Preface

These are the lecture notes for the course **Algebra and Computation** taught at MIT in Fall 1998. I'd like to thank the scribes Ted Allison, Amos Beimel, Yevgeniy Dodis, John Dunagan, Venkatesan Guruswami, Prahladh Harsha, Adam Klivans, Anna Lysyanskaya, Daniele Micciancio, Zulfikar Ramzan, Sofya Raskhodnikova, Leonid Reyzin, Amit Sahai, Rocco Servedio and Salil Vadhan for a wonderful job. The notes are much more careful and clear than I was. Thanks in particular to Salil Vadhan who lectured on November 9, 1998, two hours after Roshni Malhaar was born.

These notes are by no means polished. Nevertheless comments are appreciated - by email to `madhu@mit.edu`.

- Madhu Sudan

March 24, 1999

6.966 Algebra and Computation September 9, 1998

Lecture 1

Lecturer: Madhu Sudan *Scribe: John Dunagan*

1.1 Course Overview

In this course we will investigate the relationship between algebra and computation. Towards this end, we will survey computational results in algebra and some notions and results from algebraic models of computation. A historical goal of algebra has been to find explicit computational solutions to algebraic problems (i.e., Euclid's algorithm). Algebra has also had a very strong influence on our thinking about algorithms.

Some of the sub-areas we will focus on are: Algorithms from algebra; algebraic models of computation; and algebraic tools for dealing with complexity. We won't cover all the interesting algebraic problems and algorithms, but we'll hit a lot of the high points. Let us start by comparing algebraic problems and algorithms against the more general class of combinatorial problems and algorithms.

1.2 Two Flavors of Algorithms

Combinatorial problems typically assume an unstructured input. This typically leads to intractability relatively quickly. Algebra assumes a much greater amount of structure. To study this contrast more formally, let us look at a basic framework which captures many combinatorial and algebraic problems — namely "constraint satisfaction." In this framework, we typically describe a space of solutions, impose a collection of constraints, and then ask for a solution satisfying all the constraints. Combinatorial problems such as sorting, max flow, linear programming, and 3SAT can all be expressed in this way. E.g.:

Sorting: Given $X = \{x_1, x_2, ...x_n\}$ find $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ such that for every pair of indices $i, j \in \{1, \ldots, n\}$ with $i < j$, $\pi(i) \neq \pi(j)$ and $x_{\pi(i)} < x_{\pi(j)}$.

3SAT: Given $X = \{x_1, x_2, ...x_n\}$, and a series of functions of three variables, $x_i \vee x_j \vee x_k$, does there exist an assignment of the $\{x_i\}$ such that all the functions evaluate to true?

Similarly central algebraic problems are also "constraint satisfaction problems". Consider, for example, the classical problem of solving linear systems.

System of linear equations: Find x, y such that $ax + by = c$ and $dx + ey = f$.

Another common problem is that of finding integer solutions to the above linear system, which gives an example of a Diophantine problem. And, if the equalities in the linear system are replaced by inequalities, we obtain the problem of linear programming. All these provide examples of cosntraint satisfaction problems.

A constraint satisfaction problem of a somewhat different flavor is that of finding roots of polynomials.

Roots of quadratic polynomials: Find x such that $ax^2 + bx + c = 0$.

This problem of solving a polynomial equation can be generalized in many ways. We can look at systems of many polynomials, polynomials of higher degree, polynomials of many variables, and replacing the equalities by inequalities. A familiar system of polynomials in many variables is given below.

System of polynomial equations Given the variables $x_1, x_2, ...x_n$, the equations $x_i(1 - x_i) = 0$ for every i, and the additional equations, $x_1x_2(1 - x_3) = 0$, $x_3(1 - x_2)x_5 = 0$, ..., is there a solution?

Remark: Notice that this captures 3SAT!

These problems lead to two of the most fundamental problems in algebra: polynomial factorization and quantifier elimination. These two shall be the focus of much of this course. Since at times, the problems will look repetitive (and more so the solutions), there will occasionally be a hint of applications.

The bigger goal here is to learn about the algebraic theory of the words that end with **-ant**. These include the **determinant, permanent, discriminant, resultant, continuant, alternant,** etc. In general they are covered by the unifying word: **invariant**. Algebra is most fortunate in that not only can it solve systems when they can be solved, it can also provide simple algebraic expressions that tell you whether or not a solution exists. Classical examples of this are the **determinant** and the **discriminant**. Understanding this underlying algebra should enhance your understanding of why things work.

1.3 Complexity

A lot of recent work in complexity theory has relied on an algebraic setting for the problems of complexity theory. An example problem that is often used is:

System of degree 2 polynomials Given m polynomials of degree 2 in n variables, do they have a common zero.

Exercise: Prove above is NP-hard.

Since we often move well-known combinatorial problems to an algebraic setting, it seems worthwhile to study these problems directly in an algebraic setting. This leads to the issue of whether there is a direct way to study algebraic problems in an algebraic model of computation.

A number of algebraic models have been proposed in the past to study computation; notable examples are due to Valiant, and more recently, to Blum, Shub and Smale. These models lead to natural generalizations of the P vs. NP question. They have natural complete problems, and (under some restrictions) a separation of P from NP in these models yields some useful information about the P=NP question.

Unfortunately, we must conclude with the same observation so often made in other complexity courses; we know how to prove precious little. However, some weak separations are possible in some weak models of computation, and we will give a brief survey of some of the methods used to lower bound the algebraic complexity of computations in some of these models.

1.4 References

There are a number of books that are closely related to the topic of this course. The ones we will follow closely are:

1. Polynomial Factorization:
   Cohen: Computational Algebraic Number Theory.

2. Elimination:
   Cox, Little, O'Shea: Ideals, Varieties and Algorithms.

3. Algebraic Models of Computation:
   Blum, Cucker, Shub and Smale: Complexity and Real Computation.

4. Non-uniform methods:
   Burgisser, Clausen, Shokrollahi: Algebraic Complexity Theory.

1.5 Administrivia

There will be some obligation to scribe notes for this course and to both read and write papers based on it. The lecturer hopes that anyone[1] who meets any of the following criteria will both enjoy and benefit from this course.

1. interested in complexity theory
2. interested in number theory
3. interested in coding theory
4. very interested in algebraic methods

1.6 A Review of Some Algebra

Let's review some useful things from algebra. Fields are objects which possess both the operators $+$ and $*$, both of which are associative, commutative, and both of which possess both identities and inverses, except that 0 does not have a multiplicative inverse. Also, $*$ distributes over $+$.

Some common examples of fields are the rational numbers $\mathbb{Q}$, the reals $\mathbb{R}$, the complex numbers $\mathbb{C}$, the integers modulo p for any prime p ($\mathbb{Z}_p$ - we will shortly show this is a field), and rational functions[2] over a field $\mathbb{F}$.

Commutative rings are like fields, except that we drop the restriction that multiplicative inverses must exist. Examples of commutative rings are the integers ($\mathbb{Z}$), and polynomials over a ring ($R[x]$ where R is a ring).

Groups only possess the operation plus. Groups are not necessarily commutative. For our purposes, the only interesting result for finite groups relates to the order of an element. The **order** of an element a is the least positive integer i (if one exists) such that the sum of i copies of a is the identity element.

Theorem 1.1 *For every finite group, the order of any element of the group divides the order of the group.*

One immediate consequence of this theorem is that $a^p = a \pmod{p}$ where p is a prime, which is also referred to as Fermat's Little Theorem.[3]

Between commutative rings and fields, it is possible to come up with a number of interesting axioms, all of which try to capture some property of the integers. To start, let us consider the question of factorization.

Define an integral domain to be a ring R such that no two non-zero elements yield zero when multiplied. Formally:

A ring R is an **integral domain** iff $\forall a, b \in R : ab = 0 \Rightarrow a = 0 \vee b = 0$.

A unit in an integral domain is any element with an inverse. Formally

$u \in R$ is a **unit** iff $\exists v : uv = 1$.

It is useful to create equivalence classes in integral domains using multiplication by units. Two elements belonging to the same equivalence class under multiplication by a unit are said to be associates. Formally,

a and b are **associates** iff $\exists u$ such that u is a unit and $au = b$.

It may be verified "association" yields an equivalence relation. Relating this to our reference ring, the integers, this means that a and $-a$ are associates for every a.

With this in hand, we can now express the notion of irreducibility.

An element $a \in R$ is **irreducible** iff $\forall b, c \in R, bc = a$ implies b is a unit or c is a unit.

[1] Unless you are the type of person who wants to argue over whether zero is an ideal. In this case, you should have a long talk with Madhu.

[2] Rational functions are formally described as a ratio of two formal polynomials over with coefficients from the field.

[3] We prove this using that $\mathbb{Z}_p$ is a field. This property does not hold in general for arbitrary p. Consider $p = 4, a = 2$.

This definition does correspond to our notion of an irreducible. Since in the integers, irreducibles are primes and vice versa, let us examine a prime, such as two. $2 = 2 * 1 = -2 * -1$ are both true. In both cases, one of the factors is a unit.

We are now ready to define unique factorization domains:

> An integral domain R is a **unique factorization domain** (UFD) iff every element is expressible uniquely as a product of finitely many irreducibles (up to associates).

It is worthwhile to pause to note that in an arbitrary integral domain, not every element has a unique factorization. A common example is the ring $\mathbb{Z}[x, y]/(x^2 - y^3)$ (i.e., polynomials with integer coefficients in x and y with the relation $x^2 = y^3$). In this case, we can factor x^2 as $x * x$ or (since $x^2 = y^3$) as $y * y * y$. (Of course, to verify the example one should also check that x and y are irreducible in this domain.) Similarly, there are integral domains where some elements are reducible but are not units and are not expressible as the product of finitely many irreducible elements. We conclude that our definition is "tight." We will mostly work with unique factorization domains, and this is good for us, because irreducibles and primes are synonymous in a unique factorization domain. This is a good time to pause to introduce primes formally.

> An element $p \in R$ is **prime** iff $\forall b, c \in R, p$ divides bc implies p divides b or p divides c.

Both integers and polynomials over an integral domain satisfy the above properties, and so are unique factorization domains. In general, integral domains have nearly all the nice properties of fields. We extract these properties in the next two lemmas.

Lemma 1.2 *Every finite integral domain R is a field.*

Proof We need to show that every non-zero element $a \in R$ has a multiplicative inverse. Consider the sequence $1, a, a^2, a^3, \ldots, a^n, \ldots$ Since R is finite, the sequence will eventually repeat elements. I.e., $\exists i < j$ s.t. $a^i = a^j$. Thus $a^j - a^i = 0 \Leftrightarrow a^i \cdot (a^{j-i} - 1) = 0$. Since $a \neq 0$, we have $a^i \neq 0$ (using R is integral) and hence $a^{j-i} - 1 = 0$ (again using R is integral). Thus we have that a^{j-i-1} satisfies the properties required of a^{-1}. ■

If an integral domain R is not finite, then to get a field from R we work with the field of fractions over R. Formally

> For an integral domain R, the **field of fractions**, denoted $\tilde{R}$, has as elements, pairs from $R \times (R - \{0\})$, under the equivalence $(a, b) \sim (c, d)$ if there exist $\lambda_1, \lambda_2 \in R - \{0\}$ such that $\lambda_1 a = \lambda_2 c$ and $\lambda_1 b = \lambda_2 d$. Addition and multiplication over $\tilde{R}$ are defined as: $(a, b) + (c, d) = (ad + bc, bd)$ and $(a, b) \cdot (c, d) = (ac, bd)$.

Proposition 1.3 *The field of fractions $\tilde{R}$ forms a field.*

The canonical application of this proposition is the creation of field of the rationals, $\mathbb{Q}$, from the domain of the integers, $\mathbb{Z}$. Finally we define the notion of an ideal that are additively closed subsets of rings that are closed under multiplication with any element of the ring. Formally:

> $I \subseteq R$ is an **ideal** if $a, b \in I \Rightarrow a + b \in I$ and $a \in I, c \in R \Rightarrow ac \in I$.

An example of an ideal are multiples of m over integers (written (m)). Ideals will play a significant role in algorithms for quantifier elimination.

6.966 Algebra and Computation	September 14, 1998
Lecture 2	
Lecturer: Madhu Sudan	*Scribe: Sofya Raskhodnikova*

2.1 Primality vs. Irreducibility

Definition 2.1 *$a \in R$ [4] is irreducible if a is not a unit and $x \cdot y = a$ implies that x or y is a unit.*

Definition 2.2 *$p \in R$ is prime if $\forall a, b \in R \;\; p \mid ab \Rightarrow p \mid a$ or $p \mid b$.*

The notions of integral domain and Unique Factorization Domain (UFD) do not coincide.

Lemma 2.3 *Integral domain R is a UFD iff*

1. *Every element in R is expressible as a product of finitely many irreducible elements.*
2. *Every irreducible element is a prime.*

Proof Let $x \in R$ and $x = p_1 p_2 \ldots p_s = q_1 q_2 \ldots q_t$ where $p_1, \ldots, p_s, q_1, \ldots, q_t$ are irreducible. Equivalently,

$$p_1 p_2 \ldots p_s - q_1 q_2 \ldots q_t = 0. \tag{2.1}$$

By the second condition $p_1, \ldots, p_s, q_1, \ldots, q_t$ are primes. Since $p_1 \mid (q_1 \cdot \cdots \cdot q_t)$, by the definition of primality $\exists i \in \{1, \ldots, t\}$ such that $p_1 \mid q_i$. It follows that $q_i = p_1 \alpha$, and α is a unit. Factoring out p_1 in equation (2.1), we get

$$p_1 (p_2 \ldots p_s - \alpha q_1 q_2 \ldots q_{i-1} q_{i+1} \ldots q_t) = 0. \tag{2.2}$$

Using the fact that R is an integral domain, we arrive at

$$p_2 \ldots p_s - \alpha q_1 q_2 \ldots q_{i-1} q_{i+1} \ldots q_t = 0. \tag{2.3}$$

We repeat the process for the remaining p_i's, showing that each of them is one of the q_i's multiplied by a unit, and that $s = t$. This proves that R is a UFD, as required. The other direction is left as an exercise to the reader. ■

2.2 Polynomial Rings Over Rings

2.2.1 Definitions and Basic Properties

Given a ring R we can create a new ring by introducing an indeterminate x. The set of all polynomials in x with coefficients in R forms a ring which is denoted $R[x]$. The degree of a nonzero polynomial **deg(p)** is the largest integer i such that the coefficient p_i of x^i is not zero. Each polynomial of degree n can also be represented as $n+1$-tuple:

$$p(x) = \sum_{i=0}^{n} p_i x^i \text{ is equivalent to } p = (p_0, p_1, \ldots, p_n), \quad p_i \in R \; \forall i.$$

We can also think of a polynomial in $R[x]$ as a list of finitely many elements from R, followed by infinitely many zeroes: $p = (p_0, p_1, \ldots, p_n, 0, \ldots)$.

[4] Throughout the lecture, R denotes a ring and F denotes a field.

Addition and multiplication of polynomials over a ring can be defined naturally. Let $p, q \in R[x]$, $p = (p_0, p_1, \ldots, p_n)$, $q = (q_0, q_1, \ldots, q_m)$, $n \leq m$.

$$\text{Addition:} \qquad p + q = (p_0 + q_0, p_1 + q_1, \ldots, p_m + q_m)$$

$$\text{Multiplication:} \qquad p \cdot q = (C_0, C_1, \ldots, C_{m+n}), \text{ where } C_k = \sum_{i=0}^{k} p_i \cdot q_{k-i}$$

We can check that R[x] satisfies all the axioms of a ring.

Theorem 2.4 (Inheritance properties) *R[x] inherits the following important properties from the ring R:*

1. *$R[x]$ is commutative $\Leftrightarrow$ R is commutative.*
2. *$R[x]$ is an integral domain $\Leftrightarrow$ R is an integral domain.*
3. *$R[x]$ is a UFD $\Leftrightarrow$ R is a UFD.*

Proof

1. Easy.
2. The forward direction is very easy. Suppose that $R[x]$ is an integral domain. Consider $p, q \in R$ such that $pq = 0$. Obviously, p and q are also constant polynomials in the integral domain $R[x]$. Consequently, either $p = 0$ or $q = 0$, and R is an integral domain.

   The other direction is also not too hard. Suppose R is an integral domain. Consider two polynomials $p(x), q(x) \in R[x]$ such that $p(x)q(x) = 0$. To prove that $R[x]$ is an integral domain, we have to show that either $p(x) = 0$ or $q(x) = 0$.

   For contradiction, suppose that both $p(x)$ and $q(x)$ are nonzero. Let p_n be the highest nonzero coefficient of $p(x)$, and q_m be the highest nonzero coefficient of $q(x)$. Then the coefficient of x^{n+m} of the product polynomial is equal to $p_n q_m$. Since the product polynomial is zero, $p_n q_m = 0$. Since R is an integral domain, either $p_n = 0$ or $q_m = 0$. Contradiction. Either $p(x)$ or $q(x)$ has to be zero.
3. This property is known as Gauss's Lemma. We will not prove it in class - but see appendix to this lecture appendix for a proof.

