Lecture 7

Toda’s Theorem

April 20, 2004
Lecturer: Paul Beame
Notes: Tian Sang

Definition 7.1. If P is any predicate define

$$\begin{aligned} \oplus^k y.\, P(y) &= |\{y \in \{0,1\}^k : P(y)\}| \bmod 2, \\ \#^k y. P(y) &= |\{y \in \{0,1\}^k : P(y)\}| = \sum_{y\in\{0,1\}^k} P(y), \\ \text{Я}^k y.\, P(y) &= \frac{|\{y \in \{0,1\}^k : P(y)\}|}{2^k} \end{aligned}$$

Definition 7.2. For complexity class C, define

$$\begin{aligned} L \in \oplus\cdot\mathsf{C} \quad &\Leftrightarrow \quad \text{there is a relation } R \in \mathsf{C} \text{ and polynomial } p \text{ such that} \\ & \qquad L = \{x \mid\ \oplus^{p(|x|)}\, y.\, R(x,y)\} \\ L \in \mathsf{BP}\cdot\mathsf{C} \quad &\Leftrightarrow \quad \text{there is a relation } R \in \mathsf{C} \text{ and polynomial } p \text{ such that for some } \epsilon < 1/2 \\ & \qquad \begin{cases} \text{Я}^{p(|x|)} y.\, R(x,y) \ge 1-\epsilon & \text{for } x \in L \\ \text{Я}^{p(|x|)} y.\, R(x,y) \le \epsilon & \text{for } x \notin L \end{cases} \\ L \in \mathsf{R}\cdot\mathsf{C} \quad &\Leftrightarrow \quad \text{there is a relation } R \in \mathsf{C} \text{ and polynomial } p \text{ such that for some } \epsilon < 1 \\ & \qquad \begin{cases} \text{Я}^{p(|x|)} y.\, R(x,y) \ge 1-\epsilon & \text{for } x \in L \\ \text{Я}^{p(|x|)} y.\, R(x,y) = 0 & \text{for } x \notin L \end{cases} \\ L \in \mathsf{P}\cdot\mathsf{C} \quad &\Leftrightarrow \quad \text{there is a relation } R \in \mathsf{C} \text{ and polynomial } p \text{ such that} \\ & \qquad L = \{x \mid \text{Я}^{p(|x|)} y.\, R(x,y) > 1/2\} \end{aligned}$$

Theorem 7.1 (Toda). $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathrm{PERM}}$.

As outlined in the last lecture this is done in two steps.

Lemma 7.2 (Toda). $\mathsf{PH} \subseteq \mathsf{BP}\cdot\oplus\mathsf{P}$.

Lemma 7.3 (Toda). $\mathsf{BP}\cdot\oplus\mathsf{P} \subseteq \mathsf{P}\cdot\oplus\mathsf{P} \subseteq \mathsf{P}^{\#\mathsf{P}}$.

As a warm-up we finish the proof of the following lemma which provides the key ideas for the proof of Lemma 7.2.

Theorem 7.4 (Valiant-Vazirani, Toda). $\mathsf{NP} \subseteq \mathsf{R}.\oplus\mathsf{P} \subseteq \mathsf{RP}^{\oplus\mathsf{P}}$.

Proof. To prove Theorem 7.4, as outlined in the last lecture we convert an NP problem to equivalent problem for which there is precisely one solution (and therefore the number of solutions will be odd). This will be accomplished by adding randomly chosen linear constraints. We use the following slightly weakened version of a lemma due to Valiant and Vazirani.

Lemma 7.5 (Valiant-Vazirani). *If $\emptyset \neq S \subseteq \mathbb{F}_2^n$, then for $v_1, \cdots v_{n+1} \in_R \mathbb{F}_2^n$,*

$$\Pr[\exists i \in \{1, \cdots n\}.\ |S \cap \langle v_1, v_2, \ldots, v_{i+1}\rangle^{\perp}| = 1] > \frac{1}{8}.$$

$A^{\perp}$ means orthogonal complement of A, meaning the subspace that consists of all vectors which when dotted with any vector from A produce 0 (that is, $A^{\perp} = \{ x \mid x \cdot y = 0, \ \ \forall y \in A \}$.

