CS 810: Introduction to Complexity Theory 9/18/2003

Lecture 11: P/poly, Sparse Sets, and Mahaney's Theorem

Instructor: Jin-Yi Cai Scribe: Aparna Das, Scott Diehl, Giordano Fusco

1 From previous lectures

Last time we proved the Karp-Lipton Theorem:

THEOREM 1.1 (KARP-LIPTON) *SAT has polynomial size circuits if and only if the polynomial hierarchy collapses to* $\Sigma_2^{\mathrm{P}} \cap \Pi_2^{\mathrm{P}}$.

2 P/poly

We now introduce the class P/poly:

DEFINITION 2.1 *A set* A *is in* P/poly *if* $\exists B \in P$ *and an "advice" function* $s : \mathbb{N} \to \Sigma^*$ *such that* $|s(n)| = n^{O(1)}$ *and* $x \in A \Leftrightarrow \langle x, s(|x|)\rangle \in B$.

In other words, the advice function takes the length of the string x as input and produces a string polynomial in $|x|$ that allows us to decide membership in A in polynomial time.

Note if $s(n)$ outputs the truth table for strings of length n, we can obviously decide membership of x. However, in this case $|s(n)| = 2^n > n^{O(1)}$, so it would not be an admissible advice function for P/poly.

We now show the equivalence between P/poly and languages with polynomial size circuits.

THEOREM 2.2 $L \in$ P/poly $\Leftrightarrow$ *L has a polynomial sized circuit.*

Proof. First we show the forward direction. Since $L \in$ P/poly there is exists a $B \in P$ and an "advice" function $s(n)$ as in the above definition. Since $B \in P$, it has a polynomial sized circuit that accepts input pairs $\langle x, s(|x|)\rangle$. We fix the value of $s(n)$ in B's circuit, so that B is hard-coded to accept strings x of length n if and only if $x \in L$. Now we can use such circuits to accept L. The size of each circuit is $(n + |s(n)|)^{O(1)} = n^{O(1)}$.

Now we show the converse. Suppose L has polynomial sized circuits. Then we define $s(n)$ to be the binary encoding for that circuit at length n. Then the Turing Machine M which on inputs $\langle x, s(|x|)\rangle$ evaluates the output of the circuit given by $s(|x|)$ on x runs in polynomial time and accepts x if and only if the circuit given by $s(|x|)$ accepts x. Therefore $x \in L \Leftrightarrow \langle x, s(|x|)\rangle \in L(M)$, where $L(M) \in P$, so $L \in \mathrm{P/poly}$.

The proof above shows that saying a language is in P/poly is equivalent to saying that it has a polynomial sized circuit. Therefore we can restate the Karp-Lipton theorem as follows:

THEOREM 2.3 *If* $\mathrm{NP} \subseteq \mathrm{P/poly}$ *then the polynomial hierarchy collapses to* $\Sigma_2^{\mathrm{P}} \cap \Pi_2^{\mathrm{P}}$.

3 Sparse sets

Now we are going to introduce another instrument called *sparse sets* that is also equivalent to P/poly.

DEFINITION 3.1 *A set* $S \subseteq \Sigma^*$ *is called a sparse set if* $|S^{=n}| = n^{O(1)}$*, i.e. if the number of strings at length* n *in* S *is at most polynomial in* n.

Note that the definition could also be given as $|S^{\leq n}| = n^{O(1)}$, which is equivalent.

There are a lot problems that can be described using sparse sets. For example, the halting problems can be made very sparse. The power of this instrument is that sparse problems are combinatorially very simple. A sparse set, S, can be thought of as a set of low-information content.

THEOREM 3.2 *A language* L *has a polynomial size circuit if and only if* $L \in P^S$ *for some sparse set* S*. More concisely:*

$$L \in P/poly \iff L \leq_T^{\mathrm{P}} S \text{ for some sparse set } S$$

Proof. [of $\Leftarrow$] Let S be a sparse set and M be a polynomial time TM which uses S as an oracle. L is the language recognized by this machine. When the length of the input is n, the maximum length string that can be queried is of size $n^{O(1)}$. In S, there are at most $(n^{O(1)})^{O(1)} = n^{O(1)}$ strings with length less than the maximum query length. So we can construct the advice function, $s(n)$ to list all strings in S that M could query on input length n. Therefore $|s(n)| = n^{O(1)}$, and we can construct a Turing Machine M' that on input $\langle x, s(n)\rangle$ does as M does on x, except instead of querying S, M' simply scans $s(n)$ to find the query answers.