■

Having defined polynomials over a ring, we in fact created polynomials in more than one variable. By definition, $(R[x])[y]$ represents polynomials in y over the ring R[x]. It is the ring of polynomials of the form $\sum_{i,j} c_{ij} x^i y^j$, where $c_{ij} \in R$. We denote this ring by $R[x, y]$. Notice, that we can also define $R[x, y]$ as $(R[y])[x]$.

$$(R[x])[y] = R[x, y] = (R[y])[x].$$

Later we will use the two alternative definitions of $R[x, y]$ to obtain proofs of some nice facts. Notice that $R[x, y]$ inherits the properties of the ring R.

Theorem 2.5 (Division Theorem) *Given $a(x), b(x) \in F[x]$ $\exists!$ quotient $q(x)$, remainder $r(x) \in F[x]$ such that $a(x) = q(x) \cdot b(x) + r(x)$ and $deg(r) < deg(b)$.*

Proof (Division Algorithm)
Let $a^{(1)}(x)$ and $b(x)$ be two polynomials in $F[x]$.

$$a^{(1)}(x) = a^{(1)}_{n_1} x^{n_1} + a^{(1)}_{n_1-1} x^{n_1-1} + \cdots + a^{(1)}_1,$$
$$b(x) = b_m x^m + b_{m-1} x^{m-1} + \cdots + b_1,$$
$$\text{and } n_1 \geq m.$$

At the i^{th} step of the algorithm, we multiply $b(x)$ by $T^{(i)}(x) = \frac{a^{(i)}_{n_i}}{b_m} x^{n_i - m}$, and subtract the result from $a^{(i)}(x)$ to obtain a new polynomial $a^{(i+1)}(x)$.

$$a^{(i+1)}(x) = a^{(i)}(x) - \frac{a^{(i)}_{n_i}}{b_m} x^{n_i - m} \cdot b(x).$$

Let $n_{(i+1)}$ denote the degree of $a^{(i+1)}(x)$. Since $n_{i+1} \leq n_i$, the algorithm will eventually find $a^{(k)}$ with $n_k < m$. Set $r(x) = a^{(k)}(x)$, and $q(x) = \sum_{i=1}^{k-1} T^{(i)}(x)$.

Notice that we can divide by b_m because we are working with polynomials over a field. This algorithm is not useful for bivariate polynomials, but later we will define division in more general settings. The proof of uniqueness is left as an exercise to the reader. ■

2.2.2 GCD

Usually, concepts defined for integers have analogues for univariate polynomials. We show how familiar terminology can be applied to polynomials. We say that $a(x)$ is a **divisor** of $b(x)$ ($a(x) \mid b(x)$) if $b(x) = a(x) \cdot q(x)$ for some $q(x)$. If $g(x) \mid a(x)$ and $g(x) \mid b(x)$, $g(x)$ is called a **common divisor** of $a(x)$ and $b(x)$. Finally, $g(x)$ is a **greatest common divisor** (GCD) of $a(x)$ and $b(x)$, denoted $\gcd(a(x), b(x))$, if $g(x)$ has the largest degree among all common divisors of $a(x)$ and $b(x)$.

Lemma 2.6 *Let $S = \{p(x) \in F[x]$ such that $\exists u, v \in F[x]$ with $p = ua + vb\}$. The smallest degree polynomial $g \in S$ is*

1. *a common divisor of a and b;*
2. *$gcd(a, b)$;*
3. *unique up to multiplication by units.*

Proof By definition, $g = ua + vb$ for some $u, v \in F[x]$.

1. Suppose, for contradiction, that g does not divide a. Let r be the remainder

   $$r = a - qg = a - q[ua + vb] = (1 - qu)a + (-qv)b.$$

   Then $r \in S$, and has a smaller degree than g. Contradiction.

2. Let $g\prime$ be a common divisor of a and b. Then $g\prime \mid a$ and $g\prime \mid b$, and consequently $g\prime \mid f = ua + vb \;\; \forall u, v$. By definition of g, $g\prime \mid g$. Since any common divisor of a and b divides g, g is $\gcd(a, b)$.

3. Suppose $g\prime = \gcd(a, b)$. By (2) $g \mid g\prime$, i.e., $g = \alpha \cdot g\prime$. Since, by definition of gcd, deg(g)=deg($g\prime$), α is a unit.

■

2.2.3 Evaluation of polynomials

Polynomials (of low degree) are very interesting to coding theory and complexity theory. For these applications, we need the notion of evaluation of polynomials. Given $p(x)$, define function $EV_p : F \to F$ as $EV_p(\alpha) = \sum_{i=0}^{n} p_i \alpha^i$. $EV_p(\alpha)$ is usually denoted by $p(\alpha)$. The following properties of the evaluation map can be verified syntactically.

$$\begin{aligned} \forall p, q, \alpha \quad & EV_p(\alpha) + EV_q(\alpha) = EV_{p+q}(\alpha) \\ & EV_p(\alpha) \cdot EV_q(\alpha) = EV_{pq}(\alpha) \end{aligned}$$

Theorem 2.7 $p(\alpha) = 0$ *iff* $(x - \alpha) \mid p(x)$

Proof By Division theorem $\exists q, r$ such that $p(x) = q(x)(x - \alpha) + r(x)$ with $\deg(r) < \deg(x - \alpha) = 1$, or equivalently with $r \in F$.

$$\begin{aligned}
&0 = p(\alpha) = q(\alpha)[\alpha - \alpha] + r \Rightarrow \\
&r = 0 \Rightarrow \\
&p(x) = q(x)(x - \alpha)
\end{aligned}$$

■

Definition 2.8 *α is a* <u>*root*</u> *of $p(x)$ if $p(\alpha) = 0$.*

Lemma 2.9 *If $\alpha_1, \ldots, \alpha_{n+1}$ are distinct roots of $p(x)$, then $\Pi_{i=1}^{n+1}(x - \alpha_i) \mid p(x)$.*

Proof By theorem 2.7, $p(x) = (x - \alpha_1)q_1(x)$. Evaluating $p(x)$ at α_2, we get $0 = (\alpha_1 - \alpha_2)q_1(\alpha_2)$. Note that $\alpha_1 - \alpha_2 \neq 0$ because the roots are distinct. Since we are working with polynomials over a field (hence an integral domain), $q_1(\alpha_2) = 0$. By theorem 2.7, $(x - \alpha_2) \mid q_1(x)$. As before, $q_1(x) = (x - \alpha_2)q_2(x)$, and $p = (x - \alpha_1)(x - \alpha_2)q_2(x)$. We repeat the above derivation until we show that $\Pi_{i=1}^{n+1}(x - \alpha_i) \mid p(x)$. ■

Theorem 2.10 *A nonzero polynomial $p(x) \in F[x]$ of degree n has at most n roots.*

Proof Suppose, for contradiction, that $\alpha_1, \alpha_2, \ldots, \alpha_{n+1}$ are distinct roots of $p(x)$. Then by lemma 2.9, $\Pi_{i=1}^{n+1}(x - \alpha_i) \mid p(x)$. But a polynomial of degree $n + 1$ cannot divide $p(x)$. Contradiction. ■

Exercise Prove the analogue of the previous theorem for bivariate polynomials.

$$\begin{array}{ll}
\text{Given} & p(x, y) \in F[x, y], \\
\text{show} & p(x, q(x)) = 0 \Rightarrow y - q(x) \mid p(x, y).
\end{array}$$

Theorem 2.11 (Interpolation Theorem) *Given $x_1, y_1, \ldots, x_n, y_n \in F$ with $x_i \neq x_j$ for $i \neq j$, $\exists$ polynomial $p^{(n)} \in F[x]$ of degree less than or equal to $n - 1$, such that $p^{(n)}(x_i) = y_i$.*

Proof We prove this theorem by giving an inductive construction of the polynomial $p^{(n)}(x)$. If $n = 1$, then $p^{(1)}(x) = y_1$. Suppose we found $p^{(i)}(x)$ of degree less than or equal to $i - 1$, such that $p^{(i)}(x_j) = y_j$ for $j = 0, \ldots, i$, and we are looking for $p^{(i+1)}(x)$. If $p^{(i)}(x_{j+1}) = y_{j+1}$, we are done. Otherwise, let

$$d^{(i+1)}(x) = p^{(i+1)}(x) - p^{(i)}(x).$$

Our goal is to find a suitable $d^{(i+1)}(x)$ and, consequently, $p^{(i+1)}(x)$. For all $j = 0, \ldots, i$, $p^{(i+1)}(x_j) = p^{(i)}(x_j)$, and hence $d^{(i+1)}(x_j) = 0$. The degree of $d^{(i+1)}(x)$ is less than or equal to i. The polynomial $q^{(i+1)}(x) = \Pi_{j=1}^{i}(x - x_j)$ satisfies the above conditions. However, in general $q^{(i+1)}(x_{j+1}) \neq d^{(i+1)}(x_{j+1})$. Since $d^{(i+1)}(x_{j+1}) = y_{j+1} - p^{(i)}(x_{j+1})$, we define

$$d^{(i+1)}(x) = \frac{y_{j+1} - p^{(i)}(x_{j+1})}{q^{(i+1)}(x_{j+1})} q^{(i+1)}(x).$$

And finally,

$$p^{(i+1)}(x) = p^{(i)}(x) + d^{(i+1)}(x) = p^{(i)}(x) + \frac{y_{j+1} - p^{(i)}(x_{j+1})}{q^{(i+1)}(x_{j+1})} q^{(i+1)}(x).$$

By construction, $p^{(i)}(x)$ satisfies the required conditions. ■

The proof of the interpolation theorem yields the following algorithm for the interpolation problem:

Interpolate($\{(x_1, y_1), \ldots, (x_n, y_n)\}$); /* x_i's distinct */

Initialize

$p^{(0)}(x) \leftarrow \mathbf{0}$.

$q^{(0)}(x) \leftarrow \mathbf{1}$.

Iterate for $i = 1$ to n;

$\lambda_i \leftarrow \frac{y_i - p^{(i)}(x_i)}{q^{(i-1)}(x_i)}$;

$p^{(i)}(x) \leftarrow p^{(i-1)}(x) + \lambda_i q^{(i-1)}(x)$;

$q^{(i)}(x) \leftarrow q^{(i-1)}(x) \cdot (x - x_i)$;

Output($p^{(n)}$).

Appendix

We work towards Gauss's Lemma by first showing that the ring of polynomials over a field F is a unique factorization domain.

Lemma 2.12 *For any field F, $F[x]$ is a UFD.*

Proof We apply Lemma 2.3. To do so we need to verify that every element $p \in F[x]$ has a finite factorization into irreducibles. This is easily established: the number of irreducible factors is at most the degree of p. Thus to prove the lemma it suffices to show that every irreducible polynomial p is also a prime. I.e., if $p|ab$ then $p|a$ or $p|b$.

Assume $p \not| a$ and $p \not| b$. Then $\gcd(p, a) = 1$ and $\gcd(p, b) = 1$. (Since the common divisor must divide p and p is irreducible.) By Lemma 2.6, we know that we can express the gcd's as a linear combination of the arguments. I.e., there exist u_1, u_2, v_1 and u_2 such that

$$u_1 p + v_1 a = 1 \text{ and } u_2 p + v_2 b = 1.$$

Multiplying the above we get

$$1 = (u_1 p + v_1 a)(u_2 p + v_2 b) = (u_1 u_2 p + u_1 v_2 b + u_2 v_1 a)p + (v_1 v_2)ab.$$

In other words 1 can be expressed as a linear combination of p and ab, thus implying that $\gcd(p, ab) = 1$ contradicting the fact that $p|ab$. ■

Lemma 2.13 (Gauss) *$R[x]$ is a UFD $\Leftrightarrow$ R is a UFD.*

Proof It is easy to see that if R is not a UFD, then $R[x] \supset R$ is not a UFD. The harder direction is the other way around. The main idea behind this direction is to use the fact that the ring $\tilde{R}[x]$ is a UFD, where $\tilde{R}$ is the field of fractions of R. [5]

Again we use Lemma 2.3. To do so we need to verify first that any $p(x) \in R[x]$ can be factored into finitely many irreducibles. This time we need to be slightly careful, since some factors may be elements of R. Let $p_0, \ldots, p_n$ be the coefficients of $p(x)$. Then notice that $a \in R$ divides $p(x)$ if and only if $a|p_i$ for every $i \in \{0, \ldots, n\}$. Thus we first write $p(x) = a \cdot q(x)$ where $a \in R$ and the coefficients of $q(x)$ have no non-trivial common factors. Since R is a UFD, a is expressible as a product of finitely many irreducibles. Divisors of $q(x)$, on the other hand have positive degree in x and thus q factors into at most $\deg_x q$ irreducibles. Thus to establish the lemma it suffices to show that every irreducible element of $R[x]$ is prime. This is established below.

Claim 2.14 *If $p(x)$ is irreducible in $R[x]$, then:*

1. *$p(x)$ is irreducible (and hence prime) in $\tilde{R}[x]$.*

[5] This idea of relying on the properties of $\tilde{R}$ rather than R is a standard method, and can be quite powerful. This method is often used to argue about $R = F[x_1, \ldots, x_n]$, the ring of polynomials in $x_1, \ldots, x_n$ over a field F. In such cases $\tilde{R}$ is also denoted $F(x_1, \ldots, x_n)$ — note the subtle distinction in the notation: $F(\cdots)$ vs. $F[\cdots]$.

2. $p(x)$ is prime in $R[x]$.

Proof

1. Say $p(x) = \tilde{a}(x)\tilde{b}(x)$ for $\tilde{a}(x), \tilde{b}(x) \in \tilde{R}(x)$. Then by taking the least common multiple of the denominators of the coefficients of $\tilde{a}(x)$ and $\tilde{b}(x)$ and separating them out, we get $cp(x) = a(x)b(x)$ where $a(x), b(x) \in R[x]$, $c \in R$. If c is a unit in R, then we are done, since we find that p is reducible in $R[x]$. If not, then we need to find some contradiction. Let $c = c_1 c_2 \dots c_t$ be a prime factorization of c. We may assume that c_i does not divide some coefficient a_j of $a(x)$ and b_k of $b(x)$ (else we could remove common factors from both sides). Further, let j and k be the least indices with this property: i.e., $c_i | a_{j'}$ for $j' < j$ and $c_i | b_{k'}$ for $k' < k$. Then, consider the coefficient of x^{j+k} in the equation $cp(x) = a(x)b(x)$. On the LHS we have a multiple of c_i and on the right we have the summation $\sum_{l=0}^{j+k} a_l b_{j+k-l}$. All terms in this summation, except the term $a_j b_k$ are divisible by c_i. Rearranging terms, we get that $a_j b_k$ is divisible by c_i which gives a contradiction.

2. Let $p(x)$ be irreducible in $R[x]$. Then we have that the gcd of the coefficients of $p(x)$ is 1. Now suppose $p(x)|a(x)b(x)$. Then since $p(x)$ is prime in $\tilde{R}[x]$ we have that $p(x)|a(x)$ or $p(x)|b(x)$ in $\tilde{R}[x]$. Say it is the former. Then we can find $f(x) \in R[x]$ and $c \in R$ such that $ca(x) = f(x)p(x)$, and furthermore c does not have a common divisor with all coefficients of f. As in Part (1), we can show that every prime factor of c either divides every coefficient of f or divides every coefficient of p. In either case we have a contradiction - hence c must be a unit and thus $a(x) = (c^{-1}f(x))p(x)$ and thus $p(x)|a(x)$ over $R[x]$.

■■

6.966 Algebra and Computation	September 16, 1998
Lecture 3	
Lecturer: Madhu Sudan	*Scribe: Yevgeniy Dodis*

In this lecture we will study an application of polynomials to coding theory (Reed-Solomon code), GCD algorithm and its applications, construction of finite fields, Chinese remainder theorem (CRT) and begin our study of resultant.

3.1 Application of Polynomials to Coding Theory

Coding theory is concerned with the problem of reliable communication over a noisy channel. Assume we are given a source message $m \in \{0,1\}^k$, that we would like to transmit reliably in the presence of communication errors. We can try to encode the message by adding some redundancy to it in such a manner that one can reconstruct the original message even when large parts of the encoded message get corrupted. More formally, we wish to design an *encoding function* $E : \{0,1\}^k \to \{0,1\}^n$ where $n > k$. This function E defines a *code* $\mathcal{C} = \{E(m) | m \in \{0,1\}^k\}$. We would like all the words of the code (called the *codewords*) to be very far "apart" so that even a large corruption of a codeword enables one to reconstruct the message. Given $a, b \in \{0,1\}^n$ we let the *Hamming distance* $d(a,b)$ between a and b be the number of symbols they differ in. We define the *distance* of the code to be

$$d(\mathcal{C}) = \min_{c,c' \in \mathcal{C}, c \neq c'} d(c,c') = \min_{m,m' \in \{0,1\}^k, m \neq m'} d(E(m), E(m'))$$

The main problem of the theory of Error Correcting Codes is to construct a code with a small length n (compared to k), but a large distance d. Such constructions are quite non-trivial and algebra often comes as the main tool in designing such codes. The algebraic constructions do not directly give codes over a binary alphabet though, so we need to extend our definition of error-correcting codes to larger alphabets and in particular to encoding functions $E : \mathbb{F}^k \to \mathbb{F}^n$ over a field $\mathbb{F}$. (In many standard constructions of *binary* error-correcting codes, *non-binary* codes such as the one described below are used as starting points.)