This follows immediately from the following lemma.

Lemma 7.6. *Fix a set $S \subseteq \mathbb{F}_2^n$, then for $v_1, \ldots, v_{n+1} \in_R \mathbb{F}_2^n$,*

(a) if $0^n \in S$, then $\Pr[\ |S \cap \langle v_1, v_2, \ldots, v_{n+1}\rangle^{\perp}| = 1\ ] > \frac{1}{2}$

(b) if $0^n \notin S$, and $2^{i-1} \leq |S| \leq 2^i$ then $\Pr[\ |S \cap \langle v_1, v_2, \ldots, v_{i+1}\rangle^{\perp}| = 1\ ] > \frac{1}{8}$

Proof. We first show part (a). Since we always have $0^n \in \langle v_1, v_2, \ldots, v_{i+1}\rangle^{\perp}$, if $0^n \in S$ then $0^n \in S \cap < v_1, v_2, \ldots, v_i >^{\perp}$. For any $x \in \mathbb{F}_2^n$, if $x \neq 0^n$ we have for any j that $\Pr[\ v_j \cdot x = 0\ ] = 1/2$. Therefore, since the v_j are chosen independently, $\Pr[\ v_1 \cdot x = v_2 \cdot x = \ldots = v_{n+1} \cdot x = 0\ ] = \frac{1}{2^{n+1}}$. Thus

$$\begin{aligned} \Pr[\exists x \in S - \{0^n\}, x \in \langle v_1, v_2, \ldots, v_{n+1}\rangle^{\perp}] &\leq \sum_{x \in S - \{0^n\}} \Pr[x \in \langle v_1, v_2, \ldots, v_{n+1}\rangle^{\perp}] \\ &= \frac{|S| - 1}{2^{n+1}} < 1/2. \end{aligned}$$

It follows that with probability greater than $1/2$, 0^n is the only element of $S \cap \langle v_1, v_2, \ldots, v_{n+1}\langle^{\perp}$ which means that $\Pr[\ |S \cap \langle v_1, v_2, \ldots, v_{n+1}\rangle^{\perp}| = 1\ ] > 1/2$.

We now prove part (b). Suppose that $0^n \notin S$ and $2^{-1} \leq |S| \leq 2^i$. Define $h(x) = (v_1 \cdot x, \cdots, v_{i+1} \cdot x) \in \mathbb{F}_2^{i+1}$. As in the argument for part (a), for $x \neq 0^n$, $\Pr[h(x) = 0^{i+1}] = 1/2^{i+1}$. An alternative way of viewing this probability statement is to view the condition that $h(x) = 0$ as a system of linear equations whose variables are the coordinates of the v_j vectors and whose coefficients are given by the coordinates of x. For $x \neq 0$, each equation $v_j \cdot x = x \cdot v_j = 0$ adds an additional independent constraint and therefore the dimension of the solution space drops by 1 for each j. In total, there are $i+1$ linearly independent equations in the v_j so the solution space is a $2^{-(i+1)}$ fraction of all possible vectors.

Suppose now that $x \neq y$ and $x, y \neq 0^n$, Then the condition that $h(x) = h(y) = 0^{i+1}$ is given by $2(i+1)$ equations whose coefficients are given by the coordinates of x and y. Since $x \notin \{0^n, y\}$ and $y \neq 0^n$, x and y are linearly independent and thus all of the $2(i+1)$ equations given by $h(x) = h(y) = 0^{i+1}$ are linearly independent. Therefore $\Pr[h(x) = h(y) = 0^{i+1}] = 1/2^{2(i+1)}$ for $x, y \in S$, $x \neq y$. Thus

$$\begin{aligned} \Pr[\exists y \in S - \{x\}.\ h(x) = h(y) = 0^{i+1}] &\leq \sum_{y \in S - \{x\}} \Pr[h(x) = h(y) = 0^{i+1}] \\ &= \frac{|S| - 1}{2^{2(i+1)}} \\ &< \frac{1}{2^{i+2}} \qquad \text{since } |S| \leq 2^i. \end{aligned}$$

