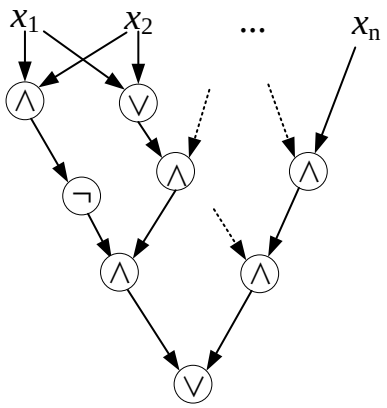

Figure 1: An illustration of a boolean circuit with n inputs and one output which computes a function $f : \{0,1\}^n \to \{0,1\}$. Each AND and OR gate is in-degree two and out-degree one and every NOT node has in-dgree one and out-degree one.

Proof. ($\Rightarrow$): If L is decidable by a polynomial-time Turing machine M with access to an advice family $\{\alpha(n)\}$ of size $p(n)$ for some polynomial p, then we can construct for every n a polynomial-sized circuit D_n such that on every $x \in \{0,1\}^n, \alpha \in \{0,1\}^{a(n)}, D_n(x,\alpha) = M(x,\alpha)$. We let the circuit C_n be the polynomial circuit that given x computes the value $D_n(x,\alpha_n)$. That is, C_n is equal to the circuit D_n with string α_n hard-wired as its second input.

($\Leftarrow$): Conversely, if L is decidable by a polynomial-sized circuit family $\{C_n\}$. We can just use the description of C_n as an advice string on inputs of size n, where the Turing Machine is simply the polynomial-time Turing Machine M that on input a string x and a string representing an n-input circuit C outputs $C(x)$. □

Similarly, for the complexity class NP/ poly, we have the following proposition.

Proposition 2. *A language $L \in$ NP/ poly if and only if there exists a family of nondeterministic circuits $\{C_n\}$, where nondeterministic circuit C_n has n inputs and one output, and there exists a polynomial $p(\cdot)$ such that for all n, $|C_n| \leq p(n)$ and C_n decides language L on $x \in \{0,1\}^n$. In symbol,*

$$\text{NP/ poly} = \{L \mid \exists c\, \forall n\ C_L(n) < n^c\}$$

where $C_L(n)$ is the size of a smallest nondeterministic boolean circuit that decides L at length n.

Proof. The proof is almost the same with the class P/ poly except the circuit family is nondeterministic. □

Next, we will discuss space-bounded computable languages with polynomial-length advice. Just as circuits are used to investigate time requirement of Turing machines, *branching programs* are used to as a combinatorial tool to investigate space complexity.

Definition 3 (Branching program)**.** *A branching program on n input variables $x_1, x_2, \ldots, x_n$ is a directed acyclic graph all of whose nodes of nonzero out-degree are labeled with a variable x_i. It has two nodes of out-degree zero that are labeled with an output value,* ACCEPT *or* REJECT*. The edges are labeled by* 0 *or* 1*. One of the nodes is designated the start node. A setting of the input*

variables determines a way to walk on the directed graph from the start node to an output node. At any step, if the current node has label x_i, then we take an edge going out of the node whose label agrees with the value of x_i. The branching program is deterministic if every nonoutput node has exactly one 0 *edge and one* 1 *edge leaving it. Otherwise it is nondeterministic. The* size *of the branching program is the number of nodes in it.*

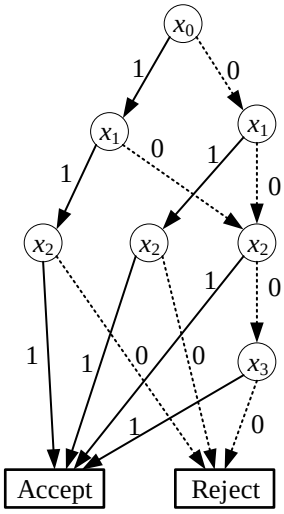

Figure 2: An illustration of a branching program. The root has in-degree zero and the leaves which have out-degree zero are labeled with accept or reject. Every non-leaf node has two outgoing edges which are labeled 0 and 1.

For the complexity class L/ poly, we have the following result.

Proposition 3. *A language $A \in$ L/ poly if and only if there exists a family of branching programs $\{BP_n\}$ where for each n, $BP_n : \{0,1\}^n \to \{0,1\}$, and there exists a polynomial $p(\cdot)$ such that $|BP_n| \leq p(n)$ and BP_n decides language A for all $x \in \{0,1\}^n$. In symbol,*

$$\text{L/ poly} = \{A \mid \exists c \, \forall n \; BP_A(n) < n^c\}$$

where $BP_A(n)$ is the size of the smallest branching program that decides a language A at length n.