3.1.1 Reed-Solomon (RS) Code

Let us view the source message as specifying the coefficients of some polynomial of degree at most $(k-1)$. So given $\bar{m} = (m_1, \ldots, m_k)$, $m_i \in \mathbb{F}$, we define the associated polynomial $M(x) = \sum_{i=1}^k m_i x^{i-1}$. Fix arbitrary n *distinct* elements $\alpha_1, \ldots, \alpha_n$ of $\mathbb{F}$ (thus, we must assume that $|\mathbb{F}| \geq n$). The encoding function $E^{RS}_{\alpha_1,\ldots,\alpha_n}(\bar{m}) = \langle M(\alpha_1), \ldots, M(\alpha_n) \rangle$. We claim that this is a great (in fact, "perfect") code. More specifically, let us compute its distance. Let $\bar{m}_1 \neq \bar{m}_2$. Then $M_1(x) \not\equiv M_2(x)$. Since encoding of $\bar{m}_i$ is a value of $M_i(x)$ at n points, and two non-equal polynomials of degree at most $k-1$ can agree in at most $k-1$ points, the encodings differ in at least $n-(k-1) = n-k+1$ places, so $d(\mathcal{C}) \geq n-k+1$. It is easy to show that the distance is exactly $n-k+1$; in fact, it follows from the more general fact below.

Exercise 3.1 *Show that for any $E : \mathbb{F}^k \to \mathbb{F}^n$ there are 2 distinct messages m_1 and m_2 s.t. $d(E(m_1), E(m_2)) \leq n-k+1$. Hence $d(\mathcal{C}) \leq n-k+1$.*

Hence, RS codes are the best possible codes in terms of minimal distance. Their only disadvantage is the requirement that $|\mathbb{F}| \geq n$.

3.2 GCD algorithm

Our first look at a non-trivial algebraic algorithm is the Euclidean algorithm for finding the greatest common divisor (GCD). We start by presenting the algorithm to find the GCD of two integers. We then extend it to finding GCD's of polynomials.

3.2.1 GCD for Integers

It is very easy to find GCD of two integers if one knows their factorization. However, factoring integers is believed to be hard (from computational perspective). The remarkable fact about the algorithm is the ability to find the GCD without factoring. Here is the algorithm.

$\gcd(a, b)$, $0 \leq b \leq a$.
  IF $b = 0$ THEN OUTPUT a,
  ELSE EXPRESS $a = b \cdot q + r$, WHERE $0 \leq r < b$, AND OUTPUT $\gcd(b, r)$.

The correctness is clear as any common factor of a and b must divide r, and any common factor of r and b must divide a. We only need to argue the efficiency.

Claim 3.2 $r \leq a/2$

Indeed, as $q \geq 1$, we have $a = bq + r \geq b + r \geq 2r$. Thus, the algorithm terminates in $\log(ab)$ iterations as at each iteration the product ab decreases by at least a factor of 2.

Notice that if $g = \gcd(a, b)$, there are some $u, v \in \mathbb{Z}$ s.t. $g = u \cdot a + v \cdot b$. As we will see, it is often convenient to have these u and v. A trivial modification of our algorithm allows us to achieve it. This algorithm is called the *extended GCD* algorithm.

EXTENDED-$\gcd(a, b)$, $0 \leq b \leq a$.
  IF $b = 0$ THEN OUTPUT $(a, 1, 0)$,
  ELSE EXPRESS $a = b \cdot q + r$, WHERE $0 \leq r < b$,
  LET $(g, u', v') =$ EXTENDED-$\gcd(b, r)$, AND OUTPUT $(g, v', u' - v' \cdot q)$.

To verify correctness notice that if $g = u'b + v'r$, then $g = u'b + v'(a - qb) = v'a + (u' - v'q)b$.

3.2.2 GCD for Polynomials

We now address the question of extending our GCD algorithm to polynomials. Integers had natural ordering based on their magnitude. The natural ordering for the polynomials is based on their degree: $a(x) \leq b(x)$ iff $\deg(a) \leq \deg(b)$. With this in mind, the above algorithm works as it is for the polynomials. The main step to verify is that the division step above does have an appropriate analog in the case of polynomials since if $a(x) \geq b(x)$ then we can uniquely express $a(x) = q(x)b(x) + r(x)$, where $\deg(r) < \deg(b)$. Here the invariant after one iteration is that $\deg(r) < \deg(a)$, so that the sum of the degrees of a and b decreases. Thus, we have termination in at most $\deg(a) + \deg(b)$ iterations.

3.3 Applications of GCD

We give just a few handy applications of the extended GCD algorithm.

3.3.1 Inverse Computations in $\mathbb{Z}_p$

Addition and multiplication in $\mathbb{Z}_p$ are very easy to implement. We show that the same holds for the inverse operation as well. One clever way to compute the inverse in polynomial time is to use Fermat little theorem stating that for any $a \in \mathbb{Z}_p^*$, $a^{p-1} \equiv 1 \bmod p$, Thus $a^{-1} \equiv a^{p-2}$. Since exponentiation is easy via repeated squaring, we can compute $a^{-1} \bmod p$ using $O(\log p)$ multiplications. Using GCD, we obtain an alternative way to compute the inverse. It will have an advantage to generalize to the situation of computing $a^{-1} \bmod n$ for composite n, where $a \in \mathbb{Z}_n^*$, i.e. $\gcd(a, n) = 1$. Fermat's trick does not work here. Even though by Euler's theorem $a^{\varphi(n)} \equiv 1 \bmod n$, it is believed to be a hard problem to compute $\varphi(n)$ from n alone.

Without further delay, here is the algorithm to compute the inverse of a even for composite n, whenever $\gcd(a, n) = 1$. Use the extended GCD to find u, v s.t. $1 = ua + vn$. Then $ua \equiv 1 \bmod n$, so $u \bmod n = a^{-1} \bmod n$. Notice that this process takes $O(\log n)$ divisions.

3.3.2 Construction and Arithmetic of any Finite Field

Theorem 3.3 *For any prime p and any $n \geq 1$ there exists a unique field of order p^n. This field is denoted $GF(p^n)$ (Galois Field of order p^n). And these are the only finite fields.*

We prove this theorem completely later in the course, but now we show a part of it by demonstrating how to construct these fields. Suppose $f(x) \in \mathbb{Z}_p[x]$ is an irreducible polynomial of degree k. We will prove later that such f exists. Let $\mathbb{F} = \{q(x) \in \mathbb{Z}_p[x] | \deg q < k\}$. Define the addition on $\mathbb{F}$ in the natural way and let multiplication on $\mathbb{F}$ be the multiplication in $\mathbb{Z}_p[x]$ modulo $f(x)$. Thus, if $q_1(x)q_2(x) = \alpha(x)f(x) + \beta(x)$, where $\deg \beta < \deg f = k$, we let $q_1 \star q_2 = \beta$. By the uniqueness of the division procedure, the multiplication $\star$ is well defined. All the properties of the field are clear for $\mathbb{F}$ except the existence of inverses. Take any non-zero $q \in \mathbb{F}$. As $\deg q < \deg f$, we must have that $\gcd(f, q) = 1$ as f is irreducible. By the extended GCD, we can find polynomials g and r s.t. $1 = gf + rq$. But then $rq \equiv 1 \bmod f$, so $r \bmod q$ is the (easily seen to be unique) inverse of q. Thus, $\mathbb{F}$ is indeed a field. Moreover, all field operations are easily computable in polynomial time (given f). The order of $\mathbb{F}$ is p^k as there are k free coefficients in choosing a polynomial of degree at most $k - 1$.

3.3.3 Chinese Remainder theorem (CRT)

Theorem 3.4 (CRT) *Let $m_1, \ldots, m_n$ be mutually relatively prime integers, i.e. $\gcd(m_i, m_j) = 1$ for $i \neq j$. Then for any $\alpha_1, \ldots, \alpha_n$, $\alpha_i \in \mathbb{Z}_{m_1}$, there exists a* unique *integer $0 \leq \alpha \leq m_1 \cdot m_2 \ldots \cdot m_n$ s.t. for any $1 \leq i \leq n$,*

$$\alpha \equiv \alpha_i \bmod m_i$$

Moreover, this α can be found in polynomial time.

Clearly, it suffices to show the theorem for $n = 2$ only. Thus, given $m_1, m_2, \alpha_1, \alpha_2$ s.t. $\gcd(m_1, m_2) = 1$, we wish to find $0 \leq \alpha < m_1 m_2$ s.t. $\alpha \equiv \alpha_i \bmod m_i$. We use the extended GCD to find u and v s.t. $um_1 + vm_2 = 1$. Let $M_1 = vm_2$, $M_2 = um_1$. Then $M_1 \equiv 1 \bmod m_1$, $M_1 \equiv 0 \bmod m_2$, $M_2 \equiv 0 \bmod m_1$, $M_2 \equiv 1 \bmod m_2$. Let $\alpha = \alpha_1 M_1 + \alpha_2 M_2 \bmod m_1 m_2$. Then α clearly satisfies the conditions. The uniqueness of α is clear as if there is another $\beta \neq \alpha$ like that, we would have $0 < |\alpha - \beta| < m_1 m_2$ and both m_1 and m_2 divide $|\alpha - \beta|$. As $\gcd(m_1, m_2) = 1$, this is impossible.

In other words, the theorem claims the following bijection (in fact, isomorphism):

$$\alpha \Longleftrightarrow (\alpha \bmod m_1, \ldots, \alpha \bmod m_n)$$

between $\mathbb{Z}_{m_1 m_2 \ldots m_n}$ and $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \ldots \times \mathbb{Z}_{m_n}$.

We can also define a good error correcting code using the CRT. Let $m_1 \leq m_2 \leq \ldots m_k \leq m_{k+1} \leq \ldots \leq m_{2k}$ be distinct l-bit primes. We can easily have $l \approx (1 + \epsilon) \log k$ by the prime number theorem. Let $N = m_1 \cdot \ldots \cdot m_k$. For any $n < N$ define $E(m) = \langle (n \bmod m_1), \ldots (n \bmod m_{2k}) \rangle$. Let $\mathbb{F} = GF(2^l)$ and let us view E as a function $E : \mathbb{F}^k \to \mathbb{F}^{2k}$. Then E maps strings of length k over $\mathbb{F}$ into ones of length $2k$. Moreover, we claim that the minimum distance of this code is at least $k + 1$. Indeed, let $E_i(n) = (n \bmod m_i)$. If $n_1 \neq n_2$, $n_1, n_2 < N$, then for any $I \subseteq [2k]$, $|I| = k$, there is $i \in I$ s.t. $E_i(n_1) \neq E_i(n_2)$. This is because equality for all $i \in I$ would imply by the CRT that $n_1 \equiv n_2 \bmod (\prod_{i \in I} m_i)$. As m_i's go in an increasing order, $\prod_{i \in I} m_i \geq N$, so we get that $n_1 = n_2$. Thus, out of any k components of the code, at least 1 of them must be distinct for any $n_1 \neq n_2$, so at most $k - 1$ components can be the same. To summarize, our code has length $2k$, distance at least $k + 1$ but is defined over a large field of size roughly $k^{1+\epsilon}$. (For people familiar with the notion of a "linear" code, we stress that this is not a linear code.)

3.4 Resultant

Let us now view the problem of determining whether $a(x)$ and $b(x)$ are relatively prime from a more algebraic point of view. Suppose $\gcd(a(x), b(x)) = 1$. Then there are $u(x)$ and $v(x)$ s.t.

$$u(x)a(x) + v(x)b(x) = 1 \tag{3.1}$$

We can also easily arrange that $\deg(u) < \deg(b)$ and $\deg(v) < \deg(a)$. Indeed, if $u'a + v'b = 1$ and $u' = qb + u$, where $\deg(u) < \deg(b)$, then $(qb + u)a + v'b = ua + (qa + v')b = ua + vb$, where $v = qa + v'$. We have by construction that $\deg(u) < \deg(b)$, so that $\deg(ua) < \deg(a) + \deg(b)$. This and (3.1) imply that $\deg(v) < \deg(a)$. Let $\deg(a) = m$, $\deg(b) = n$. Then the problem of determining whether $\gcd(a, b) = 1$ is equivalent to finding polynomials $u(x) = u_0 + u_1 x + \ldots u_{n-1}x^{n-1}$, $v(x) = v_0 + v_1 x + \ldots + v_{m-1}x^{m-1}$ s.t. identity (3.1) holds.

Let us view it as a linear system with unknowns $u_0, \ldots, u_{n-1}, v_0, \ldots, v_{m-1}$, where $u_0 a_0 + v_0 b_0 = 1$ and for any $i > 0$ the coefficient of x^i of the polynomial $u(x)a(x) + v(x)b(x)$ is 0. Hence, we have a system of linear equations (where $a_{-i} = b_{-i} = 0$ for $i > 0$):

$$a_0 u_0 + b_0 v_0 = 1 \tag{3.2}$$

$$\sum_{j=0}^{i} a_{i-j} u_j + \sum_{j=0}^{i} b_{i-j} v_j = 0 \qquad 1 \le i \le n + m - 1 \tag{3.3}$$

In matrix form,

$$m+n \left\{ \left(\begin{array}{cccc|cccc} a_0 & & & & b_0 & & & \\ a_1 & \ddots & & & b_1 & \ddots & & \\ \vdots & \ddots & \ddots & & \vdots & \ddots & b_0 & \\ a_m & & \ddots & a_0 & \vdots & & b_1 & \\ & \ddots & & a_1 & b_n & & \vdots & \\ & & \ddots & \vdots & & \ddots & \vdots & \\ & & & a_m & & & b_n & \\ \multicolumn{4}{c|}{\underbrace{\hphantom{a_0 \quad a_1 \quad a_2 \quad a_m}}_{n}} & \multicolumn{4}{c}{\underbrace{\hphantom{b_0 \quad b_1 \quad b_2 \quad b_n}}_{m}} \end{array} \right) \right. \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ u_{n-1} \\ v_0 \\ v_1 \\ \vdots \\ v_{m-1} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

The determinant of this matrix $M_{a,b}$ is called the *resultant* of $a(x)$ and $b(x)$: $\operatorname{Res}(a, b) \stackrel{\text{def}}{=} \det(M_{a,b})$. The resultant is a classical example of an "algebraic invariant" and plays an important role in algorithmic algebra. We will study its properties next time; and in particular show that

$$\gcd(a, b) = 1 \iff \operatorname{Res}(a, b) \neq 0 \ \ (\iff \det(M_{a,b}) \neq 0)$$

6.966 Algebra and Computation September 21, 1998

Lecture 4

Lecturer: Madhu Sudan *Scribe: Leonid Reyzin*

Resultant

First we recall the definition of a resultant from the previous lecture. Let $a, b \in F[x]$ be polynomials of degrees m, n, respectively. We can ask a question of whether $(a, b) = 1$ or, equivalently, whether there exist two polynomials u, v of degrees less than n, m, respectively, such that $au + bv = 1$ (here and below we use (a, b) to denote a greatest common divisor of a and b). This question can be considered as a system of $m+n$ linear equations in $m + n$ variables, where the variables are the n coefficients of u and m coefficients of v, and the coefficients for the equations come from the coefficients of a and b. We define $\mathrm{Res}(a, b)$ to be the determinant of the matrix for this system.

More precisely, let $a = \sum_{i=0}^{m} a_i x^i$ and $b = \sum_{i=0}^{n} b_i x^i$ and $M_{a,b}$ be the $(n + m) \times (n + m)$ matrix given by

$$(M_{a,b})_{ij} = \begin{cases} a_{i-j} & \text{if } j \leq n \text{ and } 0 \leq i - j \leq m. \\ b_{i-(j-n)} & \text{if } j > n \text{ and } 0 \leq i - (j - n) \leq n. \\ 0 & \text{otherwise.} \end{cases}$$

Let w be the $m + n$-dimensional vector of unknowns. I.e., the ith coordinate of w is set to u_{i-1} if $i \leq n$ and v_{i-n-1} if $i > n$, where the u_j's and v_j's are the coefficients of u and v. Then the linear system to be solved is given by $M_{a,b} w = (100\ldots0)^T$. The resultant of a and b, $\mathrm{Res}(a, b)$, is defined to be the determinant of the matrix $M_{a,b}$.

Theorem 4.1 $\mathrm{Res}(a, b) \neq 0$ *if and only if* $(a, b) = 1$.

Proof The forward direction is easy: if $\mathrm{Res}(a, b) \neq 0$, then the linear system $M_{a,b} w = (100\ldots0)^T$ has a solution (by Cramer's rule), so $\exists u, v$ such that $au + bv = 1$, so $(a, b) = 1$.

