**6.841/18.405J Advanced Complexity Theory** **March 17, 2003**

# Lecture 11: Toda's Theorem Continued

*Instructor: Prof. Madhu Sudan* *Scribe: Ed Solovey*

**Last Time**

- Defined #P
- Discussed Operators on Complexity Classes
- Began Toda's Theorem

**Today**

1. Toda's Theorem (continued)
2. Introduce Interactive Proofs

## 1 Toda's Theorem

**Toda's Theorem.** $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}}$

We will prove Toda's theorem in two parts:

- **Part I** $\mathrm{PH} \subseteq \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$
- **Part II** $\mathrm{BP} \cdot \oplus \cdot \mathrm{P} \subseteq \mathrm{P}^{\#\mathrm{P}}$

We will eventually see that we only need to make one call to the #P oracle.

### 1.1 Part I - $\mathrm{PH} \subseteq \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$

We will prove this in several steps.

$$\begin{aligned} \exists \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{P} &\subseteq \mathrm{BP} \cdot \oplus \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{P} && \text{[Step 1]} \\ &\subseteq \mathrm{BP} \cdot \mathrm{BP} \cdot \oplus \cdot \oplus \cdot \mathrm{P} && \text{[Step 2]} \\ &\subseteq \mathrm{BP} \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{P} && \text{[Step 3]} \\ &\subseteq \mathrm{BP} \cdot \oplus \cdot \mathrm{P} && \text{[Step 4]} \end{aligned}$$

Since $\exists \cdot \mathcal{C} = \neg \cdot \forall \cdot \mathcal{C}$, we can see that any level of the hierarchy can be expressed as $\mathrm{BP} \cdot \oplus \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{BP} \cdot \oplus \cdots \cdot \mathrm{P}$, and by the steps above this can be expressed as, $\mathrm{BP} \cdot \oplus \cdot \mathrm{P}$. Implying that $\mathrm{PH} \subseteq \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$. Note that it is a necessary and satisfied condition that the class P is closed under all of the above operators.

In the steps below we will think of operations on complexity classes as circuits, with each operator corresponding to gates of a particular type. Although the gates may have exponential fan-in, the circuit is well structured and easily described.

### 1.1.1 Step 1 $\exists \cdot \mathbf{BP} \cdot \oplus \cdot \mathbf{P} \subseteq \mathbf{BP} \cdot \oplus \cdot \mathbf{BP} \cdot \oplus \cdot \mathbf{P}$

We would like to replace an OR gate by an approximate majority gate followed by parity gates. First, we note that

$\exists \cdot \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$, can be written as

$\exists_x, \mathrm{BP}_y, \oplus_z$,M(w,x,y,z) where w is the input to M, which is a poly time machine and x,y,z are advice strings polynomial in length with respect to w. Next, we observe that the above can be rewritten as

$\exists_x$,N(w,x) where $\mathrm{N} \in \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$. Finally, if we could show that this is equivalent to

$\mathrm{BP}_{\bar{h}} \oplus_{\bar{x},\bar{b},c} N_1(w, \bar{h}, \bar{x}, \bar{b}, c)$ for some $N_1 \in \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$, we would have the desired result. Let $\bar{h}$ be a sequence of m hash functions, $\bar{x}$ is m non-deterministic choices for N, $\bar{b}$ is m bits, and c is a bit.

First, consider $N_2(w, h_i, x_i, b_i)$ that accepts in the following cases:

- if input is all zeros, or
- $b_i$=1 and N(w,$x_i$) accepts $h_i(x_i)$=1.

Observe that if for all i, N(w,$x_i$) rejects, $N_2$ has an odd number, one, of accepting strings. If for some i, N(w,$x_i$) accepts, with some non-zero polynomial, based on the hash function h, $N_2$ will have an even number of accepting strings. Thus $N_2$ is a parity complementer for N. $N_2$ employs the concepts from the Valiant-Vazirani Theorem.