Therefore

$$\begin{aligned} &\Pr[h(x) = 0^{i+1} \text{ and } \forall y \in S - \{x\}.\ h(y) \neq 0^{i+1}] \\ &\quad= \quad \Pr[h(x) = 0^{i+1}] - \Pr[\exists y \in S - \{x\}.\ h(x) = h(y) = 0^{i+1}] \\ &\quad> \quad \frac{1}{2^{i+1}} - \frac{1}{2^{i+2}} = \frac{1}{2^{i+2}}. \end{aligned}$$

Taking the union of these events, which are disjoint, over all choices of $x \in S$,

$$\begin{aligned} \Pr[\exists x.\ h(x) = 0^{i+1} \text{ and } \forall y \in S - \{x\}.\ h(y) \neq 0^{i+1}] \quad &> \quad \frac{|S|}{2^{i+2}} \\ &\geq \quad \frac{2^{i-1}}{2^{i+2}} = \frac{1}{8} \qquad \text{since } |S| \geq 2^{i-1} \end{aligned}$$

as required. □

We now prove Theorem 7.4 that $\mathsf{NP} \subseteq \mathsf{R} \cdot \oplus\mathsf{P}$. The key difference between showing this and showing that $\mathsf{NP} \in \mathsf{RP}^{\oplus\mathsf{P}}$ is the requirement that the algorithm make only one query to the $\oplus$P oracle and return the answer as its output. In order to do this we begin with a different basic experiment from the one outlined at the end of last lecture. Instead of trying all possible values of i we choose i at random. With probability at least $1/n$ this choice of i will be the correct i for Lemma 7.5. Here is the basic experiment E on input φ:

1. choose $i \in_R \{1, \ldots, n\}$, and $v_1, \ldots, v_{i+1} \in_R \mathbb{F}_2^n$
2. query the $\oplus$SAT oracle on $\varphi \wedge \varphi_{v_1} \wedge \cdots \wedge \varphi_{v_{i+1}}$ where φ_v is a formula that is satisfied by x iff $v \cdot x = 0$.
3. accept iff the oracle accepts.

Observe that if φ has n variables then

- if φ is unsatisfiable then $\Pr[\ E \text{ accepts } \varphi] = 0$, and
- if φ is satisfiable then $\Pr[\ E \text{ accepts } \varphi] > \frac{1}{8n}$.

This yields a randomized algorithm (with one call to an $\oplus$P oracle) for SAT with 1-sided error but its success probability is too low. To boost its success probability we make m independent trials of E. Each trial chooses an integer i in $\{1, \ldots, m\}$ and sequence of $i + 1$ vectors.

Let $r_1 = (i_1, v_1^1, \ldots, v_{i_1}^1)$ through $r_m = (i_1, v_1^m, \ldots, v_{i_m}^m)$ be the sequence of random choices of the independent trials of experiment E. Therefore

$$\Pr[\text{ all } m \text{ experiments fail}] \leq (1 - \frac{1}{8n})^m \leq e^{-\frac{m}{8n}} \leq e^{-2n} \qquad \text{for } m = 16n^2.$$

Thus

$$\varphi \in \text{SAT} \Leftrightarrow \text{Я} r_1, \ldots, r_m \underbrace{\exists j \in [16n^2].\ \varphi_{r_j} \in \oplus SAT} \geq 1 - e^{-2n}$$

where φ_{r_j} is the variant of φ created by added the linear constraints given by random string r_j. In comparison with trying all values of i, this does not seem to have helped much (except to reduce the error) because there are now $16n^2$ calls to the $\oplus$P oracle. We will concentrate on modifying the part of formula marked with the underbrace so that only one such call is required.