To show the other direction, assume $(a, b) = 1$. Then $\exists u, v \in F[x]$ of degrees less than n, m, respecitvely, such that $au + bv = 1$. We also claim that for all i, $0 \leq i < m + n$, $\exists u^{(i)}, v^{(i)} \in F[x]$ of degrees less than n, m, respectively, such that $au^{(i)} + bv^{(i)} = x^i$. This is because we can write, using the division algorithm, $ux^i = qb + r$ for $\deg(r) < \deg(b)$ and set $u^{(i)} = r = ux^i - qb$ and $v^{(i)} = vx^i + qa$. It is clear that $au^{(i)} + bv^{(i)} = aux^i - qab + bvx^i + qab = x^i(au + bv) = x^i$. It is also clear that $\deg(u^{(i)}) < \deg(b) = n$. To see that $\deg(v^{(i)}) < m$, note that $au^{(i)}$ has degree at most $m + n - 1$, therefore, $bv^{(i)}$ has to have degree at most $m+n-1$ (otherwise the equation $au^{(i)} + bv^{(i)} = x^i$ could not possibly hold, because $i \leq m+n-1$), so $\deg(v^{(i)}) \leq m + n - 1 - \deg(b) = m - 1$. Thus, the claim holds. The claim implies that the system of linear equations in the coefficients of u and v has a solution for any right-hand side (not just for $(1, 0, 0, \ldots, 0)^T$, so the matrix of the system has to have full rank, so its determinant $\mathrm{Res}(a, b) \neq 0$. ■

Another way to view the resultant is as follows. Consider the case when a and b are not relatively prime: $\exists g \in F[x]$ such that $\deg(g) > 0$ and $(a, b) = g$. Then let $p = b/g$ and $q = -a/g$. Then $pa + qb = 0$. Thus, if a and b are not relatively prime, there exist $p, q \in F[x]$, $(p, q) \neq (0, 0)$ of degrees less than n, m, respectively, such that $pa + qb = 0$. Conversely, if such p and q exist, then a and b are not relatively prime (because if $pa = -qb$, then $b|pa$; but $\deg(p) < \deg(b)$ and $p \neq 0$, so a and b have to have a non-trivial common factor because $F[x]$ is a unique factorization domain). In this case we get the linear system $M_{a,b} w = \vec{0}$, where w now represents the coefficients of p and q rather than u and v. The primary difference from the previous case is that now the right-hand side is 0. a and b are not relatively prime if and only if a non-trivial solution exists, which is equivalent to saying that the matrix doesn't have full rank, or $\mathrm{Res}(a, b) = 0$.

All the discussions about the resultant and the GCD so far have assumed that F is a field. What happens if we relax this assumption? The resultant can be still be defined in the same way for polynomials over arbitrary commutative rings; our proofs, however, relied on for the properties of the field and may not hold for arbitrary rings. We will now consider the case of general unique factorization domains I. Since I is an integral domain, it is possible to consider the quotient field Q of I and use properties of $Q[x]$ to infer properties of $I[x]$.

Lemma 4.2 *(A variant of Gauss's lemma.) Two polynomials $a, b \in I[x]$ share a common factor of degree greater than 0 if and only if they share a common factor in $Q[x]$ of degree greater than 0.*

The discussions below show how to prove this lemma. (See the appendix of the notes for Lecture 2 for more details.)

Consider now the case of $I = \mathbb{Z}$, the integers. Then Q is the rationals. Two polynomials a, b with integer coefficients are relatively prime in $Q[x]$ if and only if $\exists u', v' \in Q[x]$ such that $av' + bu' = 1$ and $\deg(u') < \deg(a)$ and $\deg(v') < \deg(b)$. Equivalently, we can multiply through by the least common denominator N of all the coefficients of u' and v', to get that a, b don't share a factor of degree greater than 0 in $\mathbb{Z}[x]$ if and only if $\exists u, v \in \mathbb{Z}[x], N \in \mathbb{Z}$ such that $av + bu = N$ and $\deg(u) < \deg(a)$ and $\deg(v) < \deg(b)$.

How large can this N be? Let's assume that a, b are of degrees m, n, respectively, and their coefficients are at most l-bit integers. Remember that coefficients of u' and v' are given by Cramer's rule, which has $\mathrm{Res}(a, b)$ in the denominator. Thus, $0 < N \leq \mathrm{Res}(a, b)$. Remember also that $\mathrm{Res}(a, b)$ is the determinant of an $(m+n) \times (m+n)$ matrix with entries that are at most l-bits each, so N has at most $l(m+n)\log((m+n)!) = (m+n)^{O(1)}l$ bits. This leads to the following proposition.

Proposition 4.3 *Let $a, b \in \mathbb{Z}[x]$ be of degree m, n respectively, with coefficients of a, b being l-bit integers. Then there exist polynomials $u, v \in \mathbb{Z}[x]$ and $N \in \mathbb{Z} - \{0\}$ such that $ua + vb = N$ iff a and b have no common factor of positive degree in x. Furthermore if N as claimed above exists then there exists one that is at most $(m+n)!2^{(m+n)l}$.*

We can go through the same exercise for bivariate polynomials $F[x, y]$ over a field F. We just need to consider $F[x, y]$ as $F[x][y]$. Its elements are polynomials in y over an integral domain $F[x]$. We can consider the quotient field $F(x)$ of $F[x]$ (note the significant difference between parentheses and brackets in this notation). We get that $a(y), b(y)$ with coefficients in $F[x]$ are relatively prime in $F(x)[y]$ if and only if $\exists u'(y), v'(y) \in F(x)[y]$ such that $av' + bu' = 1$ and $\deg(u') < \deg(a)$ and $\deg(v') < \deg(b)$. Again, multiplying by the least common denominator $p(x)$ of the coefficients of u' and v', we get that $a(x, y)$ and $b(x, y)$ don't share a factor in $F[x, y]$ of degree in y greater than 0 if and only if $\exists u, v \in F[x, y]$ and $\exists p \in F[x]$ such that $au + bv = p$.

Again we might ask how large p is. Let's assume that the degrees of a and b in x are no more than d, and that their degrees in y are m and n, respectively. By the same reasoning as before, $\deg(p(x)) \leq (m+n)d$.

Proposition 4.4 *Let $a, b \in F[x, y_1, \ldots, y_n]$ be polynomials of degree at most d in x, $y_1, \ldots, y_n$. Then there exist polynomials $u, v \in F[x, y_1, \ldots, y_n]$ and $d \in F[y_1, \ldots, y_n]$ respectively, s.t. $au + vb = d$, iff a and b have no common factor of positive degree in x. Furthermore, such a d exists with degree at most $O(d^2)$.*

Discriminant

We are now back to working with polynomials in $F[x]$, for a field F.

Suppose we'd like to know if a polynomial f has a nontrivial divisor g such that g^2 also divides f. We know that over the reals, such g exists if and only if $(f, f') \neq 1$ (here f' is the derivative of f). This is because if $f = g^2h$, then $f' = 2gg'h + g^2h' = g(2g'h + gh')$. Conversely, if $(f, f') = g \neq 1$, then $f = gh$ and $f' = gk$, so $f' = g'h + h'g = gk$, so $g|g'h$, so g and h have a non-trivial common factor t (because $F[x]$ is a unique factorization domain and $\deg(g) > \deg(g')$ and $g' \neq 0$, because $g \neq 1$). Thus, $t|g$ and $t|h$, so $t^2|gh = f$.

But the derivative f' doesn't really make sense over a non-continuous domain such as a finite field or the rationals. However, it can still be syntactically defined: if $f(x) = \sum_{i=0}^{n} f_i x^i$, then $f'(x) = \sum_{i=1}^{n} i f_i x^{i-1}$ (of course, we need to define if_i, because the field F may not contain the integers; we defined it naturally as follows: for $i \in \mathbb{Z}$ and $\alpha \in F$, $i\alpha = \sum_{j=1}^{i} \alpha$). All the expected properties of derivatives can be syntactically verified: $(a+b)' = a' + b'$, $(ab)' = a'b + b'a$ and $(a^2)' = 2aa'$. It would then seem that we could make the same statement about (f, f') over arbitrary fields as we did over the reals and that the same proof as above would go through. Indeed it does, if $f' \neq 0$. The only problem is that, over fields of characteristic $p > 0$, f' can be 0 even if $\deg(f) > 0$ (we used the fact that f' cannot be 0 in the proof above). This will happen if and only if $f = \sum_{i=0}^{n} f_i x^{pi}$. To handle this case we notice that the following conditions hold over a field of size p^k: (1) $a^p + b^p = (a+b)^p$ for every $a, b \in F[x]$, and (2) $a = a^{p^k} = (a^{p^{k-1}})^p$. Thus, we get that $g = \left(\sum_{i=0}^{n} g_i x^i\right)^p = t^p$, where $t = \sum_{i=0}^{n} g_i^{p^{k-1}} x^i$. Thus $t^2 | g$, so $t^2 | f$.

This leads us to the following definition and the following (just proven) theorem.

Definition 4.5 *The discriminant of a polynomial $f \in F[x]$, denoted $\Delta(f)$ is defined to be* $\mathrm{Res}(f, f')$.

Theorem 4.6 *Let $f \in F[x]$. The discriminant of f is zero if and only if $g^2 | f$ for some $g \in F[x]$, $\deg(g) > 0$.*

As in the case of the resultant, the discriminant can also be defined over commutative rings. Over unique factorization domains, it retains properties similar to the case of discriminant over fields.

Some Asides

The following statements demonstrate more properties of the resultants.

Proposition 4.7 *If $a(x), b(x) \in F[x]$ are of the form $a(x) = \prod_i (x - \alpha_i)$ and $b(x) = \prod_j (x - \beta_j)$, then* $\mathrm{Res}(a, b) = \prod_i \prod_j (\alpha_i - \beta_j)$.

To prove the above statement, one expresses the coefficients of $a(x)$ and $b(x)$ explicitly in terms of the α_i's and β_i's. Then by inpecting the matrix $M_{a,b}$ one can see that the determinant consists only of homogenous terms in α_i's and β_j's; and thus the resultant $\mathrm{Res}(a, b)$ is a multivariate polynomial in the α's and β's of total degree mn. Furthermore, $\alpha_i - \beta_j$ must divide this polynomial, since if $\alpha_i = \beta_j$, then the resultant is zero. Thus the resultant is of the form described above.

A similar formula also exists for the discriminant of $f(x)$ if $f(x) = \prod_i (x - \alpha_i)$.

Proposition 4.8 *If $f(x) \in F[x]$ is given by $f(x) = \prod_{i=1}^{n} (x - \alpha_i)$, then $\Delta(f) = \prod_{i<j} (\alpha_i - \alpha_j)^2$.*

Finally we see a "typical" application of the resultant.

Proposition 4.9 *Let $a(x, y), b(x, y) \in F[x, y]$ be of degree at most d in x, y. Suppose for at least cd^2 choices of $\alpha \in F$, it is the case that the univariate polynomial $a_\alpha^x(y) \stackrel{\mathrm{def}}{=} a(\alpha, y)$ divides the univariate polynomial $b_\alpha^x \stackrel{\mathrm{def}}{=} b(\alpha, y)$. (Here c is some universal constant.) Suppose further that for at least cd^2 choice of $\beta \in F$ we have that $a_\beta^y(x) \stackrel{\mathrm{def}}{=} a(x, \beta)$ divides $b_\beta^y(x) \stackrel{\mathrm{def}}{=} b(x, \beta)$. Then $a(x, y)$ divides $b(x, y)$.*

To prove the above proposition, assume a does not divide b. By removing common factors, one is left with a and b that are relatively prime. Thus one can consider the resultant of a and b wr.t. y. This must be non-zero and hence it is a low-degree polynomial in x, say $r(x)$. For a specific choice of $x = \alpha$, the resultant of the polynomials a_α^x and b_α^x equals $r(\alpha)$. If α is such that a_α^x divides b_α^x, the α is a root of r. Since r can not have too many roots, the proposition would follow. (But where did we need the conditions on a_β^y dividing b_β^y?)

Factoring Polynomials over Finite Fields

Let's start by looking at a simple problem. How do we factor $x^2 - a$ over an odd prime finite field $GF(p) = \mathbb{Z}_p$, for $a \neq 0$? We know that if it does factor, it will be $(x - \alpha)(x + \alpha)$, for $\alpha^2 = a$.

We also know that, by Fermat's Little Theorem, $\beta^p = \beta$ for all $\beta \in \mathbb{Z}_p$. Therefore, every $\beta \in \mathbb{Z}_p$ is a root of $x^p - x$, therefore $(x - \beta) | (x^p - x)$. Thus, $x^p - x = \prod_{\beta \in \mathbb{Z}_p} (x - \beta)$ (there are no other factors because the right-hand side is degree p). Thus, both $(x - \alpha)$ and $(x + \alpha)$ divide $x^p - x$.

Let $f_1(x) = x^{\frac{p-1}{2}} - 1$ and $f_2(x) = x^{\frac{p-1}{2}} + 1$ Note that $x^p - x = xf_1(x)f_2(x)$. Therefore, $x - \alpha$ divides $f_1(x)$ or $f_2(x)$, and $x + \alpha$ divides $f_1(x)$ or $f_2(x)$. If they divide different ones, then we can use the GCD algorithm on $(x^2 - a)$ and $f_1(x)$ to get $(x - \alpha)$ or $(x + \alpha)$. Note, of course, that the GCD algorithm runs in time polynomial in the degrees of its inputs, so this, if done directly, would take time polynomial in p, which may be to long. Instead, you first compute $g_1(x) = f_1(x) \mod x^2 - a$ by computing $x \mod x^2 - a$, $x^2 \mod x^2 - a$, $x^4 \mod x^2 - a, \ldots$ and then multiplying the necessary ones together to get $x^{\frac{p-1}{2}} \mod x^2 - a$. Then you compute $(g_1(x), x^2 - a) = (f_1(x), x^2 - a)$.

Thus, the only remaining question is how to make it so that $(x - \alpha)|f_1(x)$ and $(x + \alpha)|f_2(x)$ (or vice versa)? Note that $(x - \alpha)|f_1(x) \Leftrightarrow \alpha^{\frac{p-1}{2}} = 1$ and $(x + \alpha)|f_1(x) \Leftrightarrow (-\alpha)^{\frac{p-1}{2}} = -1$. Since for all $\alpha \in \mathbb{Z}_p$, $\alpha^{\frac{p-1}{2}} = 1$ or -1, the condition that interests us is equivalent to $(-1)^{\frac{p-1}{2}} = -1$. This is the case if and only if $\frac{p-1}{2}$ is odd, or $p = 3 \mod 4$.

So, we now have an algorithm for factoring polynomials $x^2 - a$ over finite fields of prime size $p = 3 \mod 4$. (We could have got to this point more quickly by using $\alpha = a^{(p+1)/4}$!) In the next lecture, we will see how to generalize the above algorithm to all fields, and to factor all polynomials!

6.966 Algebra and Computation September 23, 1998

Lecture 5

Lecturer: Madhu Sudan *Scribe: Anna Lysyanskaya*

In this lecture we will see an efficient algorithm for factoring polynomials over $GF(p^k)[x]$ (p is any prime). Recall that in the last lecture we saw how to factor a polynomial of the form $x^2 + a$ in $GF[p](x)$, where $p = 3 \bmod 4$. In this lecture we will first develop an algorithm for factoring such polynomial for any odd prime p, and then generalize our algorithm to factor any polynomial in $GF(p^k)[x]$.

5.1 An Algorithm to Factor $x^2 + a$

Consider the polynomial $f(x) = x^2 + a$ in $GF(p)[x]$, where $p = 3 \bmod 4$. Since $f(x)$ is of degree 2, we know that unless it is irreducible, it has two factors of degree 1, $f_1(x) = x - \alpha$ and $f_2(x) = x + \alpha$. Note that, since these are of degree 1, they are irreducible. In Lecture 4, we have seen that if these factors exist and they are distinct, the following algorithm will always find them:

- Compute $g(x) = x^{\frac{p-1}{2}} - 1(\bmod f(x))$.
- Compute $h(x) = GCD(g(x), f(x))$.
- Output $h(x), \frac{f(x)}{h(x)}$.

The reason that this algorithm works, as was stated in lecture 4, is that exactly one of $f_1(x)$, $f_2(x)$, divides the polynomial $g(x)$ defined above. Recall the proof of this fact: Any polynomial of degree 1 will divide the polynomial $x^p - x$ by Fermat's little theorem. Then we have:

$$f_1(x)|x^p - x,$$

$$f_2(x)|x^p - x$$

It follows that

$$f_1(x)f_2(x)|x^p - x = x(x^{\frac{p-1}{2}} - 1)(x^{\frac{p-1}{2}} + 1)$$

Suppose one of the factors of $f(x)$ divides $x^{\frac{p-1}{2}} + 1$. Without loss of generality, let $f_1(x)|x^{\frac{p-1}{2}} + 1$. Then, since $f(\alpha) = 0$, $\alpha^{\frac{p-1}{2}} = -1$. On the other hand, since $p = 3 \bmod 4$ and consequently $\frac{p-1}{2}$ is odd,

$$\begin{aligned}
(-1)^{\frac{p-1}{2}} &= -1 \Rightarrow \\
(-1)^{\frac{p-1}{2}}\alpha^{p-1} &= -1 \Rightarrow \\
(-1)^{\frac{p-1}{2}}\alpha^{\frac{p-1}{2}}\alpha^{\frac{p-1}{2}} &= -1 \Rightarrow \\
(-\alpha)^{\frac{p-1}{2}}\alpha^{\frac{p-1}{2}} &= -1 \Rightarrow \\
(-\alpha)^{\frac{p-1}{2}} &= 1 \Rightarrow \\
x + \alpha|x^{\frac{p-1}{2}} - 1.
\end{aligned}$$

The last equation follows because $-\alpha$ is the root of both the polynomial on the right and the one on the left.

