

# CMPT 710 - Complexity Theory: Week 10

Valentine Kabanets

November 3, 2006

## 1 Example of a BPP language

### 1.1 Polynomial Identity Testing

Suppose we are given two arithmetic formulas  $f(x_1, \dots, x_n)$  and  $g(x_1, \dots, x_n)$  with integer coefficients. We want to know whether these formulas are equivalent, i.e., whether  $f \equiv g$  as polynomials. One way to check this is to write out all the coefficients of the polynomials  $f$  and  $g$ , and compare them one by one. However, there may be exponentially many such coefficients, and so this approach will result in a highly inefficient algorithm.

A better way to solve this problem is by using a randomized algorithm. We simply pick random values to the variables  $x_1, \dots, x_n$ , and check if the two polynomials agree on these values. If the two polynomials are equivalent, then they will evaluate to the same value on any given point. If they are different polynomials, then it's very unlikely that they will agree on a random point. To formalize this, we need the following lemma. Recall that the degree of a monomial  $x_1^{d_1} \dots x_n^{d_n}$  is  $d_1 + \dots + d_n$ ; the total degree of a polynomial  $f$  is the maximum degree of a monomial of  $f$ .

**Lemma 1 (Schwartz-Zippel, and many others).** *Let  $f(x_1, \dots, x_n)$  be a non-zero polynomial over a field  $F$ , such that the total degree of  $f$  is  $d$ . Let  $S \subseteq F$  be a finite subset of field elements. Then*

$$\Pr_{r_1, \dots, r_n \in S}[f(r_1, \dots, r_n) = 0] \leq \frac{d}{|S|}.$$

*Proof.* By induction on the number of variables  $n$ . Base case of  $n = 1$  is easy: a univariate degree  $d$  non-zero polynomial over a field can have at most  $d$  roots. Hence, a random point  $r$  is a root with probability at most  $d/|S|$ .

Assume the statement is true for  $k$  variables, and let's prove it for  $k+1$  variables. Express a polynomial  $f(x_1, \dots, x_{k+1})$  as a polynomial  $f_1(x_{k+1})$ , whose coefficients are polynomials in  $x_1, \dots, x_k$  (by factoring out the expressions  $x_{k+1}^i$  for  $1 \leq i \leq d$ ). Let  $d_1$  be the degree in  $x_{k+1}$  of  $f_1$ . Let  $p(x_1, \dots, x_k)$  be the coefficient at  $x_{k+1}^{d_1}$  in  $f_1$ ; that is,  $f_1(x_{k+1}) = x_{k+1}^{d_1} p(x_1, \dots, x_k) + \dots$ . Note that the total degree of  $p$  is at most  $d - d_1$ , since  $d$  is the total degree of  $f$ . We have  $\Pr_{r_1, \dots, r_{k+1}}[f(r_1, \dots, r_{k+1}) = 0] =$

$$\Pr[f(r_1, \dots, r_{k+1}) = 0 \mid p(r_1, \dots, r_k) \neq 0] * \Pr[p(r_1, \dots, r_k) \neq 0] \tag{1}$$

$$+ \Pr[f(r_1, \dots, r_{k+1}) = 0 \mid p(r_1, \dots, r_k) = 0] * \Pr[p(r_1, \dots, r_k) = 0]. \tag{2}$$

We can upperbound this sum as follows.

$$\begin{aligned} & \Pr[f(r_1, \dots, r_{k+1}) = 0 \mid p(r_1, \dots, r_k) \neq 0] * \Pr[p(r_1, \dots, r_k) \neq 0] \leq \\ & \Pr[f(r_1, \dots, r_{k+1}) = 0 \mid p(r_1, \dots, r_k) \neq 0] \leq \\ & \Pr_{r_{k+1}}[f_1(r_{k+1}) = 0], \end{aligned}$$

where  $f_1(r_{k+1})$  is obtained after substituting the values  $r_1, \dots, r_k$  for  $x_1, \dots, x_k$ . Note that after such substitutions, we obtain a univariate polynomial of degree at most  $d_1$ . Hence,

$$\Pr_{r_{k+1}}[f_1(r_{k+1}) = 0] \leq d_1/|S|.$$

For the second summand, we have

$$\begin{aligned} & \Pr[f(r_1, \dots, r_{k+1}) = 0 \mid p(r_1, \dots, r_k) = 0] * \Pr[p(r_1, \dots, r_k) = 0] \leq \\ & \Pr[p(r_1, \dots, r_k) = 0] \leq \\ & (d - d_1)/|S|, \end{aligned}$$

where the last inequality is by the Inductive Hypothesis.