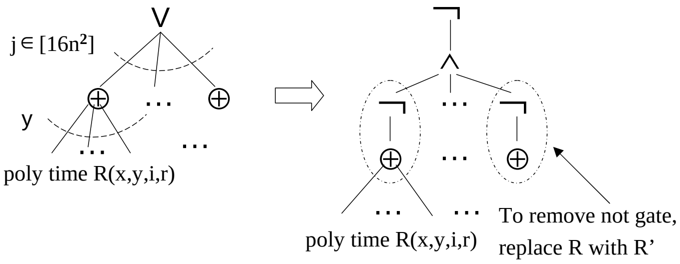

Figure 7.1: Converting from OR ($\exists$) to AND ($\forall$

We now express things more generally using the definition of $\oplus\mathsf{P}$. We have shown that for any NP language L, there is a polynomial-time computable relation R and a polynomial bound p such that

$$x \in L \Leftrightarrow \text{Я}r\, \underbrace{\exists j \in [16N^2] \oplus^N y.R(x,y,j,r)} \geq 1 - e^{-2N}$$

where $N = p(|x|)$ and $r = (r_1, \dots, r_{16N^2})$. Now

$$\begin{aligned} \exists j \in [16N^2] \oplus^N y.\, R(x,y,j,r) &= \neg\forall j \in [16N^2] \neg \oplus^N y.\, R(x,y,j,r) \\ &= \neg\forall j \in [16N^2] \oplus^{N+1} y.\, \overline{R}(x,y,j,r) \end{aligned}$$

where $\bar{(}R)$ is defined as in the following lemma.

Lemma 7.7. *If* $A \in \oplus\mathsf{P}$ *then* $\overline{A} \in \oplus\mathsf{P}$.

Proof. We simply add an additional 1 input to the parity quantifier to negate its value. More precisely, if $x \in A \Leftrightarrow \oplus^N y.\, R(x,y)$ for polynomial-time computable R then $x \in \overline{A} \Leftrightarrow \oplus^{N+1} y'.\, \overline{R}(x,y')$ where $\overline{R}(x,y') = ((y' = 0^{N+1}) \vee ((y' = 1y) \wedge R(x,y)))$. □

Since an $\exists$ quantifier acts as an OR and a $\forall$ acts as an AND we can view this as in Figure 7.1.

Since its inputs are 0-1-valued, the $\forall$ (or AND) acts simply as a fan-in $16N^2$ multiplication of large fan-in sums modulo 2. Expanding out this product of sums as a sum of products yields

$$\begin{aligned} \neg\forall j \in [16N^2] \oplus^{N+1} y.\, \overline{R}(x,y,j,r) &= \neg \oplus^{16N^2(N+1)} y_1, \dots, y_{16N^2} \bigwedge_{j=1}^{16N^2} \overline{R}(x,y_j,j,r) \\ &= \neg \oplus^{16N^2(N+1)} yR'(x,y,j,r) \qquad \text{for some polytime } R' \\ &= \oplus^{16N^2(N+1)+1} y\overline{R'}(x,y,j,r) \qquad \text{incorporating the negation.} \end{aligned}$$

This is only a single call to a $\oplus\mathsf{P}$ oracle. Plugging this in for the quantity in the underbrace yields the desired $\mathsf{R} \cdot \oplus\mathsf{P}$ algorithm. □

This argument has yielded almost all the ideas we will need for proving Lemma 7.2.

Proof of Lemma 7.2. Suppose $L \in \mathsf{PH}$, then there is some k such that $L \in \Sigma_k\mathsf{P}$. Therefore there is some relation R and polynomial p such that

$$\begin{aligned} L &= \{x \mid \exists^{p(|x|)} y_1 \forall^{p(|x|)} y_2 \exists \dots Q^{p(|x|)} y_k.\, R(x,y_1,\dots,y_k)\} \\ &= \{x \mid \exists^{p(|x|)} y_1 \neg\exists^{p(|x|)} y_2 \neg\exists\neg \dots Q^{p(|x|)} y_k.\, R(x,y_1,\dots,y_k)\}. \end{aligned}$$