Now, consider $N_1(w, \bar{h}, \bar{x}, \bar{b}, c)$ that accepts in the following cases:

- if input is all zeros, or
- if c=1 and for all i $N_2(w, h_i, x_i, b_i)$

Observe that if $N_2$ ever rejects then $N_1$ will have an odd number, one, of accepting strings. If $N_2$ always accepts, then $N_1$ will have an even number, two, of accepting strings. Thus if for all i N(w,$x_i$) rejects, $N_1$ will have an even number of accepting strings, and if for some i N(w,$x_i$) accepts, then with non-zero probability, $N_1$ will have an odd number of accepting strings. $N_1$ is the parity complementer of the product function of $N_2$

By observing the parity of $N_1$'s computation and using a BP gate to account for the probabilistic behavior, we can see that $N_1 \in \mathrm{BP} \cdot \oplus \cdot \mathrm{P}$.

### 1.1.2 Step 2 $\mathbf{BP} \cdot \oplus \cdot \mathbf{BP} \cdot \oplus \cdot \mathbf{P} \subseteq \mathbf{BP} \cdot \mathbf{BP} \cdot \oplus \cdot \oplus \cdot \mathbf{P}$

Here we want to reverse the order of a parity gate and a BP gate.

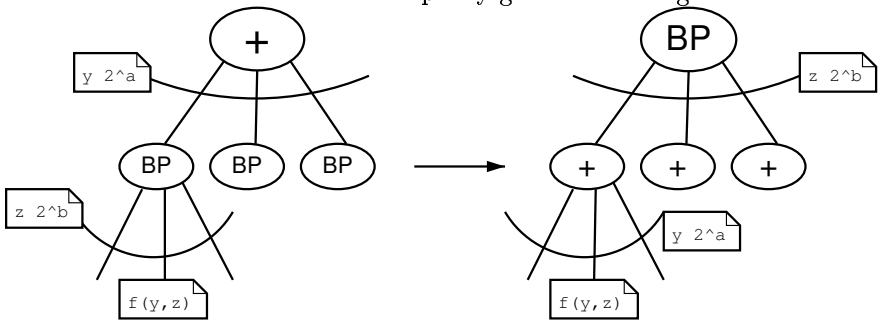


*Figure 1 : Reversing the order of BP and parity gates*

In the initial case the possibility of error is not introduced until the last level of gates, thus the probability that any f(y,z) value is erroneous is $2^{-c}$. Once we have made the switch in gate order, each

parity gate is now receiving many inputs each one of which might have been computed erroneously. Thus taking the union bound over the maximum number of possible inputs to a particular parity gate, we get that each f(y,z) value is erroneous with probability $2^b \cdot 2^{-c} = 2^{b-c}$. As long as c is much larger than b, this probability is still significantly small. Since, we can amplify BP gates, we can ensure this property.

### 1.1.3 Step 3 $\mathbf{BP \cdot BP \cdot \oplus \cdot \oplus \cdot P \subseteq BP \cdot BP \cdot \oplus \cdot P}$

Here we want to eliminate one of two consecutive parity gates.

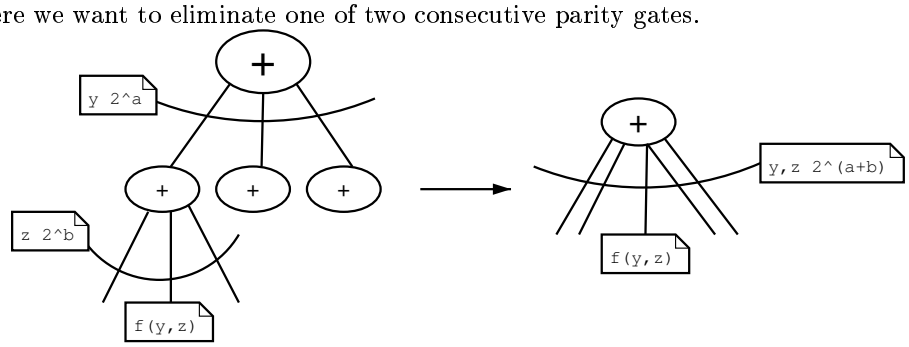


*Figure 2 : Eliminating one of two consecutive parity gates.*

Since the sum modulo 2 of other sums modulo 2, can be accomplished on a single "sum modulo 2" machine, we can collapse the parity gates, with added fan-out being the only cost.