Putting everything together, we get

$$\Pr[f(r_1, \dots, r_{k+1}) = 0] \leq d_1/|S| + (d - d_1)/|S| = d/|S|,$$

as promised. □

## 1.2 Applications of the Schwartz-Zippel lemma

First, let's go back to our problem of testing whether two given arithmetic formulas  $f(x_1, \dots, x_n)$  and  $g(x_1, \dots, x_n)$  are equivalent. Let  $d_1$  and  $d_2$  be the total degrees of  $f$  and  $g$ , respectively. Consider the new polynomial  $h = f - g$ , of total degree at most  $d = \max\{d_1, d_2\}$ . Let  $S = \{1, 2, 3, \dots, N\}$ , for  $N = 2d$ . Then, testing whether  $f$  and  $g$  agree on a random  $n$ -tuple of elements from  $S$  is the same as testing whether  $h$  is zero on that tuple. If  $f \neq g$ , then  $h \neq 0$ , and by the Schwartz-Zippel lemma, the probability of picking a root for  $h$  is at most  $d/|S| = 1/2$ .

On the other hand, if  $f \equiv g$ , then  $h \equiv 0$ , and so  $h$  is zero on every point. Thus, we have a randomized polytime algorithm for testing if two arithmetic formulas are equivalent over integers. This algorithm has one-sided error, bounded from above by  $1/2$ .

In the above argument, we didn't say what the degrees  $d_1$  and  $d_2$  were. Let us define an arithmetic formula as a tree whose leaves are labeled by variables  $(x_1, \dots, x_n)$  or constants, and whose inner nodes are labeled by  $+$  or  $*$ . It is easy to prove by induction on the size of the formula, that a formula of size  $s$  can have degree at most  $s$ . Here the size of the formula is defined to be the number of leaves in the formula.

Here is the proof. For  $s = 1$ , the degree of the formula is either 0 or 1; so we're done. Let  $f = p_1 + p_2$ , where the size of  $f$  is  $s$ , and the sizes of  $p_1$  and  $p_2$  are  $s_1$  and  $s_2$ , respectively. Note that  $s = s_1 + s_2$ . Then by the Inductive Hypothesis, the degrees of  $p_1$  and  $p_2$  are at most  $s_1$  and  $s_2$ , respectively. Hence, the degree of  $f$  is at most  $\max\{s_1, s_2\} \leq s$ . The case of  $f = p_1 * p_2$  is similar - the degree of  $f$  is at most  $s_1 + s_2 = s$ .

So, when given an arithmetic formula in  $x_1, \dots, x_n$  of size  $m$ , we can always upperbound the degree of the polynomial computed by this formula by  $m$ . Then we can choose a set  $S = \{1, 2, \dots, 2m\}$ , from which we'll randomly pick values for the variables. The Schwartz-Zippel lemma guarantees that our error probability will be less than  $m/(2m) = 1/2$ .

One important point. Once we picked random values for the variables of our polynomial  $f(x_1, \dots, x_n)$  and started evaluating our polynomial, we may obtain big intermediate numbers. For example, if the degree of  $f$  is  $m$ , and if a largest element in  $S$  is  $2m$ , then the final value of  $f$  may be as big as  $(2m)^m$ . However, this value can be written down using  $O(m \log m)$  bits only, which is polynomial in the input size  $m$ .

What happens if we are given an arithmetic *circuit* rather than a formula? The difference now is that instead of a tree, we have a graph (DAG). Now, the degree of the polynomial given by an arithmetic circuit of size  $m$  can be much bigger than  $m$  - think of doing "repeated squaring" by having a sequence of multiplication gates  $g_1, \dots, g_m$  such that the inputs to gate  $g_{i+1}$  are two copies of the output from gate  $g_i$ ; then each such gate squares the degree of the polynomial, so that after  $m$  gates, the degree can be  $2^m$ , exponential in the size of the arithmetic circuit. However, this is the worst possible. Again, a simple argument by induction shows that the degree of a polynomial computed by an arithmetic circuit of size  $m$  is at most  $2^m$ .

This degree is big, and if we start evaluating a polynomial of degree  $2^m$  (even on small numbers), the final result may be as big as  $2^{2^m}$ , whose bit-size is at least  $2^m$ , exponential in the circuit size. We cannot hope to even write down such huge number in time polynomial in  $m$ . So what do we do?