Proof. [of $\Rightarrow$.] Let L be in $P/poly$. This means that some polynomial time Turing Machine N accepts strings $\langle x, s(n)\rangle$ if and only if $x \in L$. We want to design a sparse set S and a machine

M so that M can discover $s(n)$ in polynomial time using S as an oracle. If we can do this, then afterwards M can simply simulate N (since it now knows $s(n)$), so that $L(M^S) = L$.

Consider the language $S = \{1^n\#p \mid p \text{ is a prefix of } s(n)\}$. It is clear that S is sparse (in fact there are at most linearly many strings of a given length in S). Using S as an oracle, we find $s(n)$ one bit at a time: first, ask to the oracle if the strings $1^n\#0$, $1^n\#1 \in S$. Let $1^n\#b$ be the string out of these two which is in S. Then we can extend it to find another bit of $s(n)$ by asking which of $1^n\#b0$ or $1^n\#b1$ is in S. We proceed in this manner until neither extension of our string is in S. Then when this happens, we must have $s(n)$. $s(n)$ has polynomially many bits, so this can be done in polynomial time.

Now that we have proved another definition equivalent to polynomial sized circuits, we can restate the Karp-Lipton theorem again:

THEOREM 3.3 *If* SAT $\leq_T^p$ S *for some sparse set* S*, then the polynomial hierarchy collapses to* $\Sigma_2^p \cap \Pi_2^p$.

4 Mahaney's Theorem

This raises the question: What if SAT is *Karp* reducible to a sparse set S? i.e. $SAT \leq_m^p S$ for some sparse set S. This result is given by Mahaney's Theorem, and we will give a proof due to Ogihara and Watanabe.

THEOREM 4.1 (MAHANEY'S THEOREM) $SAT \leq_m^{\mathrm{P}} S$ *for some sparse set* $S \neq \emptyset$ *if and only if* NP = P

Proof. [Ogihara and Watanabe]

- $\Leftarrow$ This direction is trivial: If NP = P then our sparse set S can simply be $\{0\}$ and the reduction simply maps satisfiable formulas to $\{0\}$ and unsatisfiable formulas to $\{1\}$ (which is polynomial time computable by assumption).
- $\Rightarrow$ Now assume that SAT $\leq_m^p S$ for some sparse set S. Then we will show how to solve SAT in polynomial time. To do this we will work with the language of *left cut* SAT:

  DEFINITION 4.2 $L_{\mathrm{SAT}} = \{\langle \varphi, \sigma \rangle \mid \varphi$ *is a boolean function on* $x_1, \ldots, x_n$ *and* σ *is a partial assignment to* $x_1, \ldots, x_n$ *such that there is some* $\sigma' \preceq_l \sigma$ *where* $\varphi|_{\sigma'} = T\}$

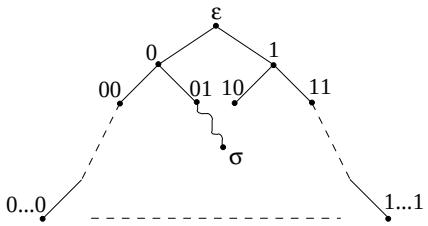
  The $\preceq_l$ relation can be thought of as the "to the left of" function in the tree of partial truth assignments under the left-right-root ordering. More formally:

DEFINITION 4.3 *Let $\sigma = \sigma_1\sigma_2\cdots\sigma_m$ and $\tau = \tau_1\tau_2\cdots\tau_k$ be binary strings representing partial truth assignments to φ. Then we say that σ is to the left of τ (and write $\sigma \preceq_l \tau$) if either σ is an extension of τ or in the bit immediately before the first entry they differ at, σ has bit 0 and τ has bit 1. In other words, let $i = max\{j \mid 0 \leq j \leq min\{k, m\},\ (\forall j',\ 1 \leq j' \leq j)[\sigma_{j'} = \tau_{j'}]\}$. Then $\sigma \preceq_l \tau$ if either $i = k$ or $i < k$ with $\sigma_{i+1} = 0$ and $\tau_{i+1} = 1$.*

So L_{SAT} can be seen to consist of pairs of boolean formulas and partial assignments such that the formula can be satisfied by an assignment to the left of the one given.