### 1.1.4 Step 4 $\mathbf{BP \cdot BP \cdot \oplus \cdot P \subseteq BP \cdot \oplus \cdot P}$

Here we want to eliminate one of two consecutive BP gates.

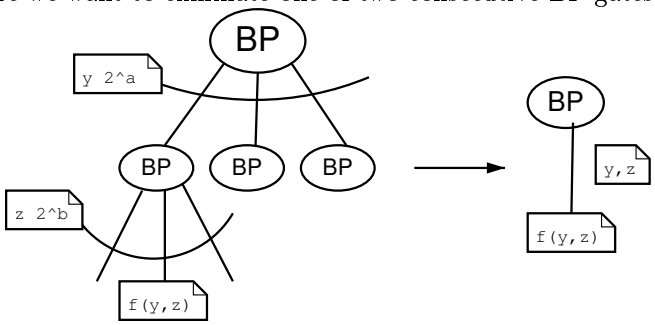


*Figure 3 : Eliminating one of two consecutive BP gates.*

If in the first case the error of the first level BP gate is $\epsilon$ and that of the second level BP gates is $\lambda$, then by the union bound, the error of the single BP gate is $\epsilon + \lambda$. Notice that we did not use the Razborov - Smolensky theorem because we wanted to preserve uniformity.

We observe that most of the power of $\mathrm{BP} \cdot \oplus \cdot \mathrm{P}$ seems to come from the parity operator. The ability to know whether the number of accepting strings is even or odd seems extremely powerful because this calculation is so sensitive to change. A single string in or out of the set changes the result.

## 1.2 Part II - $\mathbf{BP \cdot \oplus \cdot P \subseteq P^{\#P}}$

Consider a language $L = \mathrm{BP}_y \cdot \oplus_z \cdot \mathrm{L'}$, where $\mathrm{L'} \in \mathrm{P}$. Using amplification on BP we can view L as:

1. if $x \in L$, then $Pr_y[|z : L'(x,y,z) = 1| \text{ is odd}] \geq 2/3$; and
2. if $x \notin L$, then $Pr_y[|z : L'(x,y,z) = 1| \text{ is odd}] < 1/3$

And in the circuit conceptualization as:

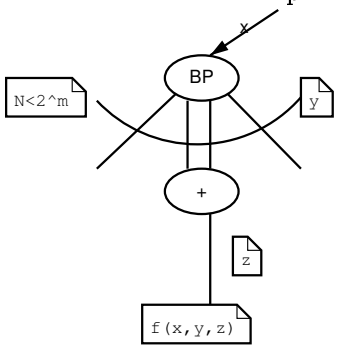


*Figure 4 : Circuit for language L* Where N is the size of the fan out of the BP gate.

### 1.2.1 Boosting parity gate

Thus by posing the question, "for a certain x how many f(y,z)'s exist such that f(x,y,z)=1?", we want to able to determine whether $x \in L$. We would thus be asking the oracle for something of the form $\sum_y \sum_z L'(x,y,z)$. The problem with our current modulo-2 parity operator is that because the sum of two odds is even we can't distinguish between cases such as,

1. for $y_1$ there are 3 accepting z's, for $y_2$ there are 5 accepting z's, for $y_3$ there are 9 accepting z's, and for $y_4$ there are 2 accepting z's.
2. for $y_1$ there are 3 accepting z's, for $y_2$ there are 2 accepting z's, for $y_3$ there are 4 accepting z's, and for $y_4$ there are 6 accepting z's.