Proof. ($\Rightarrow$): The proof is similar with the above. There exists a advice function α mapping an integer n to a string of length polynomial in n, and a Turing machine M with tapes of size logarithmic in the input size. Because L has a input x with length n, M decides L and accepts the input (x, α). The configuration of each step of M can be seen as the node in the branching program. At each node, the program look at the bit of input (0 or 1) corresponding to this node's label. It moves along the outgoing 0 edge to the next node, likewise take the 1 edge if the bit is a 1. The configuration state of M changes step by step also in this way. Because we just need to keep track of the current node, running an advice-encoded BP only requires logarithmic space.

($\Leftarrow$): If there exists a series of branching program of polynomial size exists, it can be specified by the advice function and simulated by the Turing machine. Running an advice-encoded BP only requires logarithmic space, because we just need to keep track of the current node. □

Similarly, we have the following proposition for complexity class NL/*poly*.

Proposition 4. *A language* $A \in \mathrm{NL}/\mathrm{poly}$ *if and only if there exists a family of nondeterministic branching program* $\{NBP_n\}$ *where for each* n, $NBP_n : \{0,1\}^n \to \{0,1\}$, *and there exists a polynomial* $p(\cdot)$ *such that* $|NBP_n| \leq p(n)$ *and* NBP_n *decides language* A *for all* $x \in \{0,1\}^n$. *In symbol,*

$$\mathrm{NL}/\mathrm{poly} = \{A \mid \exists c\, \forall n\ NBP_A(n) < n^c\}$$

where $NBP_A(n)$ *is the size of the smallest branching program that decides a language* A *at length* n.

2 Karp-Lipton Theorem

One of the motivation to define P/poly is to separate P from NP. If we can show there is a language which is in NP but not in P/poly, the like that $\mathrm{P} \neq \mathrm{NP}$. Karp-Lipton theorem shows that if $\mathrm{NP} \subseteq \mathrm{P}/poly$, the Polynomial-time hierarchy collapses to $\Sigma_2^{\mathrm{P}} \cap \Pi_2^{\mathrm{P}}$.

Theorem 1. *If* $\mathrm{NP} \subseteq \mathrm{P}/\mathrm{poly}$, *then* $\mathrm{PH} = \Sigma_2^{\mathrm{P}}$.

Proof. The assumption implies that there exists a series of polynomial circuits which decide SAT. To proof this theorem. We use the following key claims to prove. This key claim is unconditional and does not depend on the assumption of the above theorem.

Key Claim: Given a Boolean circuit C and a integer n, checking whether C decides SAT on inputs of length n is in coNP.

If $\mathrm{NP} \subseteq \mathrm{P}/\mathrm{poly}$, then there exists a series of polynomial size Circuits $\{C_n\}\,(n \in \mathrm{N})$ which decides SAT. For C_i in $\{C_n\}$, the size $|C_i|$ is bounded by $p(i)$ and i is just an given integer and p is a polynomial function. From the key claim, we know that checking a circuit whether decides SAT is in coNP, so the checking the following Π_2^{P}-SAT

$$\left(\forall x \in \Sigma^{|\varphi|^c}\right)\left(\exists y \in \Sigma^{|\varphi|^c}\right)\varphi(x,y) = 1$$

is Π_2^{P}-complete where φ is a boolean function and x and y are bounded by the length of φ. Based on the above formula, We have

$$\forall x \exists y\, \varphi(x,y) \iff \exists \{C_n\} \forall x\, C_{|\varphi(x,\cdot)|}\, decides\, \varphi(x,\cdot)$$

the latter describes a computation in Σ_2^{P}. So, $\Pi_2^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}}$ and PH collapses to the second level.

There are two approaches to proof the key claim. One is based self-reducibility and the other is based on search-to-decision for SAT.

(1) Based on self-reducibility of SAT: Consider a Turing machine M. M firstly checks the size of the Circuit C is polynomial-bounded by $p(n)$ where n is the length of the boolean formula. This process takes $O(n \cdot p(n))$. If the size of the formula $|\varphi| = 1$, M checks if C accepts φ. Otherwise, M choose deterministically choose some variables x from φ and and set the x to 0 or 1 to check

$$C(\varphi) = 1 \iff C(\varphi | x \leftarrow 0) = 1\ or\ C(\varphi | x \leftarrow 1) = 1$$

Based on the defination of coNP, M accepts C only if it dose not find any formula of length at most n for this test fails.