Thus we have showed that

$$f_1(x)|x^{\frac{p-1}{2}} + 1,$$

$$f_2(x)|x^{\frac{p-1}{2}} - 1.$$

The other direction can be showed in a similar manner.

The fact that this is true when $p = 3 \bmod 4$ looks like a happy coincidence; however it turns out that a very similar factoring algorithm will work when $p = 1 \bmod 4$. First observe that factoring $f(x) = x^2 - a$ can be reduced to factoring $f^{(1)}(x) = (x-d)^2 - c^2 a$ for any $c, d \in GF(p)$. This is true because if $f(x) = (x-\alpha_1)(x-\alpha_2)$, then $f^{(1)}(x) = (x-d)^2 - c^2 a = ((x-d) - c\alpha_1)((x-d) - c\alpha_2)$, and so if we discover that $f^{(1)}(x) = (x-\beta_1)(x-\beta_2)$, we can compute α_1 from β_1 and α_2 from β_2.

Now suppose that c, d are picked at random. What are the chances that exactly one of the factors of $f^{(1)}(x)$ divides $x^{\frac{p-1}{2}}$? It turns out that it is approximately $\frac{1}{2}$ by the following argument:

$$\forall \alpha_1 \neq \alpha_2, \beta_1, \beta_2 \Pr_{c,d,\in F}[c\alpha_1 + d = \beta_1 \text{ and } c\alpha_2 + d = \beta_2] = \frac{1}{|F|^2}$$

On the other hand,

$$\Pr_{\beta_1}[(x-\beta_1) \text{ divides } (x^{\frac{p-1}{2}} - 1)] = \frac{\frac{p-1}{2}}{p} \approx \frac{1}{2}$$

$$\Pr_{\beta_2}[(x-\beta_2) \text{ does not divide } (x^{\frac{p-1}{2}} - 1)] = \frac{\frac{p-1}{2}}{p} \approx \frac{1}{2}$$

Therefore, uniform choice of c, d guarantees that β_1, β_2 are independent of each other, and so the probability that $(x-\beta_1)$ and $(x-\beta_2)$ divide different factors of $(x^{p-1} - 1)$ is approximately $\frac{1}{2}$.

Then the algorithm proceeds as follows:

1. Compute $g(x) = x^{\frac{p-1}{2}} - 1 (\bmod f(x))$.

2. Pick random $c, d \in F$ and compute $f^{(1)}(x) = (x-d)^2 - c^2 a$.

3. Compute $h(x) = GCD(f^{(1)}(x), g(x))$.

4. If $h(x) = 1$, go back to step 2.

5. We now know that $f^{(1)}(x) = h(x)\frac{f^{(1)}(x)}{h(x)} = (x-\beta_1)(x-\beta_2)$. Compute $\alpha_i = \frac{\beta_i - d}{c}$ for $i \in \{1, 2\}$.

5.2 Extending the Algorithm

5.2.1 Challenges

Even though the algorithm above can only factor a special case of polynomials, it can be extended to an algorithm that factors polynomials in $GF(p^k)[x]$, where p is prime. Furthermore, it finds a non-trivial factorization of any polynomial of the form $(x-\alpha_1)\cdots(x-\alpha_l)$; and thus by recursing on such a factorization, one can get a complete factorization of any such polynomial.

Now let us enumerate the problems with applying the algorithm as it is to the more general case.

1. The algorithm works only for polynomials whose factorization into irreducible factors is square-free, i.e., for all $g(x)$ that divide $f(x)$, $g^2(x)$ does not divide $f(x)$.

2. Only the first degree irreducible polynomials divide $(x^q - x)$, where $q = |F|$. We have to find polynomials that will be divisible by irreducibles of higher degrees.

3. The trick of classifying factors of a polynomial into those that divide $x^{\frac{q-1}{2}} - 1$ and those that don't needs to be generalized to work for non-linear irreducible factors.

4. The algorithm is designed to work over odd prime fields.

5.2.2 Square-Free Factorization

Let us (re)introduce the notion of the derivative of a polynomial in $GF(p^k)[x]$ (see also Lecture 4).

Definition 5.1 *For a polynomial $f(x) = \sum_{i=0}^{n} c_i x^i$, the derivative $f'(x)$ is defined as*

$$f'(x) = \sum_{i=1}^{n} c_i i x^{i-1}.$$

The properties of this derivative often coincide with the properties of the more conventional derivative in the reals. In particular, the derivative of a sum is the sum of the derivatives. This property is straightforward to show. Also, if $u(x)$ and $v(x)$ are two polynomials, then the derivative of their product $(uv)'(x) = u'(x)v(x) + u(x)v'(x)$. This property can be derived by induction on the number of monomials of $u(x)$ and $v(x)$ in a fairly straightforward fashion. In dealing with eliminating repeated factors, the derivative of a polynomial is useful due to the following theorem:

Theorem 5.2 *Suppose there exist non-zero degree polynomials in $GF(p^k)[x]$, $f(x)$ and $g(x)$, and a polynomial $h(x)$ such that $f(x) = g^2(x)h(x)$. Then $GCD(f(x), f'(x)) \neq 1$.*

Proof Let us show that $g(x)$ divides $f'(x)$: $f'(x) = (g^2h)'(x) = (g^2)'(x)h(x) + g^2(x)h'(x) = 2g(x)g'(x)h(x) + g^2(x)h'(x) = g(x)(2g'(x)h(x) + g(x)h'(x))$. Clearly, since both $f(x)$ and $f'(x)$ are divisible by $g(x)$, their greatest common divisor is different from 1. ■

Since the greatest common divisor is different from 1, it is us a non-unit factor of $f(x)$. Of course, the factor is non-trivial only when $GCD(f, f') \neq f$. What happens in the case when $GCD(f, f') = f$? Observe that this can only happen when $f' = 0$, because the degree of f' is always smaller than the degree of f. What does the fact that $f' = 0$ tell us about f? Since the coefficients of f come from an integral domain, the only way that the coefficients of f' can be 0's is if all the monomials present in f are of some degree that is divisible by p. But that fact alone gives us a factorization of $f(x)$: $f(x) = \sum a_i x^{ip} = (\sum a_i^{p^{k-1}} x^i)^p$, because over $GF(p^k)$, $(a+b)^p = a^p + b^p$ (recall that the other terms cancelled because their coefficients are divisible by p, which is the characteristic of the field).

5.2.3 Distinct Degree Factorization

In order to generalize our present algorithm, we need to find an analogue of the polynomial $x^q - x$. Recall that $x^q - x$ has the property that any element of the field is its root, and therefore this polynomial is divisible by all the linear polynomials in $F[x] = GF(p^k)[x]$.

Theorem 5.3 *The polynomial $x^{q^d} - x$ is a a multiple of any degree d irreducible polynomial. Furthermore if an irreducible polynomial $f(x)$ divides $x^{q^d} - x$, then the degree of f divides d.*

Proof We only prove the first part. We postpone the proof of the "furthermore" part to the next lecture.

Let $f_1(x)$ be an irreducible polynomial of degree d. Let $K = F[x]/(f_1(x))$. K is a field that contains all polynomials in variable x of degree $\leq d-1$. Therefore, for all $\alpha \in K$, $\alpha^{|K|} - \alpha = 0$. Since the polynomial x also belongs to the field K, $x^{|K|} - x = 0 \bmod f_1(x)$, and so $f_1(x)$ divides $x^{|K|} - x$. What is left to note is that $|K| = q^d$. That is true because K consists of all polynomials in variable x of degree $\leq d-1$. Any such polynomial has d coefficients, one per degree, from F. Thus $|K| = |F|^d = q^d$. ■

Owing to this theorem, we now have an algorithm for factoring $f(x)$ into $g_1(x), ..., g_l(x)$, where $g_i(x)$ is a product of irreducible polynomials of degree i. The algorithm goes as follows:

- Set $i = 1$.
- While $f(x)$ is non-unit,
  - $g_i(x) := GCD(f(x), x^{q^i} - x)$

  - $f(x) := f(x)/g_i(x)$
- Output $g_1(x), ..., g_l(x)$.

Note that the above algorithm is also an irreducibility test: if it outputs $f(x)$ in the end, we know that $f(x)$ must be irreducible.

5.2.4 Generalizing the Trick of Factor Splitting

At this point, we may assume that we have a polynomial $f(x)$ over $F[x] = GF(p^k)[x]$ (p is an odd prime) that is a product of distinct irreducible factors of degree d. This is a situation similar to the one we have seen, where we have to find a polynomial that is a multiple of exactly at least one, but not all factors of $f(x)$. We focus on the case $f(x) = f_1(x).f_2(x)$ and show how the algorithm works in this case. (The algorithm remains the same for $f(x) = f_1(x) \cdots f_l(x)$, only it only produces a non-trivial factorization, on which we may recurse.)

We have already seen that for any $T(x)$, $g(x) = T(x)^{q^d} - T(x)$ should be a multiple of any irreducible polynomial $f_1(x)$ of degree d. Observe that

$$g(x) = T(x)(T(x)^{\frac{q^d-1}{2}} - 1)(T(x)^{\frac{q^d-1}{2}} + 1) = T(x)A_1(T(x))A_2(T(x))$$

.

Theorem 5.4 *Let $f_1(x) \in F[x] = GF(p^k)[x]$ be an irreducible polynomial of degree d and let $d' > d$. Then*

$$\Pr_{T(x)}[f_1(x)|A_1(T(x))] \approx \frac{1}{2}.$$

where $T(x)$ is chosen uniformly from the space of polynomials of degree d' over F[x].

Proof Notice first that picking $T(x)$ at random over polynomials of degree $d' > d$, corresponds to picking $T'(x) = T(x) \bmod f_1(x)$ at random from the field $K = F[x]/(f_1(x))$. Furthermore $f_1(x)|A_1(T(x))$ if and only if $f_1(x)|A_1(T'(x))$. Thus the question of whether $f_1(x)|(T(x)^{\frac{q^d-1}{2}} - 1)$ is really the same as the question of whether $T'(x)$ is a quadratic residue in the field $K = F[x]/(f_1(x))$. Since $K = GF(q^d)$, and $GF(q^d) \setminus \{0\}$ is a cyclic group of even order $q^d - 1$ under multiplication, exactly half of its elements are quadratic residues. Then there are $\frac{q^d-1}{2}$ quadratic residues modulo $f_1(x)$, and so

$$\Pr_{T(x)}[f_1(x)|A_1(T'(x))] \approx \frac{q^d - 1}{2}.$$

Notice that in the above argument we could have used any other factorization of the polynomial $x^{q^d} - x$ into $A_1(x).A_2(x)$ and the final probability depends only on the degree of A_1. ∎

Now we need to show that the event that a polynomial $T(x)$ is a quadratic resudue modulo $f_1(x)$ is independent from the event that it is a quadratic residue modulo $f_2(x)$ for a randomly chosen $T(x)$ of degree $\leq 2d - 1$.

Theorem 5.5 *For any $T_1(x), T_2(x)$ over $GF(q)[x]$ of degree $\leq d-1$, and irreducible polynomials f_1, f_2 over $GF(q)[x]$, of degree d,*

$$\Pr_{T(x)}[T(x) = T_1(x) \ mod\ f_1(x) \ and\ T(x) = T_2(x) \ mod\ f_2(x)] = \frac{1}{(q^d)^2}$$

where $T(x) \in GF(q)[x]$, of degree $\leq 2d - 1$.

Proof This result is a direct consequence of the Chinese Remainder Theorem. ∎

The previous two theorems, combined, give us the desired result that if f_1 and f_2 are two factors of f, then with probability approximately $\frac{1}{2}$ one of them divides $A_1(T(x))$ while the other does not, and so by computing $GCD(f, A_1(T(x))$, we get a factor of f with probability approximately $\frac{1}{2}$.

5.2.5 The Remaining Case: $p = 2$

The only remaining problem with the approach we have to factoring polynomials in $GF(p^k)[x]$ is that the polynomial $T(x)^{p^k d} - T(x)$ can be factored into $T(x)(T(x)^{\frac{p^{kd}-1}{2}} - 1)(T(x)^{\frac{p^{kd}-1}{2}} + 1)$ only if p is odd. What happens if $p = 2$?

Note that $T(x)^{2^{kd}} - T(x) = A(T(x))(A(T(x)) + 1)$ where $A(T(x)) = T(x)^{2^{kd-1}} + T(x)^{2^{kd-2}} + \ldots + T(x) + T(x)^0$. This fact is straightforward to verify keeping in mind that the characteristic of the field is 2.

Theorem 5.6 *For any irreducible polynomial $f_1(x) \in F[x] = GF(2^k)[x]$ of degree d,*

$$\Pr_{T(x) \in F[x]/f_1}[A(T(x)) = 0 \ mod \ f_1] \approx \frac{1}{2}.$$

Proof We first observe the following (can be verified straightforwardly):

Lemma 5.7 *If $T_1(x) \neq T_2(x) \in F[x]/f_1$ are such that $A(T_i(x)) = 0 \ mod \ f_1$ $(i \in \{1,2\})$, then $A(T_1(x) + T_2(x)) = 1 \ mod \ f_1$. On the other hand, if $T_1(x) \neq T_2(x) \in F[x]/f_1$ are such that $A(T_i(x)) = 1 \ mod \ f_1$ $(i \in \{1,2\})$, then $A(T_1(x) + T_2(x)) = 0 \ mod \ f_1$.*

Now, suppose $\Pr_{T(x) \in F[x]/f_1}[A(T(x)) = 0 \text{ mod } f_1] \geq \frac{1}{2}$. Let

$$S_1 = \{T(x) \text{ such that } A(T(x)) = 0 \text{ mod } f_1\}$$

By our assumption, S_1 must occupy at least a half of $F[x]/f_1$. Let $T_1(x) \neq 0$ be a polynomial from S_1. Define

$$S_2 = \{T(x) \text{ such that } T(x) = T_1(x) + T'(x), \text{ where } 0 \neq T'(x) \neq T_1(x) \in S_1\}$$

By the lemma, all the polynomials in $T(x) \in S_2$ are such that $A(T(x)) = 1 \text{ mod } f_1$. On the other hand, by construction, $|S_2| = |S_1| - 2$. Therefore, we get:

$$\Pr_{T(x) \in F[x]/f_1}[A(T(x)) = 0 \text{ mod } f_1] \geq \frac{1}{2} \Rightarrow \Pr_{T(x) \in F[x]/f_1}[A(T(x)) = 0 \text{ mod } f_1] \approx \frac{1}{2}$$

The other direction can be showed in exactly the same way. ■

This theorem combined with the theorem from the previous section, guarantees that the greatest common divisor of $f(x)$ and $A(T(x)))$ for a randomly chosen $T(x)$ of degree $2d - 1$, is a non-trivial factor of $f(x)$ with probability $\geq \frac{1}{2}$.

5.3 Summary

As a result, we have the following algorithm to factor any polynomial $f(x) \in F[x] = GF(p^k)[x]$[6]:

1. Use $f'(x)$ to factor out repeated factors.
2. Use distinct degree factorization to obtain $g_1(x), \ldots, g_l(x)$, where $g_i(x)$ is a multiple of distinct irreducibles of degree i.
3. Factor each $g_d(x)$ by the following procedure:
   - Pick a random $T(x)$ of degree $2d - 1$.
   - Compute $g(x) = GCD(f(x), A(T(x)))$, where $A(T(x)) = T(x)^{\frac{p^{kd}-1}{2}} - 1(\text{ mod } f(x))$ if $p \neq 2$, $A(T(x)) = T(x)^{2^{kd-1}} + T(x)^{2^{kd-2}} + \ldots + T(x) + T(x)^0(\text{ mod } f(x))$ if $p = 2$.
   - Repeat for $g(x)$, if its degree $\geq d$, and for $f(x)/g(x)$, if its degree $\geq d$.

[6] The first randomized polynomial time algorithm for factoring univaraite polynomials was given by Berlekamp ('70). The algorithm we have seen is different and is due to Cantor and Zassenhaus. The primary difference in the algorithm is the part addressed in Section 5.2.4 (i.e., how it addresses the splitting of degree d factors). See Knuth (v. 2)/Cohen for more details.

6.966 Algebra and Computation September 28, 1998

Lecture 6

Lecturer: Madhu Sudan *Scribe: Zulfikar Ramzan*

6.1 Introduction

In this lecture we covered Berkelamp's Algorithm for factoring polynomials and we also dicussed some properties of irreducible polynomials over finite fields. As an aside, during the last lecture, the algorithm covered was due to Cantor and Zassenhaus. Knuth Volume 2 and the text by Cohen are good references.

6.2 Berkelamp's Factoring Algorithm

In this section we describe Berkelamp's Factoring Algorithm. It is one of the older algorithms for factoring polynomials, and it's very fundamental. In this lecture, we cover the deterministic variant of the algorithm, which was developed in 1969. This variant has running time polynomial in the degree of the polynomial and in the field size. A randomized version of the algorithm was developed in 1970, but we won't be discussing it during this lecture.

6.2.1 Main Idea

Say we are given a polynomial $f(x)$, and we need to find $f_1(x), \ldots, f_k(x)$ such that $f(x) = f_1(x)f_2(x)\cdots f_k(x)$. As in Lecture 5, we first reduce to the case where all the f_i's are irreducible, distinct, and of degree d. We work over the field $F = GF(q)$ where $q = p^m$ for some prime p. For conceptual simplicity, we restrict ourselves to the case $k = 2$.

