18.405J/6.841J Advanced Complexity Theory March 11, 1997

Lecture 10

Lecturer: Daniel A. Spielman *Scribe: Daniel Lewin*

Toda's Theorem

Theorem 1 $PH \subseteq P^{\#P}$

We will present a proof of Toda's theorem (simpler than the original) that makes use of two new objects: a class of functions called $\mathsf{Gap-P}$ and a class of circuits called AC_0. In this lecture we will define these objects and prove the properties needed for the proof of Toda's theorem.

1 Gap−P

Let M be a non-deterministic Turing Machine. We denote by $\#M(x)$ the number of accepting computation paths of M on input x, and by $\#\overline{M}(x)$ the number of rejecting computation paths.

Definition 2 *If M is a non-deterministic Turing Machine, we define the function $gap_M : \sum^* \to Z$ as:*

$$gap_M(x) = \#M(x) - \#\overline{M}(x)$$

gap_M represents the gap between the number of accepting computation paths and the number of rejecting computation paths of M.

Definition 3

$$\mathsf{Gap-P} = \{\mathsf{gap_M} \mid M \textit{ is a non-determenistic polynomial time Turing Machine}\}$$

In other words, $\mathsf{Gap-P}$ *is the set of functions that are gap_M for some non-deterministic polynomial time Turing Machine.*

Intuitively, the difference between the classes $\#P$ and $\mathsf{Gap-P}$ is that unlike $\#P$, $\mathsf{Gap-P}$ is closed under *subtraction*. In addition, $\mathsf{Gap-P}$ has many other natural closure and composition properties that $\#P$ does not posess.

The reason that we are interested in $\mathsf{Gap-P}$ in the context of Toda's theorem is the following observation which is easy to prove [1].

Observation 4 $P^{\mathsf{Gap-P}} = P^{\#P}$

Proposition 5 *If $f, g \in \mathsf{Gap-P}$ then:*

1. $-f \in \mathsf{Gap-P}$
2. $f + g \in \mathsf{Gap-P}$
3. $f \cdot g \in \mathsf{Gap-P}$
4. $\sum_{|y| \leq p(|x|)} f(x, y) \in \mathsf{Gap-P}$
5. $\prod_{y \leq p(|x|)} f(x, y) \in \mathsf{Gap-P}$ *(note that y is taken to be an integer in binary representation)*

Proof

Let M_f and M_g be the non-det poly-time TM's such that $f = gap_{M_f}$ and $g = gap_{M_g}$.

1. Just swap accepting and rejecting states in M_f.
2. The machine M_{g+f} makes a non-deterministic decision at the first step to simulate M_f or to simulate M_g on the given input. $gap_{M_{f+g}} = \#M_f + \#M_g - \#\overline{M}_f - \#\overline{M}_g = gap_{M_f} + gap_{M_g} = f + g$.
3. The machine $M_{g \cdot f}$ works as follows:
   - Simulate M_f
   - If M_f accepts, simulate M_g and accept iff M_g accepts.
   - If M_f rejects, simulate M_g and accept iff M_g rejects.

   We have: $gap_{M_{f \cdot g}} = \#M_{f \cdot g} - \#\overline{M}_{f \cdot g} = (\#M_f \cdot \#M_g + \#\overline{M}_f \cdot \#\overline{M}_g) - (\#M_f \cdot \#\overline{M}_g + \#M_g \cdot \#\overline{M}_f) = (\#M_f - \#\overline{M}_f) \cdot (\#M_g - \#\overline{M}_g) = gap_{M_f} \cdot gap_{M_g} = f \cdot g$.

   Note that each time we carry out this multiplication operation the running time of the resulting machine doubles. Thus, this construction can only be carried out a polynomial number of times.
4. Non-deterministically choose y and then simulate M_f on (x, y).
5. Iterate the multiply operation. Note that since y is taken to be the binary representation of an integer and we require $y < p(|x|)$, we only multiply a polynomial number of times.

■

2 AC_0

Definition 6 *AC_0 is the class of predicates computable by (languages accepted by) uniform,* constant *depth, polynomial sized circuits with NOT gates and* unbounded *fan-in AND and OR gates.*

Definition 7 *[2] A* probabilistic polynomial *is a polynomial in two types of variables: actual variables $x_1, \ldots, x_n$ from, $\{0,1\}$, and probabilistic variables $r_1, \ldots, r_l$ where each r_i is chosen uniformly and independently from $\{0,1\}$. If P is a probabilistic polynomial, then for any given $x_1, \ldots, x_n$ we think of $P(x_1, \ldots, x_n)$ as a random variable. Let P be a probabilistic polynomial and let f be a function in n variables. P* computes f with error ϵ *if for every $x_1, \ldots, x_n$, the probability that $P(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ is at least $1 - \epsilon$.*

We will show that every $f(x_1, \ldots, x_n)$ in AC_0 can be computed with small error by a low degree probabilistic polynomial with small sample space (with a poly-logarithmic number of probabilistic variables).