(2) Based search-to-decision reduction of SAT: Search-to-decision is a reduction from finding a satisfying assignment to deciding whether a given formula a satisfiable (SAT). It works by setting the values of the variables one by one. Given a circuit C which is supposed to compute SAT. This C may make mistakes. We can transform the C to a circuit $C\prime$ which has no false positives, i.e., it can only reject a satisfiable formula. We first substitute $x_1 = 0$ and $X_1 = 1$ in φ and plug in C to check which is acceptable while at least one of them is acceptable. Then fix the x_1 and change the x_2 to do the similar process. In the end, we can get a arrangement of the variables and ask whether this arrangement indeed satisfy the φ. If so, it accepts, otherwise it rejects. So, given an language that solves SAT on inputs of length smaller than n we can solve SAT on inputs of length n. This is the the downward self-reducibility of SAT. The rejection instances can be accepted in polynomial time by a non-deterministic Turing machine, so it is in coNP.

□

Exercises:

- $\mathrm{NP} \subseteq \mathrm{P}/O(\log n) \Rightarrow \mathrm{NP} = \mathrm{P}$;
- $\mathrm{NL} \subseteq \mathrm{P}/O(\log n) \Rightarrow \mathrm{NL} = \mathrm{L}$;
- $\mathrm{coNP} \subseteq \mathrm{NP}/poly \Rightarrow \mathrm{PH} = \Sigma_3^{\mathrm{P}}$.

3 Selector

For a given language L, instead of solving the decision problem of L for an arbitrary input x, we may be more interested: which one of two given input strings x and y is more likely to be in L? The notion of P-selective sets is presented to study the polynomial-time reproducibilities on NP. It plays a great role in researching several important structural concepts such as self-reducibility and reducing search to decision and function complexity classes [2].

Definition 4. *Let Σ be a finite alphabet. A language $L \subseteq \Sigma^*$ is said to be* P*-selective exactly if there is a function $f : \{\Sigma^*, \Sigma^*\} \to \Sigma^*$ which is computable in polynomial time such that*

- *$f(\{x, y\}) \in \{x, y\}$;*
- *if $\{x, y\} \cap L \neq \emptyset$, then $f(\{x, y\}) \in L$.*

The function f is said to be a P*-selector function for language L.*

Theorem 2. *The Language L has a selector function $\Rightarrow L \in \mathrm{P}/poly$.*

Notation: Let f denote the selector function. We write $x^* \to x$ if the selector selects x^* among x and x^*, i.e., $f(\{x, x^*\}) = x^*$. To be precise, we always include the self-loops $x^* \to x^*$.

Key idea: If a string x^* is not in L, we know that any string x such that $x^* \to x$ is also not in L. By taking up x^* in the advice, we can vouch for the non-membership of all those x's by evaluating f. By making sure there are many such x's, we only need consider a small number of x^* in order to be able to vouch for all the non-members.

Proof. Fix an input length n, and let X_0 denote the strings of length n in L, i.e., the intersection of $\{0,1\}^n$ and the complement of L. These are the strings we need to vouch for.

For x^* in X_0, let $V_0(x^*) = \{x \in X | x^* \to x\}$, which we refer to as the sets of inputs from X_0 "vouched for" by x^*.

The arrow relation defines a tournament on X_0, i.e., a digraph in which between any two distinct vertices precisely one of the two directed edges is presents. As all self-loops are also present, the average out-degree equals $|X_0|/2$. Thus, there exists at least one x_0^* in X_0 such that $|V_0(x_0^*)| \geqslant |X_0|/2$. We conclude x_0^* in the advice.

We denote by X_1 the inputs from X_0 that we still need to vouch for. Note that $|X_1| \leqslant |X_0|/2$. By the same argument above, there exists a string x_1^* that vouches for half of X_1. We include such a string x_1^* in the advice.

We keep repeating this process until there are no more strings to be vouched for. As $|X_i| \leqslant |X_{i-1}|/2$, and $|X_0| \leqslant 2^n$, we know that the number of steps s is no more than $n+1$.

To the end, we have that for an input $x \in 0, 1^n$, $x \in L$ if and only if there exists i in $\{0, 1, 2, ...s\}$ such that $x_i^* \to x$. Given the x_i^* 's as advice, the predicate can be evaluated in time $poly(n)$. The advice itself is of length $s_n = O(n^2)$. This finishes the proof.

□

4 Next Time

In the next lecture, we are going to talk about the kernelization lower bound. The statement of the kernelization lower bound is: Vertex Cover does not have kernels consisting of $O(k^{2-\epsilon})$ edges for any positive constant ϵ unless coNP is in NP/ poly. Recall that Vertex Cover is the problem of deciding whether, given a graph G and an integer k, G has a vertex cover of size at most k. A kernel is a mapping reduction from a parameterized problem to itself such that the size of the reduced instance is bounded by a function of the parameter only. Kernelization gives us quantitative insights in what can be achieved by polynomial time preprocessing.

5 Acknowledgements

In writing the nodes for this lecture, we perused the notes by David Guild for Lecture 9 from the Spring 2011 offering of CS 710.

References

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[2] Alan L Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. In *International Colloquium on Automata, Languages, and Programming*, pages 546–555. Springer, 1979.