Consider expanding the tree of quantifiers as a giant tree of possible values. For each of the $2^{jp(|x|)}$ values of the prefix $y_1, \ldots, y_j$ for $0 \leq j \leq k-1$ there is an $\exists$ node in the tree. The total number of such nodes over all values of $j \leq k-1$ is less than $2^{kp(|x|)}$. Let $N = kp(|x|)$. Choose $16N^2$ tuples $r_j = (i_j, v_1^j, \ldots, v_{i_j+1}^j)$ where $i_j \in [N]$ and $v_i^j \in \mathbb{F}_2^N$ as in the proof of Theorem 7.4. Apply the argument of Theorem 7.4 (before the conversion to a single $\oplus$P call) simultaneously to all the predicates corresponding to the all $\exists$ nodes, using the same sequence of random choices. (This involves adding the same set of linear constraints to augment each of the $\exists$ nodes in this tree.) For a fixed input x and a fixed node, the probability that the value at that node is incorrectly computeed is at most e^{-2N}. There are fewer than 2^N nodes in the tree and only 2^n inputs x of length n. Therefore the probability that there is some node of the tree that is computed incorrectly is at most $2^n \cdot 2^N \cdot e^{-2N} < \frac{1}{4}$.

So we have an computation for $x \in L$ described as

$$\text{Я}\overrightarrow{r}\exists j_1 \in [16N^2] \oplus^N y_1 \cdots \exists j_k \in [16N^2] \oplus^N y_k.\ R(x, \overrightarrow{r}, j_1, \ldots, j_k, y_1, \ldots, y_k)$$

for some polynomial-time computable relation R that gives the correct value all but at most $1/4$ of the time (over the random choices r). This is a bounded-error algorithm for L. (Note that because of the negations in the tree, the error is 2-sided and no longer 1-sided as in the case of NP.)

Again we multiply out the small fan-in $\exists$ quantifiers to yield to rewrite this expression as:

$$\text{Я}\overrightarrow{r} \oplus^{(16N^2)^k N} (y_1^1 \ldots y_k^1) \ldots (y_1^{16N^2}, \ldots, y_k^{16N^2}) \bigwedge_{j_1 \ldots j_k \in [16N^2]^k} \overline{R}(x, \overrightarrow{r}, j_1, \ldots, j_k, y_1^{j_1}, \ldots, y_k^{j_k}).$$

Since k is constant, this vector of y values has polynomial size and the interior computation is polynomial time, and thus $L \in \mathsf{BP} \cdot \oplus\mathsf{P}$. □

Proof of Lemma 7.3. We show that $\mathsf{BP} \cdot \oplus\mathsf{P} \subseteq \mathsf{P}^{\#\mathsf{P}}$. Suppose that $L \in \mathsf{BP} \cdot \oplus\mathsf{P}$. (It will be easy to see that the result also holds for the unbounded-error complexity class $\mathsf{P} \cdot \oplus\mathsf{P}$.) Then there is some polynomial-time TM M and polynomial function $N = p(|x|)$, such that

$$\begin{aligned} L &= \{x \mid \text{Я}^N r \oplus^N y.\ M(x, r, y) > 1/2\} \\ &= \{x \mid \sum_{r \in \{0,1\}^N} B(x, r) > 2^{N-1}\}, \end{aligned}$$

where $B(x, r) = \sum_{y \in \{0,1\}^N} M(x, r, y) \bmod 2$

Here is the basic proof idea: Suppose we could create a polynomial time TM M' such that

$$\sum_{y' \in \{0,1\}^{N'}} M'(x, y', r) \equiv B(x, r) \pmod{2^{N+1}}.$$

Then we would have $\sum_{r \in \{0,1\}^N} B(x, r) \equiv \sum_{r \in \{0,1\}^N} \sum_{y' \in \{0,1\}^{N'}} M'(x, y' r) \pmod{2^{N+1}}$.

Then to tell whether or not $x \in L$, we can simply compute $\#^{N+N'}(r, y').\ M'(x, y', r)$ using the #P oracle, take that value $\bmod 2^{N+1}$, and accept if the result is $> 2^{N-1}$.