The trick is to use modular arithmetic! We just pick a random prime  $p$  in the interval  $[1..2^{2^m}]$ , and perform all our calculations modulo that prime  $p$ . What is the probability that we are unlucky to pick a prime  $p$  that divides the value of circuit on some random inputs? First of all, let's pick the set  $S = \{1, 2, \dots, 2^{2^m}\}$ . The biggest possible value of our polynomial, given the inputs from  $S$ , will be  $v = (2^{2^m})^{2^m}$ . This number has at most  $2m2^m$  possible prime divisors. The interval  $[1..2^{2^m}]$  has at least  $2^{2^m}/(2m \ln 2) \geq 2^{1.5m}$  primes (by the Prime Number Theorem). Therefore, the probability of picking a prime divisor of the number  $v$  is at most  $2m2^m/(2^{1.5m}) \leq 2^{-0.4m} \ll 1$ , is very small. Thus, our randomized algorithm, modified to do modular arithmetic, is still very likely to be correct. But now this algorithm is guaranteed to run in polytime because all intermediate values  $\pmod p$  can be written down using only  $2m$  bits.

## 2 Randomized NP

Randomness is a computational resource that can be combined with other resources, e.g., nondeterminism. We will consider such a combination next.

Recall that a language  $L \in \text{NP}$  if there is polytime verifier such that, for every  $x \in L$ , there is a proof (or witness) of small size that makes the verifier accept. (The verifier is the polytime relation  $R(x, y)$ , where  $x$  is the input, and  $y$  is supposed to be a "proof" that  $x \in L$ .) In this definition, the verifier is a *deterministic* polytime algorithm. By letting the verifier be *randomized* polytime algorithm, we obtain an extension of NP, called MA (which stands for Merlin-Arthur). Here, Merlin is an all-powerful wizard that tries to convince a

probabilistic polytime verifier Arthur that an input string is in the language. If a string is indeed in the language, Merlin can make Arthur accept most of the time. If, on the other hand, the string is not in the language, then no matter what kind of proof Merlin shows to Arthur, Arthur will reject most of the time.

More formally, we say that a language  $L \in \mathbf{MA}$  if there is a constant  $c$ , and a polytime relation  $R(x, y, z)$  such that, for every  $x$  of length  $n$ , we have

$$\begin{aligned} x \in L &\Rightarrow \exists y \Pr_z[R(x, y, z) = 1] \geq 3/4 \\ x \notin L &\Rightarrow \forall y \Pr_z[R(x, y, z) = 1] \leq 1/4, \end{aligned}$$

where  $|y| = |z| \leq n^c$ . In other words, if  $x \in L$ , then there is a short proof  $y$  that will convince Arthur with probability at least  $3/4$ , and if  $x \notin L$ , then every  $y$  will be rejected by Arthur with probability at least  $3/4$ .

### 3 Example of a language in MA

Recall that an *arithmetic formula* is a tree whose leaves are labeled by variables or constants, and whose inner nodes are labeled by arithmetic operations  $+$ ,  $-$ , and  $*$ . The *size* of an arithmetic formula is the number of leaves in the tree representation of the formula. (Note that variables labeling the leaves may repeat, i.e., the same variable may label more than one leaf.)

Each arithmetic formula  $f(x_1, \dots, x_n)$  computes some polynomial  $p(x_1, \dots, x_n)$  over the integers. The same polynomial may or may not be computable by a smaller formula. We say that an arithmetic formula  $f$  is *optimal* if it is a smallest possible formula that computes the polynomial  $p$ , i.e., any smaller arithmetic formula computes a different polynomial than  $p$ . We say that a formula is *non-optimal* if there is a smaller formula computing the same polynomial.

Define  $L = \{f(x_1, \dots, x_n) \mid f \text{ is a non-optimal arithmetic formula}\}$ . We will show that  $L \in \mathbf{MA}$ .

Let  $f(x_1, \dots, x_n)$  be an input arithmetic formula of size  $s$ . From the last lecture, we know that the degree of the polynomial computed by  $f$  is at most  $s$ . Let  $S = \{1, 2, \dots, 4s\}$ . Merlin will try to convince Arthur that  $f$  is non-optimal by providing Arthur with a strictly smaller formula  $g(x_1, \dots, x_n)$  that is supposed to compute the same polynomial as  $f$ . Upon receiving a formula  $g(x_1, \dots, x_n)$ , Arthur will pick a sequence of random integers  $r_1, \dots, r_n$  from the set  $S$  defined above, and will check whether  $f(r_1, \dots, r_n) = g(r_1, \dots, r_n)$ . If the equality holds, then Arthur accepts; otherwise, Arthur rejects.