The key idea of the algorithm is to find a polynomial $g \in F[x]$ such that $g(x)^q = g(x) \pmod{f(x)}$, with $0 < \deg(g) < \deg(g)$. This polynomial will turn out to have useful properties which enable us to find non-trivial factors of $f(x)$. We must answer the following questions:

1. Why is finding such a polynomial $g(x)$ useful?
2. Does such a polynomial always exist?
3. If it does exist, how do we find it?

6.2.2 Usefulness of Finding Such a Polynomial

Consider the polynomial $z^q - z$. Observe that:

$$z^q - z = \prod_{\alpha \in F} (z - \alpha)$$

Similiarly:

$$g(x)^q - g(x) = \prod_{\alpha \in F} (g(x) - \alpha)$$

Since $g(x)^q = g(x) \bmod f(x)$, it follows that $f(x) | g(x)^q - g(x)$. Now if $0 < \deg(g) < \deg(f)$ we can exploit our knowledge of the factorization of $g(x)^q - g(x)$ to help factor f. In particular we observe there must be an $\alpha \in F$ such that $gcd(f(x), g(x) - \alpha) \neq 1$. And this greatest common divisor will give you a non trivial factor of $f(x)$. Notice that it was important for us to impose the restriction that $0 < \deg(g) < \deg(f)$. For example, if g was a constant, then $g^q - g = 0$ which is not too useful. Also, if $g(x) = f(x)$ you gain nothing useful.

Now, if we run through all $\alpha \in F$, and take gcd's, we can obtain a non-trivial factorization of $f(x)$. All that remains is to show that the polynomial $g(x)$ exists and we can find it efficiently.

6.2.3 Showing Existence of Such a Polynomial

To show existence of a polynomial $g(x)$ satisfying the previously mentoined constraints, we make use of the Chinese Remainder Theorem for polynomials. Consider the polynomial $g(x) \bmod f(x)$. We can view this polynomial in terms of the factors $f_1(x), f_2(x)$ of $f(x)$. In particular, we have $(g(x) \bmod f_1(x))$ and $(g(x) \bmod f_2(x))$. And it follows that $g(x)^q = g(x) \bmod f_1(x)$ and $g(x)^q = g(x) \bmod f_2(x)$

So, one approach to constructing $g(x)$ might be to set it equal to constants modulo $f_1(x)$ and $f_2(x)$. In particular, for two constants a, b we let $g(x)$ be the polynomial which satisfies: $g(x) = a \bmod f_1(x)$ and $g(x) = b \bmod f_2(x)$. After fixing a, b we can construct $g(x)$ using the Chinese remainder theorem in the usual way and since a, b are constants, we have $g(x)^q = a^q = a = g(x) \bmod f_1(x)$. Similarly we find $g()^q = g(x) \bmod f_2(x)$ and thus $g(x)^q = g(x) \bmod f(x)$.

Of course this is not useful if g turns out to be a constant, but it is easy to see that there are non-constant polynomials g satisfying this property. To see this, consider all possible pairs $a, b \in F$; Using all such pairs, we can construct q^2 distinct polynomials g, satisfying $g(x)^q = g(x) \bmod f(x)$, all of which have degree less than f. Of these polynomials, only q of them turn out to be a constant polynomial (and this happens whenever $a = b$). So, there are $q^2 - q$ polynomials which have degree greater than 0, but still have a smaller degree than $f(x)$. We can take $g(x)$ to be any one of these polynomials. So, we've now shown that there is a polynomial $g(x)$ satisfying the constraints specified earlier. Unfortunately, we can't use this Chinese Remainder Technique for actually constructing g. Why not? Because in order to construct g via the Chinese Remainder Theorem, we would already need to know $f_1(x)$ and $f_2(x)$ – and these are the polynomials that we've been trying to find in the first place!

6.2.4 Constructing Such a Polynomial

We've shown that there exists a polynomial g which satisfies $g(x)^q = g(x) \bmod f(x)$ with $0 < \deg(g) < \deg(f)$. Now we show how to construct such a polynomial efficiently. We make the following observation:

Observation 6.1 *The space of all polynomials $g(x)$ satisfying $g(x)^q = g(x) \bmod f(x)$ is a linear space. In particular, if $g_1(x)^q = g_1(x) \bmod f(x)$ and $g_2(x)^q = g_2(x) \bmod f(x)$ then $(g_1(x) + g_2(x))^q = (g_1(x) + g_2(x)) \bmod f(x)$.*

Proof
$(g_1(x) + g_2(x))^q \bmod f(x)$
$= g_1(x)^q + g_2(x)^q + \text{multiples of p}$
$= g_1(x) + g_2(x) \bmod f(x).$ ■

Since we are looking at a linear space, we can find a basis for the set of all possible g's that work. In particular, suppose $g(x) = g_0 + g_1 x + \ldots + g_l x^l$. First we observe that $g(x)^q = g(x^q)$ and $g(x)^q = (\sum_i g_i x^i)^q = (\sum_i g_i^q x^{iq}) +$ multiples of p $= \sum g_i x^{iq} = g(x^q)$. Now, let's look at $x^{iq} \bmod f$. We can find values α_{ij} such that: $x^{iq} \bmod f = \sum_j \alpha_{ij} x^j (\text{mod} f)$. So if we have $g(x)^q = g(x) \bmod f$ then:

Since $g(x) = g(x)^q = g(x^q)$, it follows that

$$\sum_{i=0}^{l} g_i \sum_{j=0}^{l} \alpha_{ij} x^j = \sum_{i=0}^{l} g_i x^i$$

And we can find the basis for the set of all solutions.

6.2.5 Summary of Berkelamp's Factoring Algorithm

The following summarizes the steps needed in order to obtain a non-trivial factor of a polynomial $f(x)$ of degree d over the field $F = GF(q)$ in time $poly(d, q)$ (where $q = p^k$ for some prime p).

Input: A polynomial $f(x)$ defined over the field $F = GF(q)$.
Output: A non trivial factor $h(x)$

1. Obtain a polynomial $g(x)$ satisfying:
   (a) $0 < \deg(g(x)) < \deg(f(x))$
   (b) $g(x)^q = g(x) \bmod f(x)$
2. **for** all elements $\alpha \in F$ **do**:
   (a) let $h(x) \leftarrow gcd(f(x), g(x) - \alpha)$
   (b) **if** $h(x) \neq 1$ **then** output $h(x)$ and terminate.

6.3 Properties of Irreducible Polynomials

We now shift gears and talk about various properties of irreducible polynomials. Specifically, our goal is to demonstrate that there are many irreducible polynomials of degree d over $GF(q)$. We are also going to find out which irreducible polynomials divide $x^{q^d} - x$, since we needed this for the factoring algorithms. In particular, we show that:

1. For every prime $q = p^k$ and every integer $d > 0$, there exists an irreducible polynomial of degree d over $F = GF(q)$.
2. The probability that a monic polynomial over $GF(q)$ is irreducible is approximately equal to $1/d$.

In order to show these results, we make use of two algebraic concepts. The first is the concept of a Field Extension as a Vector Space, and the second is the Splitting Field of a Polynomial.

6.3.1 Field Extensions as Vector Spaces

Consider a Field K with a subfield F. Then K can be viewed as a vector space over F. In particular, there are elements $\alpha_1 \ldots \alpha_n \in K$ such that $\sum \lambda_i \alpha_i \neq 0$ for all $\lambda_1, \ldots, \lambda_n \in F$ with $(\lambda_1, \ldots, \lambda_n) \neq (0, \ldots, 0)$. You can think of $\lambda_1, \ldots, \lambda_n$ as a vector in some n-dimensional space over F (with the usual properties); namely, every element in K can be expressed as a linear combination of elements of $\alpha_1, \ldots, \alpha_n$ over F. Moreover, we can define the dimension of K over F (denoted $[K : F]$) as n – where n is the maximum number satisfying the above contraint that $\sum \lambda_i \alpha_i \neq 0$ for all $\lambda_1, \ldots, \lambda_n \in F$ with $(\lambda_1, \ldots, \lambda_n) \neq (0, \ldots, 0)$.

Proposition 6.2 *If $[K : F] = n$ then $|K| = |F|^n$.*

Proof Suppose there exist $\lambda_1, \ldots, \lambda_n$ and $\lambda'_1, \ldots, \lambda'_n$ such that $\sum \lambda_i \alpha_i = \sum \lambda'_i \alpha_i$ where $(\alpha_1, \ldots, \alpha_n) \neq (0, \ldots, 0)$ then $\sum (\lambda_i - \lambda'_i) \alpha_i = 0$ which implies that $\lambda_i = \lambda'_i$. Since there are $|F|^n$ choices of $\lambda_1, \ldots, \lambda_n$ all of which yeild distinct elements under a linear combination with $\alpha_1, \ldots, \alpha_n$, it follows that there are $|F|^n$ elements in K. ■

Proposition 6.3 *If K, L, F are fields such that $K \supseteq L \supseteq F$, then $[K : F] = [K : L] \cdot [L : F]$.*

Proof Suppose $\alpha_1, \ldots \alpha_n$ are a generating set for the extension of K over L, and suppose $\beta_1, \ldots, \beta_m$ are a generating set for the extension of L over F. Then, $\alpha_i \beta_j$ form a generating set for K over F, and they are linearly independent – which proves the proposition. ■

Exercise 6.4 *As an exercise, you can try: Given $Ax = b$ with A, b from $GF(p^k)$, find a vector $x \in GF(p)^n$ such that $Ax = b$ if such a vector exists.*

6.3.2 Splitting Field of a Polynomial

In this section we define the notion of a *Splitting Field* and prove the existence of a splitting field for any polynomial over a field.

Definition 6.5 *Given a polynomial $f(x) \in F[x]$, a field extension K of F (i.e. $K \supseteq F$ and K is itself a field) is said to be a splitting field if $f(x)$ splits into linear factors over K (but not over any proper subfield of K).*

Lemma 6.6 *Every polynomial $f(x) \in F[x]$ has a splitting field.*

Proof Consider an irreducible factor $f_1(x)$ of $f(x)$ where $f_1(x)$ has degree greater than 1 (if none exist, then all factors are linear and we are trivially done). Let $K_1 = F[u]/(f_1(u))$. Then there is an element $u \in K_1$ such that $f_1(u) = 0$ (when we look at $f_1(x) \in K_1[x]$). Therefore, $(x-u)|f_1(x)$ which implies that $(x-u)|f(x)$ (again, we are looking at these polynomials over the field K_1). Now, let $f_2(x) = f(x)/f_1(x)$. Inductively apply the above construction to $f_2(x)$ and K_1. This yields a splitting field. ■

6.3.3 Existence of Extension Fields of size q^d extending $GF(q)$

We now prove the following theorem regarding the existence of extension fields of size q^d which extend $GF(q)$:

Theorem 6.7 *For all $q = p^k$ where p is a prime, and for all d, there exists a field of size q^d extending $GF(q)$.*

Proof Consider the splitting field of $x^{q^d} - x$ over $GF(q)$. Call this splitting field K. Now, consider the set $S = \{\alpha \in K \text{ such that } \alpha^{q^d} - \alpha = 0\}$. We claim S satisfies the conditions of the theorem. We need to show that:

1. S is a field.
2. $|S| = q^d$

To see that S is a field, it suffices to check that for all $\alpha, \beta \in S$, $\alpha+\beta, \alpha\cdot\beta, -\alpha, \alpha^{-1} \in S$ – which is fairly straightforward:

1. $(\alpha+\beta)^{q^d} = \alpha^{q^d} + \beta^{q^d} +$ multiples of p $= \alpha^{q^d} + \beta^{q^d} = \alpha + \beta$. Therefore, $\alpha + \beta \in S$.
2. $(\alpha\cdot\beta)^{q^d} = \alpha^{q^d}\cdot\beta^{q^d} = \alpha\cdot\beta$. Therefore, $\alpha\cdot\beta \in S$.
3. $(-\alpha)^{q^d} = -1^{q^d}\cdot\alpha^{q^d} = -1^{q^d}\cdot\alpha = -\alpha$. The last equality follows from the previous one because q^d is odd whenever q is odd, and q is even only if the underlying prime p is 2, in which case, $-1 = 1$ for that field. Therefore, $-\alpha \in S$.
4. $(\alpha^{-1})^{q^d} = (\alpha^{q^d})^{-1} = \alpha^{-1}$. Therefore, $\alpha^{-1} \in S$.

To compute the cardinality of S observe that S cannot have more than q^d elements since $x^{q^d} - x$ has at most q^d roots (since its degree is q^d). Moroever, if you compute $gcd(f, f')$ we see that there are no multiple roots (hence S has at least q^d roots). Specifically, $gcd(f, f') = gcd(x^{q^d} - x, -1) = 1$. We know from a previous lecture that if any polynomial and it's derivative have a greatest common divisor of 1, then the polynomial does not have multiple roots. So, we have shown that S has exactly q^d elements. ■

6.3.4 Number of Monic Irreducible Polynomials of Degree d

We now show some theorems regarding the number of irreducible polynomials over a field F:

Theorem 6.8 *There are at least $(q^d - 1)/d$ monic irreducible polynomials of degree at most d over $GF(q)$.*

Proof Consider the unique extension field K of F where $|K| = q^d$. Now, for all non zero elements $\alpha \in K$ we form the sequence $\alpha^0, \alpha^1 \ldots, \alpha^l$ where l is chosen to be smallest element such that there exist $\lambda_0, \lambda_1, \ldots, \lambda_l \in F$ with $\sum \lambda_i \alpha_i = 0$ and $(\lambda_0, \lambda_1, \ldots \lambda_l) \neq (0, 0, \ldots, 0)$. We know that $l \leq d$ because K has dimension d. Now, we can restrict $\lambda_l = 1$ because the last term is always non-zero, and we can divide out. Therefore, $f(x) = \sum_{i=0}^{l} \lambda_i x^i$ has degree at most d. Moreoever, $f(\alpha) = 0$. We claim that $f(x)$ is irreducible over $F[x]$. This follows from the fact that if $f(x)$ splits, then we can find a smaller degree polynomial satisfying the above – and this contradicts the minimality of l. We have thus shown that for all $\alpha \in K$ there is a corresponding irreducible monic polynomial of degree d. But some of these polynomials may be the same – fortunately, we can account for all the repetitions by noting that each polynomial of degree d has at most d roots – hence a polynomial can repeat at most d times. So, by going through all values of α we can construct $(|K| - 1)/d$ monic irreducible polynomials. Since $|K| = q^d$, we have shown that there are at least $(q^d - 1)/d$ distinct monic irreducible polynomials of degree at most d. ■

6.3.5 Properties of $x^{q^d} - x$

We state and prove two lemmas about the polynomial $x^{q^d} - x$.

Lemma 6.9 *Let $f(x)$ be an irreducible polynomial. Then $f(x) | x^{q^d} - x$ if and only if* $\deg(f)|d$.

Proof Consider the field K that's the extension of F containing all and only the roots of $x^{q^d} - x$. Since $f(x)|x^{q^d} - x$ there must be an element $\alpha \in K$ such that $f(\alpha) = 0$. Now let's consider the field $F(\alpha) = \{\sum_{i=0}^{\deg(f)-1} \lambda_i \alpha^i \mid \lambda_i \in F\}$. Now, $K \supseteq F(\alpha) \supseteq F$ which implies that $[K : F(\alpha)][F(\alpha) : F] = [K : F]$. Now, $F(\alpha)$ has $q^{\deg(f)}$ elements – hence $[F(\alpha) : F] = \deg(f)$; $[K : F] = d$ therefore $\deg(f)|d$. ■

The following lemma and its consequence are from Lidl and Niedereitter's book on Finite fields.

Lemma 6.10 $x^{q^d} - x =$ *the product of all monic, irreducible polynomials f where* $\deg(f)|d$.

Proof This follows immediately from the fact that $x^{q^d} - x$ has no repeated factors and from the previous lemma. ■

The lemma above leads to a tight estimate on the number of irreducible polynomials of degree d. Let $N(i)$ denote the number of monic irreducible polynomials of degree i. Then, by considering the degrees of the LHS and the RHS above, we have

$$q^d = \sum_{i|d} iN(i).$$

By Moebius inversion, the above equation implies:

$$N(d) = (1/d)q^d \sum_{i|d} \mu(d/i),$$

where

$$\mu(k) = \begin{cases} 1 & \text{if } k = 1 \\ (-1)^i & \text{if } k \text{ is a product of } i \text{ distinct primes.} \\ 0 & \text{if } p^2|k \text{ for some prime } p. \end{cases}$$

Using the crude estimates: $\mu(1) = 1$ and $\mu(k) \geq -1$ we get $N(d) \geq \frac{1}{d}\left(q^d - \frac{q^d - q}{q-1}\right)$. In particular, $N(d)$ is at least 1 for every d.

6.966 Algebra and Computation	September 30, 1998
Lecture 7	
Lecturer: Madhu Sudan	*Scribe: Daniele Micciancio*

7.1 Factoring Bivariate Polynomial: Introduction

In this lecture we start the study of bivariate polynomial factorization. We already saw how to factor polynomials in a single variable. Factoring bivariate polynomials is the first step towards our goal of understanding general multivariate factorization. In the following lectures we will see that the steps to go from univariate polynomials to bivariate polynomials are similar (although not exactly the same) to those to go from bivariate to multivariate factorization.