Clearly $L_{\text{SAT}} \in$ NP since given φ and σ, we can guess a truth assignment τ, and check that τ satisfies φ and $\tau \preceq_l \sigma$ in polynomial time. Since SAT is NP-complete, then we must have that $L_{\text{SAT}} \leq_m^p$ SAT. We can then compose Karp reductions to see that $L_{\text{SAT}} \leq_m^p S$.

Let $f(\langle\varphi, \sigma\rangle)$ the the function that computes this reduction. Clearly $|f(\langle\varphi, \sigma\rangle)| \leq |\varphi|^{O(1)}$ (since the length of a partial truth assignment σ can be at most the number of variables in φ, so $\langle\varphi, \sigma\rangle$ is no more than double $|\varphi|$). Similarly there are polynomial many strings in S of length at most $|\varphi|^{O(1)}$. Therefore there are at most $(|\varphi|^{O(1)})^{O(1)} = |\varphi|^{O(1)}$ strings in S of lengths that f can produce on formula φ. Let N be this number.



Now consider the binary tree of partial assignments to the variables of φ. We will find a satisfying truth assignment (if one exists) by exploring the tree while maintaining at most N nodes per level via pruning. If we can do this, then since there are n levels of the tree and N is polynomial in $|\varphi|$, we will have a $n \cdot N^{O(1)} = |\varphi|^{O(1)}$ time bound for our algorithm. Therefore SAT $\in$ P, so P = NP.

We will show that a satisfying truth assignment can be found in such a manner by induction. At level i of the tree, we have at most N nodes (partial truth assignments), and if φ is satisfiable, then at least one node can be extended to a "leftmost" satisfying truth assignment for φ (i.e. a satisfying truth assignment σ such that for every other satisfying truth assignment τ, $\sigma \preceq_l \tau$).

- Initially (at $i = 0$) this is trivially true since the only node is ϵ.
- Now assume it is true for levels $i \leq k$. Let r be the number of nodes at level k, giving us partial truth assignments $\sigma_1, \ldots, \sigma_r$. Then $r \leq N$ by assumption, but the number of nodes that can be produced for level $k+1$ is $2r$ and may be more than N. If this is the case, then we will form $2r$ query strings $s_1, \ldots, s_{2r}$ of the form $f(\langle\varphi, \sigma_i'\rangle)$, where σ_i' for $1 \leq i \leq 2r$ are the possible extensions of σ_i for $1 \leq i \leq r$. We will look for duplicates amongst such query strings in order to prune our tree.

  Suppose we have duplicates from nodes σ and τ – $f(\langle\varphi, \sigma\rangle) = f(\langle\varphi, \tau\rangle)$. Note that because their images under f are the same, and f reduces L_{SAT} to S, either both $\langle\varphi, \sigma\rangle$

and $\langle \varphi, \tau \rangle$ are in L_{SAT} or both are not. However, one must be "left" ($\preceq_l$) of the other. Therefore the right one (without loss of generality we can assume this is τ) cannot extend to the "leftmost" satisfying assignment for φ (if one exists). Therefore we can safely prune the node τ.

After pruning the right duplicates under f, we might still have more than N nodes. In this case, we can then iteratively prune the leftmost truth assignment σ until we have N nodes left. To see that this works, suppose some such σ could be extended to the leftmost truth assignment for φ. This would then imply that all pairs $\langle \varphi, \tau \rangle$ for τ to the right of σ would be in L_{SAT}. Since all the images of the nodes under f are distinct at this point, this would give us more than N distinct elements of S. However, N is defined to be the maximum number of strings of S that f can produce, so this gives us a contradiction. Therefore, after pruning, we have only N nodes at level $k+1$, and if there is a satisfying truth assignment for φ, at least one node can be extended to the leftmost truth assignment.

Therefore SAT can be solved in polynomial time by iteratively filling out and pruning the levels of this tree until full truth assignments are generated. We can then test to see if any satisfy φ, and if so, accept. Otherwise, reject. If φ is satisfiable, the leftmost satisfying assignment will be amongst the truth assignments tested, and we will accept. If φ was not satisfiable, then obviously no satisfying truth assignments will be generated, so we will reject. This shows that we can solve SAT in this manner in polynomial time, which completes the proof.