Let us analyze the correctness of the described  $\mathbf{MA}$  protocol. Suppose  $f$  is indeed non-optimal. Then Merlin can send to Arthur a smaller equivalent formula  $g$ , and Arthur will accept with probability  $1 \geq 3/4$  (since  $f \equiv g$  means that  $f$  and  $g$  agree on all possible inputs). Now suppose that  $f$  is optimal. Then whatever smaller formula  $g$  is sent to Arthur, the polynomial  $f - g$  is non-zero, of degree at most  $s$ . Hence, by the Schwartz-Zippel lemma, Arthur will accept in this case with probability at most  $s/(4s) = 1/4$ .

**Remark** The reasoning above can be generalized to show that a language of non-optimal arithmetic *circuits* is also in  $\mathbf{MA}$  (by using the modular-arithmetic algorithm from the last lecture for testing if a given arithmetic circuit is zero).

**Remark** Earlier we saw in class that the language of non-optimal *Boolean* circuits is in  $\Sigma_2^P$ . The case of *arithmetic* circuits is easier — the language of non-optimal arithmetic circuits is in **MA**, which is a subset of  $\Sigma_2^P$ . (The inclusion  $\mathbf{MA} \subseteq \Sigma_2^P$  can be shown using the ideas from the proof that  $\mathbf{BPP} \subseteq \Sigma_2^P$  that we saw in class. *Exercise:* Prove that  $\mathbf{MA} \subseteq \Sigma_2^P$ . Hint: Use the error reduction result from the next section.)

## 4 Error Reduction in MA

As in the case of **BPP**, we can reduce the error probability of any **MA** protocol to be less than an inverse exponential in the input size. Here's how.

Let  $L \in \mathbf{MA}$  be any language. Let  $R(x, y, z)$  be a polytime relation for  $L$  such that, for every  $x \in L$ , there is a  $y$  with  $\Pr_z[R(x, y, z) = 1] \geq 3/4$ ; and for every  $x \notin L$ , for every  $y$ , it holds that  $\Pr_z[R(x, y, z) = 1] \leq 1/4$ .

Consider a new protocol where, upon receiving a string  $y$ , Arthur randomly and independently chooses  $k$  strings  $z_1, \dots, z_k$ , and accepts iff  $R(x, y, z_i) = 1$  for more than half of these  $k$  strings.

We use Chernoff bounds to analyze the correctness of the described protocol. Suppose first that  $x \in L$ . Then Merlin can send Arthur a string  $y$  such that  $\Pr_z[R(x, y, z) = 1] \geq 3/4$ . Every string  $z_i$ ,  $1 \leq i \leq k$ , randomly chosen by Arthur has probability at least  $3/4$  of satisfying  $R$ . The expected number of  $z_i$ 's that satisfy  $R$  is  $\mu \geq \frac{3}{4}k$ . Let  $X_i$ ,  $1 \leq i \leq k$ , be a random variable that is 1 if  $z_i$  satisfies  $R$ , and 0 otherwise. Let  $X = \sum_{i=1}^k X_i$ . As we just argued, the expectation of  $X$  is  $\mu$ . Using Chernoff bounds, we get that  $\Pr[X \leq k/2] \leq \Pr[|X - \mu| > k/4] < 2e^{-k/48}$ . Thus, Arthur will accept with probability exponentially close to 1.

On the other hand, suppose that  $x \notin L$ . Then, whatever  $y$  is sent to Arthur,  $\Pr_z[R(x, y, z) = 1] \leq 1/4$ . Let  $X_i$  be random variables as defined above, and let  $X = \sum_{i=1}^k X_i$ . Then the expectation of  $X$  is  $\leq \frac{1}{4}k$ . The probability that Arthur accepts in this case is  $\Pr[X > k/2]$ , which, by Chernoff bounds, is at most  $2e^{-k/48}$ . Thus, in this case, Arthur will accept with probability exponentially close to 0.

## 5 Class AM

Suppose we change the order in which Merlin and Arthur communicate by letting Arthur go first, and Merlin go second. More formally, we say that a language  $L \in \mathbf{AM}$  if there is a constant  $c$  and a polytime relation  $R(x, y, z)$  such that, for every  $x$  of length  $n$ ,

$$\begin{aligned} x \in L &\Rightarrow \Pr_z[\exists y : R(x, y, z) = 1] \geq 3/4, \\ x \notin L &\Rightarrow \Pr_z[\exists y : R(x, y, z) = 1] \leq 1/4, \end{aligned}$$

where  $|y| = |z| \leq n^c$ .

In other words, if  $x \in L$ , then Merlin can successfully answer with  $y$  almost every random challenge  $z$  from Arthur. If  $x \notin L$ , then, for most random challenges from Arthur, Merlin does not have a good answer.

How are the classes **MA** and **AM** related to each other? We show the following.

**Theorem 2.**  $MA \subseteq AM$

We'll give the proof next time.