Recall we proved that if R is a unique factorization domain (u.f.d.) then also $R[x]$ is a u.f.d. Similarly, if $R[x]$ is a u.f.d. then also $R[x,y]$ is a u.f.d. By induction, one can show that for any number of variables, if R is a u.f.d., then also $R[x_1,\ldots,x_n]$ is a u.f.d.

One could hope to make the step from R to $R[x]$ algorithmic, i.e., given an algorithm to factor in R, devise a factoring algorithm for $R[x]$. In other words, we want to answer the question: assuming we can factor elements of R, can we factor elements of $R[x]$? We don't know how to answer this for a general R, but we show that when R is a polynomial ring, then factoring in $R[x]$ can be "black-box" reduced to factoring in R. In particular we show that for any u.f.d. R, if we can factor in $R[x]$ (univariate polynomials) than we can factor in $R[x,y]$ (bivariate polynomials).

The idea to go from two variables to one variable is the following. Assume polynomial $f(x,y) \in \mathbb{Z}[x,y]$ factors as

$$f(x,y) = f_1(x,y)f_2(x,y)\cdots, f_n(x,y).$$

Then, the univariate polynomials $f(\cdot,0)$, $f(\cdot,1)$, ... factor as

$$f(x,0) = f_1(x,0)f_2(x,0)\cdots, f_n(x,0).$$

$$f(x,1) = f_1(x,1)f_2(x,1)\cdots, f_n(x,1).$$

$$f(x,2) = f_1(x,2)f_2(x,2)\cdots, f_n(x,2).$$

We can factor $f(\cdot,0)$, $f(\cdot,1)$, ..., using a univariate polynomial factorization algorithm, and try use the collected information to reconstruct a factorization of $f(x,y)$.

This is a very simplified view of the bivariate factoring algorithm, as shown by the following example. Consider the polynomial

$$f(x,y) = x(x+1) + y^2.$$

This is an irreducible polynomial, i.e., it does not factor. However the univariate polynomial

$$f(x,0) = x(x+1)$$

is reducible. So, the factorization of $f(x,0)$ does not immediately gives a factorization of $f(x,y)$, still it gives useful information. In the following we will introduce a technique called Hensel lifting that will tell us how to extract this information. Notice that factoring univariate polynomial $f(x,0) = g(x,0)h(x,0)$ is equivalent to finding a factorization

$$f(x,y) = g(x,y)h(x,y) \pmod{y}.$$

Hensel's lifting allows us to "lift" factorization modulo (y^k) to factorizations modulo (y^{2k}). This lifting is the central idea in the bivariate factorization algorithms, which we outlined below:

1. Initial preprocessing.
2. (a) Factor $f(x,y) = g(x,y)h(x,y) \pmod{y}$ using a univariate factorization algorithm.

(b) Iteratively "lift" the factorization to obtain

$$f(x,y) = g_k(x,y)h_k(x,y) \pmod{y^k}.$$

3. Cleaning up: Given factors of f modulo y^k, find a non-trivial factorization $f = g.h$

7.2 Hensel's Lifting Lemma

Hensel's Lemma tell us how to lift a factorization modulo y, to obtain a factorization modulo y^k for any k. For example, given polynomial $f(x,y) = x(x+1) + y^2$ we obtain factorizations

$$\begin{array}{lcl} f &=& x(x+1) \pmod{y} \\ f &=& x(x+1) \pmod{y^2} \\ f &=& (x+y^2)(x+1-y^2) \pmod{y^4} \\ \vdots && \end{array}$$

One may think of this process as finding approximate roots of polynomial $f_y(x) = f(x,y)$, when y is "small". If $y \approx 0$, than $f_y(-1) \approx 0$ is an approximate solution and $x+1$ is a factor modulo y. A better approximation is $-1+y^2$ which gives the factor $x+1-y^2$ modulo y^4. This is reminiscent of methods from numerical analysis which iteratively try to get closer and closer approximations to a solution for a given equation. Indeed Hensel's lifting is based on such intuition.

Formally, Hensel's Lifting Lemma tell us how to transform a factorization modulo y^k into a factorization modulo y^{2k}. The lemma works for any ideal and can be stated in a very general form. Recall that an ideal is a subset $I \subset R$ of a ring such that

$$\forall x,y \in I,\ \forall \gamma \in R,\ (x+y) \in I`mbox\ and\ \gamma x \in I.$$

Example of ideals are:

- Integral multiples of an integer m: $\mathbb{Z}_m = \{mn : n \in \mathbb{Z}\}$.
- Multiples of a polynomial p.
- The set of bivariate polynomials in x, y with monomials $x^i y^j$ such that $i + j \geq d$.

Ideals are important because we can define the quotient R/I, where $p \equiv q \pmod{I}$ if $p - q \in I$, and operations on R/I are well defined. The *product* of two ideals I, J is the set of all finite sums of products of elements of I and J:

$$I \cdot J = \{\sum_{i=1}^{n} a_i b_i : a_i \in I, b_i \in J\}.$$

For example, if, for some $p \in R$, I_p is the ideal $\{pq : q \in R\}$ (denoted (p)) then $I^2 = I.I = (p^2)$.

For u.f.d. $f, g, \in R$ and ideal I in R, we say that $\mathrm{pseudo}-\gcd(f,g) = 1 \pmod{I}$ to imply that there exist $a, binR$ such that $af + bg = 1 \pmod{I}$. Notice that since R/I is not necessarily a u.f.d., it does not make sense to talk of a real gcd in R/I.

Lemma 7.1 (Hensel's Lifting) *Let R be a ring and $I \subset R$ be an ideal. For any $f \in R$ and for any factorization $f = gh \pmod{I}$ of f in R/I such that* $\mathrm{pseudo}-\gcd(g,h) = 1 \pmod{I}$ *there exist g^* and h^* such that*

$$\begin{cases} f \equiv g^* h^* \pmod{I^2} \\ g^* \equiv g \pmod{I} \\ h^* \equiv h \pmod{I}. \end{cases} \tag{7.1}$$

Moreover, the following holds:

- *for any solution g^*, h^* to (7.1),* $\mathrm{pseudo-gcd}(g^*, h^*) = 1 \pmod{I^2}$ *(ie. $a^*g^* + b^*h^* \equiv 1 \pmod{I^2}$ for some a^*, b^*).*
- *given a, b, g, h, one can easily compute a^*, b^*, g^*, h^*.*
- *The solution g^*, h^* is unique in the following sense. Any other pair g', h' is a solution to (7.1) iff*

$$\begin{aligned} g' &\equiv g^*(1+u) \pmod{I^2} \\ h' &\equiv h^*(1-u) \pmod{I^2} \end{aligned}$$

  for some $u \in I$.

Proof We know that $m \stackrel{\text{def}}{=} (f - gh) \in I$. Let $a, b \in R$ be such that $af + bg = 1 \pmod I$. Let

$$\begin{aligned} g^* &\stackrel{\text{def}}{=} g + bm \text{ and} \\ h^* &\stackrel{\text{def}}{=} h + am. \end{aligned}$$

Then g^* and h^* are congruent to g and h modulo I, respectively. Let's check that $f = g^*h^* \pmod{I^2}$:

$$\begin{aligned} f - g^*h^* &= f - (g+bm)(h+am) \\ &= f - gh - m(ag+bh) - abm^2 \\ &= m(1-(ag+bh)) + abm^2 \\ &\equiv 0 \pmod{I^2} \end{aligned}$$

because $m \in I$ and $ag + bh \equiv 1 \pmod I$. This proves that system (7.1) always has solution.

Now, let g^*, h^* be any solution and let's prove that $\mathrm{pseudo-gcd}(g^*, h^*) = 1$. Let $q = ag^* + bh^* - 1$. Notice that $ag^* + bh^* \equiv_I ag + bh \equiv_I 1$, and therefore $q \in I$. Define

$$\begin{aligned} a^* &= a - aq \\ b^* &= b - bq \end{aligned}$$

and let's check that $a^*g^* + b^*g^* = 1 \pmod{I^2}$:

$$\begin{aligned} a^*g^* + b^*g^* - 1 &= ag^* + bg^* - 1 - q(ag^* + bg^*) \\ &= q(1 - (ag^* + bg^*)) \\ &= -q^2 \in I^2. \end{aligned}$$

It remains to prove the "uniqueness" of the solution. Clearly, for any $g' \equiv_{I^2} g^*(1+u)$, $h' \equiv_{I^2} h^*(1+u)$ (for $u \in I$), we have $g' \equiv_I g^* \equiv_I g$, $h' \equiv_I h^* \equiv_I h$ and $g'h' \equiv_{I^2} g^*h^*(1-u^2) \equiv_{I^2} f$. Now assume g', h' is a solution to (7.1). Let $g_1 = g' - g^*$ and $h_1 = h' - h^*$. Since $g' \equiv_I g \equiv_I g^*$ and $h' \equiv_I h \equiv_I h^*$ we have $g_1, h_1 \in I$. Define

$$u = a^*g_1 - b^*h_1.$$

Since $g_1, h_1 \in I$ we also have $u \in I$. We claim that $g' \equiv_{I^2} g^*(1+u)$ and $h' \equiv_{I^2} h^*(1-u)$. For g' we have

$$\begin{aligned} g^*(1+u) &\equiv g^*(1 + a^*g_1 - b^*h_1) \\ &= g^* + a^*g^*g_1 - b^*g^*h_1 \\ &\equiv g^* + (1 - b^*h^*)g_1 - b^*g^*h_1 \\ &= g^* + g_1 - b^*(h^*g_1 + g^*h_1) \\ &\equiv g' - b^*(h^*g_1 + g'h_1) \\ &= g' - b^*(h^*g' - h^*g^* + g'h' - g'h^*) \\ &\equiv g' - b^*(f - f) = g'. \end{aligned}$$

The proof for h' is analogous. ■

When $R = F[x,y]$ is the bivariate polynomial ring and $I = (y^k)$ the "uniqueness" of the solution can be stated in a more expressive way.

Corollary 7.2 *Let $R = F[x,y]$ and $k \in \mathbb{Z}^+$. Let $f, g, h \in R$, with g monic in x satisfying $f \equiv_{(y^k)} gh$ and* $\text{pseudo}-\gcd(g,h) = 1 \pmod{y^k}$*. Then there exists a unique $g^* \pmod{y^{2k}}$ monic in x such that $g^* \equiv g \pmod{y^k}$ and g^* divides f modulo y^{2k}.*

Proof By Lemma 7.1 there exists $g^* \equiv g \pmod{I}$ and $h^* \equiv h \pmod{I}$ such that $f \equiv g^*h^* \pmod{I^2}$. We know $g^* - g$ is a multiple of y^k and g is monic in x. Therefore we can apply the division algorithm to divide the polynomial $a \stackrel{\text{def}}{=} (g^* - g)/y^k$ by g and obtain polynomials q and r such that

$$a = qg + r$$

where $deg_x r < deg_x g$. Let $g' = g + y^k r$. Clearly, g' is monic and $deg_x g' = deg_x g$. Moreover, $g' = g^*(1 - y^k qg^*) + y^k q(g^* - g) \equiv_{I^2} g^*(1+u)$ where $u = -y^k qg^* \in I$. It follows from lemma 7.1 that $f \equiv g'h' \pmod{I^2}$ for $h' = h^*(1-u)$, ie. g' divides f modulo I^2.

This proves that there exists a solution g' such that g' is monic and of degree $\deg_x g$. It remains to prove that the solution is unique. Let $g'' \equiv_I g$ be such that $f = g''h'' \pmod{I^2}$. We have $g''(h'' - h) \equiv_I g''h'' - gh \equiv_I f - f = 0$, ie. $y^k | g''(h'' - h)$. Since g'' is monic in x, this is possible iff $y^k | (h'' - h)$, ie. $h'' \equiv_I h$. We can now apply the uniqueness part of lemma 7.1 and get $g'' = g'(1+u)$ for some u. Since g'' are both monic of degree $deg_x g$, it must be $g'' = g'$. ■

7.3 The factoring algorithm

We now give further details of the three steps of the bivariate polynomial factoring algorithm outlined in Section 7.1.

Let f be the given bivariate polynomial. Our basic idea is to use the univariate factoring algorithm to find a factorization $f(x,y) = g_0(x,y)h_0(x,y) \pmod{y}$. The goal of Step 2 is to lift this factorization to one modulo y^{2^k}. For this we will assume that g_0 is monic and relatively prime to h_0, so as to be able to use Lemma 7.1. For what follows it will also be useful to know that g_0 is irreducible, so we assume that too. In the next lecture we will show how to get rid of these assumptions by some initial preprocessing in Step 1. (Notice that either of the assumptions (1) g_0 is relatively prime to h_0 or (2) g_0 is irreducible is trivial to achieve; but it is non-trivial to get the two properties together.) Having lifted the factorization for sufficiently large value of k, the main sub-steps of Step 3 of the algorithm are the following:

Step 3(a). Find $\tilde{g}$ and l_k such that

$$\left.\begin{array}{l} \tilde{g} = g_k l_k \pmod{y^{2^k}} \\ \deg_x \tilde{g} < \deg_x f \\ \deg_y \tilde{g} \leq \deg_y f \\ \tilde{g} \neq 0 \end{array}\right\} \qquad (7.2)$$

Step 3(b). Find $\gcd_x(f, g)$ (viewed as polynomials in $F(y)[x]$) and if non-trivial, a non-trivial factorization of f in $F[x,y]$ may be found using Gauss's Lemma.

The following questions arise:

- How do we know $\tilde{g}$ and l_k satisfying (7.2) exist?
- How do we find such a pair $\tilde{g}$ and l_k?
- Why does the existence of $\tilde{g}$ and l_k yield anything in Step 3(b)? (The answer to this will also answer the question: "How large should k be?")

We answer these questions in turn. To do so we have to study the properties of the polynomials g_0 and g_k (relative to f). Notice that g_0 does not necessarily correspond to a factor of f. However, g_0 does *divide* (modulo y) an irreducible factor of f. In other words, there exist polynomials $g(x,y), h(x,y)$ such that $f(x,y) = g(x,y)h(x,y)$, $g(x,y)$ is irreducible and $g(x,y) = g_0(x,y)l_0(x,y) \pmod y$ and $h_0(x,y) = l_0(x,y)h(x,y) \pmod y$. We claim that g forms part of a candidate solution to (7.2).

Claim 7.3 *If $f(x,y) = g(x,y)h(x,y)$ for some g and h that are relatively prime; and $g(x,y) = g_0(x,y)l_0(x,y) \pmod y$ for some polynomial l_0, then there exists a polynomial l_k such that $\tilde{g} = g$ and l_k form a solution to (7.2).*

Proof Notice first g satisfies the degree conditions and the condition $g \neq 0$. It suffice to show that there exists l_k such that $g(x,y) = g_k(x,y)l_k(x,y) \pmod{y^{2^k}}$.

We find l_k by applying the Hensel lifting procedure to the factorization of g modulo y. This yields polynomials $\tilde{g}_k$ and l_k such that $g(x,y) = \tilde{g}_k(x,y).l_k(x,y) \pmod{y^{2^k}}$, where $\tilde{g}_k$ is monic in x, and satisfies $\tilde{g}_k = g_0 \pmod y$. We claim $\tilde{g}_k = g_k$. To do so we set $\tilde{h}_k(x,y) = l_k(x,y)h(x,y)$ and then notice that

$$f(x,y) = g(x,y)h(x,y) = \tilde{g}_k(x,y)l_k(x,y)h(x,y) \pmod{y^{2^k}} = \tilde{g}_k(x,y)\tilde{h}_k(x,y) \pmod{y^{2^k}}.$$

Additionally $\tilde{g}_k$ is monic and equals $g_0 \pmod y$. Thus $\tilde{g}_k$ and $\tilde{h}_k$ satisfy all the properties of the solution to the Hensel lifting procedure after k iterations. But the solution to the Hensel lifting is unique, implying $g_k = \tilde{g}_k$. This yields the claim. ■

Now that we know a solution exists, we can easily find polynomials $\tilde{g}$ and l_k of the required degree just by solving a system of linear equations, where the unknown are the coefficients of $\tilde{g}$ and l_k. This answers the second question above.

Finally, we need to see why computing the greatest common divisor of f and $\tilde{g}$, as polynomials in $F[y](x)$, yields a non-trivial factorization.

Claim 7.4 *Let g, l_k be any pair of polynomials satisfying (7.2). Let $2^k > 2d^2$ where d is the (total) degree of f. Then $\gcd_x(f,g)$ is non-trivial.*

Proof Obviously $\gcd(f,\tilde{g}) \neq f$ because $\deg_x g < \deg_x f$. Assume for contradiction that $\gcd_x(f,\tilde{g}) = 1$. Then there exist polynomials $u(x,y)$ and $v(x,y)$ such that

$$u(x,y)f(x,y) + v(x,y)g(x,y) = \mathrm{Res}_x(f,g)$$

where $\mathrm{Res}_x(f,g)$ is the resultant of f and g w.r.t. x. Recall that the resultant is a non-zero polynomial in y of degree at most $2d^2$ (see Lecture 4). Call this polynomial $D(y)$. Using the assumption that $2^k > 2d^2$, we get that $D(y) \neq 0 \pmod{y^{2^k}}$. Thus reducing the equation above modulo y^{2^k} we get

$$\begin{aligned}
D(y) \pmod{y^{2^k}} &= u(x,y)f(x,y) + v(x,y)\tilde{g}(x,y) \pmod{y^{2^k}} \\
&= u(x,y)g_k(x,y)h_k(x,y) + v(x,y)g_k(x,y)l_k(x,y) \pmod{y^{2^k}} \\
&= g_k(x,y)(u(x,y)h_k(x,y) + v(x,y)l_k(x,y) \pmod{y^{2^k}}),
\end{aligned}$$

But the above is contradicting. Oh the LHS we have a non-zero element of $F[x,y]$ with degree 0 in x. On the RHS we get either zero (by letting $u(x,y)h_k(x,y) + v(x,y)l_k(x,y) = 0 \pmod{y^{2^k}}$) or we get a term of positive degree in x since g_k is monic in x and has positive degree in x. In either case we are done. ■

In the next lecture we wrap up the above algorithm, by performing the right preprocessing.