In both situation the result of the summation is 1 (mod 2). However, in case 1 $Pr_y[|z : L'(x,y,z) = 1| \text{ is odd}] = 3/4$ and in case 2, $Pr_y[|z : L'(x,y,z) = 1| \text{ is odd}] = 1/4$.

It thus appears that if we could increase our field by having a parity operator that worked modulo-v, for some large v we could avoid this overlap of information and could correctly distinguish between the two cases based on the result of the sum. Say we had a parity gate that operated modulo-$2^m$, where $2^m > N$, (where N is the fan out of the BP gate, maximum number of y's). We could then see that if

- $x \in L$ then the sum (mod $2^m$) $\in [2/3N, N]$, and
- $x \notin L$ then the sum (mod $2^m$) $\in [0, 1/3N]$.

If we start with a parity gate operating in modulo-K and want to boost it to a gate operating modulo-$K^2$, we see that the relationship is not preserved when operating in {0,1}, but is preserved when operating in {0,-1}. This slight modification causes us to update the previous expression to:

- $x \in L$ then the sum (mod $2^m$) $\in [-2/3N, -N]$, and
- $x \notin L$ then the sum (mod $2^m$) $\in [0, -1/3N]$.

### 1.2.2 Arithmetic Games

The idea of boosting parity gates motivates the use of arithmetic on the number of accepting paths in non-deterministic Turing machines. Let $n_1(x)$ and $n_2(x)$ be the number of accepting paths of $N_1$ and $N_2$ on some input x, where $N_1$ and $N_2$ are NTM's.

**Definition 1.1** $N_+(x)$ is an NTM such that $n_+(x) = n_1(x) + n_2(x)$

Thus the number of accepting strings of $N_+$ is equal to the sum of the number of accepting strings of $n_1(x)$ and $n_2(x)$. To see that this is plausible, construct $N_+(x)$ in this manner:

On input w, $N_+$="

1. non-deterministically make a choice between $N_1$ and $N_2$
2. if $N_1$ chosen, simulate $N_1$ on w. Accept if it accepts
3. if $N_2$ chosen, simulate $N_2$ on w. Accept if it accepts

The running time of $N_+$ is the maximum of the running times of $N_1$ and $N_2$. Consider this with circuits, $C_1$ and $C_2$, that accept $n_1$ and $n_2$ n-bit inputs respectively, we can construct $C_+$ to be $C_+(x, b) = (b \wedge C_1(x)) \vee (\bar{b} \wedge C_2(x))$, where b is a single bit.

**Definition 1.2** $N_*(x)$ is an NTM such that $n_*(x) = n_1(x) * n_2(x)$

Thus the number of accepting strings of $N_*$ is equal to the product of the number of accepting strings of $n_1(x)$ and $n_2(x)$. To see that this is plausible, construct $N_*(x)$ in this manner:

On input w, $N_*$="

1. simulate $N_1$ on w
2. if $N_1$ accepts, simulate $N_2$ on w. Accept if $N_2$ accepts.
3. if $N_1$ rejects, reject.

The running time of $N_*$ is the sum of the running times of $N_1$ and $N_2$. Consider this with circuits, $C_1$ and $C_2$, that accept $n_1$ and $n_2$ n-bit inputs respectively, we can construct $C_*$ to be $C_*(x) = C_1(x) \wedge C_2(x)$.

Given a polynomial time NTM that has n(x) accepting paths, using $N_+$ and $N_*$ we can construct a polynomial time NTM $N_p(x)$ that has $P_{|x|}(n(x))$ accepting paths, where $P_{|x|}(n(x))$ is a polynomial of degree poly(n) and has positive coefficients with values at most $2^{\mathrm{poly(n)}}$.

Now we return to the idea of boosting our parity gate. We want to find a polynomial p that preserves the relationship:

- $p(x) = 0 \ (mod K^2)$ if $x = 0 \ (mod K)$, and
- $p(x) = -1 \ (mod K^2)$ if $x = -1 \ (mod K)$.