We will not quite be able to do this but we will be able to find an M' such that

$$\sum_{y' \in \{0,1\}^{N'}} M'(x, y', r) \equiv -B(x, r) \pmod{2^{N+1}}.$$

By a similar argument we can decide L by making a call to compute $\#^{N+N'}(r, y')$. $M'(x, y', r)$ using the $\#$P oracle, taking that value $\bmod 2^{N+1}$, and accepting if the result is $< 2^{N+1} - 2^{N-1}$.

In order to satisfy these conditions we need to convert the number of accepting computations from being either 0 or -1 modulo 2 into a number that is either 0 or -1 modulo 2^{N+1}. The key technical lemma is the following:

Lemma 7.8. *For integers a and $m > 0$,*

(a) if $a \equiv 0 \pmod m$ then $4a^3 + 3a^4 \equiv 0 \pmod{m^2}$, and

(b) if $a \equiv -1 \pmod m$ then $4a^3 + 3a^4 \equiv -1 \pmod{m^2}$.

Proof. Part (a) follows because $m^2|a^2$ and $a^2|4a^3 + 3a^4$.

For part (b) write $a = km - 1$. Then

$$\begin{aligned} 4a^3 + 3a^4 &= 4(km-1)^3 + 3(km-1)^4 \\ &\equiv 12km - 4 + (-12km + 3) \pmod{m^2} \\ &\equiv -1 \pmod{m^2}. \end{aligned}$$

□

We apply this lemma inductively to construct polynomial-time machines $M_i(x, y, r)$ such that $\sum_y M_i(x, y, r) \equiv -B(x, r) \pmod{2^{2^i}}$. Applying the construction until $i = \lceil \log_2(N + 1) \rceil$ will yield the desired M'. For the base case $i = 0$, choosing $M_0(x, y, r) = M(x, y, r)$ we have $\sum_y M_0(x, y, r) \equiv -B(x, r) \pmod 2$ as required. For the inductive step, suppose that we have already constructed M_i. We will apply Lemma 7.8 with $m = 2^{2^i}$ and note that $m^2 = 2^{2^{i+1}}$. We create a new machine M_{i+1} so that on input (x, r) if a is the number of accepting choices for y in M_i then M_{i+1} will have $4a^3 + 3a^4$ accepting choices of its corresponding y'.

Let $y' = (z_1, z_2, z_3, y_1, y_2, y_3, y_4)$, such that $z_i \in \{0, 1\}$, and $y_i \in \{0, 1\}^{|y|}$. Define

$$\begin{aligned} M_{i+1}(x, r, y') = \; & (z_1 \wedge M_i(x, r, y_1) \wedge M_i(x, r, y_2) \wedge M_i(x, r, y_3) \wedge (y_4 = 0^{|y|})) \\ & \vee((\overline{z_1} \wedge (z_2 \vee z_3)) \wedge M_i(x, r, y_1) \wedge M_i(x, r, y_2) \wedge M_i(x, r, y_3) \wedge M_i(x, r, y_4)). \end{aligned}$$

It is easy to see the M_{i+1} has the desired number of accepting choices of y' as a function of the number of choices of y for M_i. By Lemma 7.8, we know that $\sum_{y'} M_{i+1}(x, y', r) \equiv -B(x, r) \pmod{2^{2^{i+1}}}$.

It remains to confirm the final machine M' is polynomial-time computable. It clearly will be polynomial in the length of the final string y'. Since the number of times to repeat this construction is $i = \lceil \log_2(N+1) \rceil$, and at each iteration $|y'| = 3 + 4|y|$, $|y'|$ grows by factor ≤ 7 per iteration. Therefore the total length of y' at the end is $\leq 7^{\log_2(N+1)} N$, which is polynomial in N and therefore polynomial in $|x|$ as required.

We now simply fit this into the framework to yield a polynomial-time algorithm for L that makes a single call to a $\#$P oracle. □