Lemma 8 *For every family $\{A_n\}$ in AC_0 there exists a family of probabilistic polynomials $\{p_n\}$ over Z such that: p_n has degree $\log(n)^{O(1)}$, has $\log(n)^{O(1)}$ probabilistic variables, and computes A_n with error at most $1/3$. In other words:*

$$\forall x : Pr[A_n(x) = P_n(x, \vec{r})] \geq \frac{2}{3}$$

Where the probability is over $\vec{r} \in \{0,1\}^{\log(n)^{O(1)}}$.

We make use of the results from [3]. In particular the following theorem (which was saw in class):

Theorem 9 *[3] Let $S \subseteq \{0,1\}^k$ be non-empty. Suppose $r_0, r_1, \ldots, r_k$ are randomly chosen from $\{0,1\}^k$. Let $S_0 = S$ and let S_i be:*

$$\{v \in S : v \cdot r_0 \equiv v \cdot r_1 \equiv \ldots v \cdot r_i \equiv 0 \ (mod\ 2)\}$$

for each $i \in \{0, \ldots, k\}$. Then

$$Pr[|S_i| = 1 \ for\ some\ i \in \{0, \ldots, k\}] \geq \frac{1}{4}$$

We begin by showing that such a family of polynomials exists for unbounded fan-in NOR gates.

Lemma 10 *Let $\{A_n\}$ be the family of unbounded fan-in NOR gates (A_n has n inputs). There exists a family of probabilistic polynomials $\{p_n\}$ over Z such that: p_n has degree $\log(n)^{O(1)}$, has $\log(n)^{O(1)}$ probabilistic variables, and computes A_n with error at most $1/3$.*

Proof Let $x_1, \ldots, x_n$ be the inputs to A_n, and let $k = \lceil \log(n) \rceil$. Consider elements of $\{0,1\}^k$ as indexes to the inputs. For $\vec{j} \in \{0,1\}^k$ and a vector of k probabilistic variables $\vec{r}$ let $VV(\vec{j}, \vec{r})$ be the polynomial computing the function which is 1 if $\vec{j} \cdot \vec{r} = 0 \ (mod\ 2)$ and 0 otherwise. (This polynomial is the arithmatization of the boolean expression $r_{t_1} \oplus r_{t_2} \oplus \ldots \oplus r_{t_g} \oplus 1$ where the t's are the non-zero bit indices of $\vec{j}$) Let $S = \{\vec{j} \in \{0,1\}^k : x_{\vec{j}} = 1\}$, and let the S_i's be defined as in Theorem 9 where the bits of each $\vec{r_i}$ are taken from different sets of probabilistic inputs. The important point is that $|S_i|$ can be computed by the following probabilistic polynomial:

$$S_i(x_1, x_2, \ldots, x_n, \vec{r_1}, \vec{r_2}, \ldots, \vec{r_k}) = \sum_{\vec{j} \in \{0,1\}^k} x_{\vec{j}} \left(\prod_{l=1}^{i} VV(\vec{j}, \vec{r_l}) \right)$$

Where the $\vec{r}$'s are sets of $k = \log(n)$ probabilistic variables. Note that the number of probabilistic variables is k^2. Consider the polynomial

$$T(x_1, x_2, \ldots, x_n, \vec{r_1}, \vec{r_2}, \ldots, \vec{r_k}) = \prod_{i=1}^{k} (1 - S_i(x_1, x_2, \ldots, x_n, \vec{r_1}, \vec{r_2}, \ldots, \vec{r_k}))$$

T has real variables $x_1, \ldots, x_n$ and k^2 probabilistic variables.

Now, if all the x_i's are zero then $|S_i| = 0$ for all i, so $T = 1$. On the other hand, if at least one x_i is 1, then by Theorem 9, we have that some $|S_i| = 1$, and thus $T = 0$ with probability at least $1/4$. We can reduce the error probability from $3/4$ to $1/3$ by constructing a *constant* number of polynomials that use different probabilistic variables and multiplying them together. Thus, the total number of probabilistic variables is $\log(n)^{O(1)}$, and the degree of the polynomial also turns out to be $\log(n)^{O(1)}$.

■

References

[1] Stephen A. Fenner, Lance J. Fortnow, Stuart A. Kurtz, Gap-Definable Counting Classes, http://www.cs.uchicago.edu/~fortnow/gaps.ps.Z

[2] Richard Beigel, Nick Reingold, Daniel Spielman, The Perceptron Strikes Back, http://www-math.mit.edu/ spielman

[3] L.G. Valiant V.V Vazirani, NP is as Easy as Detecting Unique Solutions, Theoretical Computer Science 47 (1986) 85-93