Consider the polynomial $p(x) = 3x^4 + 4x^3$. It can be verified that this polynomial preserves the necessary relationship. Try adding $a \cdot x^2(x+1)^2$ to p(x), for some integer a, and note that the values (mod K) and (mod $K^2$) are not effected. Once we have this polynomial to test membership in $L = \mathrm{BP}_y \cdot \oplus_z \cdot \mathrm{L'}$ (defined above), we look at the sum $\sum_y p(\sum_z L'(x, y, z)) \ mod(2^m)$. This sum can be returned via a single query to a #P oracle. Thus $L \in \mathrm{P}^{\#\mathrm{P}}$ and $\mathrm{BP} \cdot \oplus \cdot \mathrm{P} \subseteq \mathrm{P}^{\#\mathrm{P}}$.

# 2 Interactive Proofs

## 2.1 Background Work

The notion of interactive proofs was defined by Goldwasser, Micali, and Rackoff (ca. 1982-1984). The motivation for their work was cryptography. A few years later, Babai and Moran(1988) addressed this notion from a number theoretic perspective.

## 2.2 Cryptography

A common problem in cryptography is secure user identification. For example, a user P wants to identify himself to a computer C, by proving to C that he knows the password X. C might prompt P with some questions and an interaction will ensue, after which C will or will not be convinced that the party she is talking to knows the password X. Because this interaction may take place over insecure communication channels, user P does not want to send the entire password X over the channel. The original belief was that, "to prove that I know X, I must show you X." Goldwasser addressed the formal definitions of "knowledge" and "proof".

## 2.3 Introduction to Interactive Proofs

*The power of a theorem proof system is modeled by the language for which it can prove membership.*

An interactive proof consists of a verifier, V, and a prover, P. V knows some value x and P attempts to prove to V that it also knows the value x by sending some kind of proof, $\pi$ to V. Thus,

- $V(x, \pi)$ = Accept $\rightarrow x \in L$
- $V(x, \pi)$ = Reject $\rightarrow \pi$is not a valid proof that$x \in L$

**Definition 2.1** A verifier V is said to be complete for a language L if $\forall x \in L \ \exists \ \pi$s.t. $V(x, \pi)$ accepts.

**Definition 2.2** A verifier V is said to be sound for a language L if $\forall x \notin L \ \forall \ \pi \ V(x, \pi)$ rejects.

In the interaction, V is assumed to be a polynomial time machine, while P is an infinitely powerful one. The interaction starts by P sending something to V to let V know that P wants to prove something about x. V responds with $q_1 = q(x)$, a question about x. P responds with $a_1 = a(x, q_1)$, an answer based on what he is trying to prove and the verifier's question. V then responds with $q_2 = q(x, a_1)$. The interaction continues in this manner until P sends the final answer to V. At this point V either rejects or accepts. All of the functions q, as well as the final decision, computed by V, must be computed in polynomial time.

**If P is infinitely powerful and V is a polynomial machine, why is interaction useful? Can't P just compute the entire interaction ahead of time?** In the model outlined above this indeed appears to be the case. The power of interaction is apparent when V is allowed to have randomness. In the interaction above, V is now also allowed to flip coins and send random strings to P as part of its questions. The definitions of completeness and soundness are modified for a random polynomial time verifier.

**Definition 2.3** A verifier V with randomness is said to be complete for a language L if $\forall x \in L \ \exists$ a prover that convinces V that x $\in L$ with probability $\geq 2/3$

**Definition 2.4** A verifier V is said to be sound for a language L if $\forall x \notin L$ and $\forall$ provers, V accepts P's proof with probability $< 1/3$

Because of its power of randomness, V might not necessarily know whether x $\in L$. However, by repeatedly asking P certain questions and observing that the same random strings always produce the same answer V will be convinced that P indeed knows that x $\in L$. For example, the invisible chalk on paper game.

We will continue the discussion of Interactive Proofs in the next lecture.