6.966 Algebra and Computation	October 7, 1998
Lecture 8	
Lecturer: Madhu Sudan	*Scribe: Venkatesan Guruswami*

In this lecture we will wrap up the algorithm presented in the previous lecture for bivariate polynomial factoring. We will then present some thoughts on multivariate factoring which we will see in detail in the next lecture. Finally, we will digress a little bit and present an application to decoding Reed-Solomon codes.

8.1 Bivariate Polynomial Factoring

We first recall the skeleton of the algorithm that we discussed in the last lecture. The idea is to use univariate polynomial factoring over any U.F.D for bivariate factoring. Recall that if R is a U.F.D then so is $R[x_1, x_2, \ldots, x_n]$ for any $n \geq 1$, so the factorization will be unique up to associates. Also we only seek to find some non-trivial factor, since then we can recursively compute the complete factorization.

Algorithm Bivariate-Factor:
Input: A polynomial $f \in R[x, y]$, R a U.F.D, with $\deg_x(f) \leq d$, $\deg_y(f) \leq d$.
Output: Some non-trivial monic factor $g \in R[x, y]$ of f.

1. Preprocess f appropriately – this step will be expanded soon.
2. Factor $f_0(x) = f(x, 0)$ as $f_0(x) = g_0(x)h_0(x)$ where g_0, h_0 are relatively prime and g_0 is irreducible (This is possible assuming f_0 has no repeated factors, and we will see how to remove this assumption soon). Equivalently, we can view this factorization as $f(x, y) \equiv g_0(x, y)h_0(x, y)(\mod y)$. Repeatedly apply Hensel lifting K times, for $K = 2\lceil \log d \rceil + 2$, to find g_K, h_K such that
$$f(x, y) \equiv g_K(x, y)h_K(x, y) \quad (\mod y^{2^K}).$$
3. Solve the *linear system* (with unknowns being the coefficients of G, L) to find G and L with $\deg_x(G), \deg_x(L) < \deg_x(f)$, and $\deg_y(G) \leq \deg_y(f)$ $\deg_y(l) \leq 2^K$; that satisfy
$$G(x, y) \equiv g_K(x, y)L(x, y) \quad (\mod y^{2^K}).$$
4. Find $\gcd(G, f)$, and if non-trivial output G as it as a factor of f. If it is trivial, output that f is irreducible.

We saw in the previous lecture that the Steps 2,3,4 all work if we assume that f_0 has no repeated factors. We can as usual, by a preliminary gcd computation like $\gcd(f, \frac{\partial f(x,y)}{\partial x})$, assume that f has no repeated factors, but that will not suffice – for instance when $f(x, y) = x^3 + y(y + 1)$ (which is irreducible and hence definitely square-free), $f_0(x) = x^3$, which has repeated factors. The idea to fix this is to consider "shift" y by a value $\beta \in R$ – specifically, consider $f_\beta(x, y) \stackrel{\text{def}}{=} f(x, y + \beta)$ for $\beta \in R$. Clearly, if we factor $f_\beta(x, y)$ then we can recover the factors of $f(x, y)$ as well. We hope to find (and work with) a β such that $f_\beta(x, 0)$ $(= f(x, \beta))$ has no repeated factors as a polynomial in $R[x]$.

Lemma 8.1 *If $|R| > 4d^2$, then for any polynomial $f(x, y) \in R[x, y]$ with $\deg_x(f) \leq d$ and $\deg_y(f) \leq d$, there exists a $\beta \in R$ such that $f_\beta(x, 0) \in R[x]$ has no repeated factors.*

Proof: Consider the discriminant $\text{disc}_x(f)$ treating f as belonging to $(R[y])[x]$. It is clearly a polynomial $D(y)$ in y of degree at most $(2d)^2 = 4d^2$. Hence we can find $\beta \in R$ that is not a root of D, and for such a β, we have $\text{Disc}(f_\beta(x, 0)) \neq 0$, implying $f_\beta(x, 0)$ has no repeated factors. □

Thus the preprocessing step 1 in algorithm Bivariate-Factor expands as follows:

1a. Obtain square-free factorization of f and work with each square-free factor separately. +

1b. Find $\beta \in R$ such that $f(x,\beta)$ is a square-free polynomial in $R[x]$.

1c. Let $f_\beta(x,y) \stackrel{\text{def}}{=} f(x, y+\beta)$, and work with f_β (and at the end of the algorithm recover factors of f from those of f_β).

Corollary 8.2 *For any U.F.D R with $|R| > 4d^2$, then factorization of degree d polynomials in $R[x,y]$ can be reduced to univariate factorization in $R[x]$.*

In particular the above implies we can always reduce factorization in $R[x_1, x_2, \ldots, x_n]$ to factorization in $R[x_1, x_2, \ldots, x_{n-1}$ and inductively all the way down to factorization in $R[x_1, x_2]$.

We still have to deal with the issue of small values of $|R|$. But note that if $|R|$ is small, it is in particular finite, and hence R is in fact a field. Now given $f \in R[x,y]$, we work with a finite extension R' of R such that $|R'| > 4d^2$ and obtain a non-trivial irreducible factor (if any) $g' \in R'[x,y]$ of $f \in R'[x,y]$ using the above corollary. We now need to recover a factor $g \in R[x,y]$ using $g' \in R'[x,y]$.

Lemma 8.3 *Let g', f be as above. Let $g \in R[x,y]$ be the minimal polynomial such that $g'(x,y)|g(x,y)$ (in $R'[x,y]$). Then $g(x,y)|f(x,y)$.*

Proof: Since g' is irreducible, so is the minimal polynomial $g \in R[x,y]$ such that $g'|g$. Now if g does not divide f, then $\exists\ u_x, v_x \in R[x,y]$ and $p \in R[x]$ (resp. $u_y, v_y \in R[x,y]$ and $q \in R[y]$) such that $u_x(x,y)f(x,y) + v_x(x,y)g(x,y) = p(x)$ (resp. $u_y(x,y)f(x,y) + v_y(x,y)g(x,y) = q(y)$). But then $g'(x,y)$ divides both $p(x)$ and $q(y)$ which is impossible (since g' is not a constant polynomial). $\square$

We can thus recover a factor g of f in $R[x,y]$ using g' as follows (we give only an informal description to illustrate the idea – details can be easily filled in). For every possible values d_x, d_y (try them in increasing order), try to find a polynomial $h \in R'[x,y]$ with $\deg_x(h) = d_x$ and $\deg_y(h) = d_y$ such that $g \cdot h' = g \in R[x,y]$. This can be clearly viewed as a linear system over R' with the added condition that the coefficients of g come from R. But since R' is a finite field extension of R, we can treat it as R^l for some l, and then it is easy to transform the above requirements into a linear system over R. Indeed, assume the original system is $Ax = y$ with A being a $r \times n$ matrix over R', and we seek solutions $x \in {R'}^n$ and $y \in R^r$. We can rewrite this system as the homogeneous system $Bz = \bar{0}$ over R' where $B = [A| - I_r]$ is a $r \times n + r$ matrix and we seek solutions $z_1, z_2, \ldots, z_{n+r} \in {R'}^{n+r}$ with $z_{n+i} \in R$ for $1 \le i \le r$. Since R' is a finite field extension of R, it can be viewed as the vector space R^l, with a basis $\{\alpha_1, \alpha_2, \ldots, \alpha_l\}$ where $\alpha_i \in R'$ for $1 \le i \le l$ and $\alpha_1 = 1 (\in R)$. Hence we can replace each z_i, $1 \le i \le n$, by l variables $z_{i1}, z_{i2}, \ldots, z_{il}$ that take values in R, and form a $rl \times (nl + r)$ matrix C over R by replacing each entry $b_{ij} \in R'$ of the matrix, for $1 \le j \le n$, by the $l \times l$ matrix over R to which b_{ij} corresponds when viewed as a linear transformation of R' (over the basis $\{\alpha_1, \ldots, \alpha_l\}$ for R' over R), and replacing column $n+i$ of B, for $1 \le i \le r$, by the vector which is zero everywhere except for a (-1) in row $1 + (i-1)l$. This gives a homogeneous linear system $Cz' = \bar{0}$ over R solving which will give us the original polynomials we sought (after tracing back through the above conversions).

8.2 Multivariate Factoring

We now present some thoughts on extending this to multivariate polynomials. The first natural method for this is to use the reduction of Corollary 8.2 repeatedly starting from n variables down to $n-1$ and so on till we reach two variables when we can use the previous algorithm. Note that in all these steps the underlying U.F.D is a polynomial ring and hence is infinite and in particular the condition $|R| > 4d^2$ will be satisfied. This procedure, however, is clearly not very efficient. An alternative procedure is sketched below.

Let $f \in R[x, y_1, y_2, \ldots, y_n]$ be a polynomial that has to be factored. As in the bivariate case, we first factor f (using univariate factorization) as

$$f(x, y_1, y_2, \ldots, y_n) \equiv g_0(x, y_1, y_2, \ldots, y_n) \cdot h_0(x, y_1, y_2, \ldots, y_n) \quad (\mod (y_1, y_2, \ldots, y_n)).$$

Define $p \stackrel{\text{def}}{=} f - g_0 h_0$; then $p \in (y_1, y_2, \ldots, y_n)$. As we did for the bivariate case, we once again perform repeated Hensel-lifting up to a certain number k of times, working with the ideal generated by p (i.e in the

jth step we find g_j, h_j such that $f \equiv g_j \cdot h_j (\mod p^{2^k})$. The rest of the argument will also be analogous to the bivariate case, for instance instead of computing resultants over $R(y)$ (as was done in the case of $R[x, y]$) we will now compute resultants over $R(y_1, y_2, \ldots, y_n)$, etc. We omit the details which can be easily filled in.

The second approach is more efficient than the first one we suggested, but still both the methods perform poorly for one non-circumventable (at least apparently) reason. Even though the original multivariate polynomial may be sparse and hence have small description, it may very well be the case that one of its factors in not sparse. And since a degree d polynomial on n variables can have up to $\binom{n+d}{d}$ non-zero coefficients, the run-time of these methods in general grows as $O(n^d)$ even if f itself has a small description. This is unavoidable for any algorithm that hopes to present an explicit representation of the factors in terms of their coefficients. The way around this is to ask for an algorithm that when given a degree d polynomial on n variables $f = \prod_{i=1}^{k} f_i$ (the f_i's not necessarily distinct) as a black-box (i.e the algorithm has *oracle access* to the function f), must produce k black-boxes $O_1, O_2, \ldots, O_k$ where each O_i computes f_i by (possibly) making queries to the oracle for f, and such that on any $\bar{a}$, O_i computes $f_i(\bar{a})$ efficiently by making poly(n, d) queries to the oracle for f. We will describe an algorithm by Kaltofen that achieves this in the next lecture. We now take a small digression to discuss a decoding algorithm for Reed-Solomon codes.

8.3 Application: Decoding Reed-Solomon Codes

Recall that in a $[n, k+1, n-k]_q$ Reed-Solomon code the message is given by degree k polynomials in $F_q[X]$ and the codewords are obtained by evaluating the message polynomials at n *distinct* points $x_1, x_2, \ldots, x_n \in F_q$ (so we need $q \geq n$). The distance of the code id $(n-k)$ since no two distinct degree k polynomials over a field can agree at more than k points.

Suppose a message $p \in F_q[X]_{\leq k}$ was thus encoded and transmitted, and we receive $\vec{y} = y_1, y_2, \ldots, y_n \in F_q^n$, with, say at most e, errors in transmission. We now wish to decode $\vec{y}$ and "reconstruct" p. If $e < \frac{n-k}{2}$, then we know that at most one such polynomial p exists. We now describe an algorithm to find the polynomial p in such a case.

Let $\mathcal{E} = \{i : p(x_i) \neq y_i\}$ so that $\mathcal{E}$ is the set of positions that are in error. We define the *error-locator polynomial* $E(x) = \prod_{i \in \mathcal{E}}(x - x_i)$ – we know that $\deg(E) \leq e$. Now for every i, $1 \leq i \leq n$, one of $y_i = p(x_i)$ or $E(x_i) = 0$ holds. Hence if we set $M(x) = p(x)E(x)$, then $\deg(M) \leq e + k$ and $E(x_i)y_i = M(x_i)$ for $1 \leq i \leq n$.

This motivates the following algorithm:

Algorithm Unique-Decode-RS-codes:
Input: A set of n pairs $\{(x_i, y_i) \in F_q^2 : 1 \leq i \leq n\}$ such that $\exists$ a polynomial $p \in F_q[X]$ of degree at most k such that $p(x_i) \neq y_i$ for at most e values of i. (with $e < (n-k)/2$)
Output: The polynomial p.

1. Find polynomials $F, N \in F_q[X]$, $\deg(F) \leq e$, $\deg(N) \leq e + k$, $F \neq 0$, such that $F(x_i)y_i = N(x_i)$ for $1 \leq i \leq n$.
2. Output $p(x) = \frac{N(x)}{F(x)}$. /* See the remark below for an alternative, and in fact more preferred, implementation of this step. */

To prove the algorithm works, we first note the linear system in Step 1 above has a solution – indeed the polynomials E, M from our discussion preceding the algorithm form a solution. A solution (F, N) can thus be found by solving the linear system.

We now argue that for any F, N found in Step 1, we will have $N(x) = F(x)p(x)$. To prove this, set $S = \{i : p(x_i) = y_i\}$, by hypothesis $|S| \geq n - e$. Also for every $i \in S$, $N(x_i) - F(x_i)p(x_i) = 0$. Thus the polynomial $N(x) - F(x)p(x)$ has at least $n - e$ roots, while its degree is at most $e + k$. Hence if $e + k < n - e$, the polynomial will be identically zero implying $p(x) = \frac{N(x)}{F(x)}$. But this condition is precisely $e < \frac{n-k}{2}$ which is given to us. The algorithm is thus guaranteed to work.

Remarks on Run-time of the Algorithm

- Solving the linear system $F(x_i)y_i = N(x_i)$ can be done in $O(n\text{poly}(\log n))$ time using the special form of the system.

- Often a much faster way to obtain the message polynomial $p(x)$ is to compute the (at most e) roots of $F(x)$ using a root-finding algorithm, and to then obtain $p(x)$ by fast interpolation at (any of) the points x_i which are *not* roots of $F(x)$. Since the degree of F is at most e, and e is often very small, this method is in practice much faster. In fact, this was one of the primary motivations to develop fast algorithms for root-finding of univariate polynomials.

6.966 Algebra and Computation October 8, 1998

Lecture 9

Lecturer: Madhu Sudan *Scribe: Amos Beimel*

In this lecture we will discuss Hilbert's irreducibility and show how to use it for black-box factorization of multivariate polynomials.

9.1 Hilbert Irreducibility Theorem

9.1.1 Motivating Discussion

Let f be a multivariate polynomial over the integers, that is $f(x_1, x_2, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$. We want to check if f is irreducible. Clearly, this can be done using an exhaustive search. Hilbert asked if this can be done more efficiently. His suggestion was to substitute the variables, that is, let $x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n$, where $a_i \in \mathbb{Z}$, and check if $f(a_1, a_2, \ldots, a_n)$ is a prime. If f is reducible, say

$$f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) \cdot h(x_1, x_2, \ldots, x_n),$$

then "usually" $g(a_1, a_2, \ldots, a_n) \neq 1$ and $h(a_1, a_2, \ldots, a_n) \neq 1$ and $f(a_1, a_2, \ldots, a_n)$ is composite. Hilbert showed that some converse of this statement is true; if $f(a_1, a_2, \ldots, a_n)$ is often a composite then f is reducible.

The previous example is, in some sense, a reduction in the wrong direction since over the integers it is easier to check irreducibility then to check primality. However, we want to use this idea over arbitrary fields. The first obstacle is that $f(a_1, a_2, \ldots, a_n)$ is always a product of two elements of the field. To overcome this problem we use *partial substitutions*. That is, we substitute the values of the variables such that the result is a polynomial in few variables. A simple partial substitution is to substitute all the variables except for x_1: given $f(x_1, x_2, \ldots, x_n)$ we choose $a_2, \ldots, a_n$ and consider the polynomial $f(x_1, a_2, \ldots, a_n)$ (this is a univariate polynomial in x_1). This substitution and every other substitution which results in univariate polynomial are not good enough since over algebraically closed fields (e.g., $\mathcal{C}$) every univariate polynomial always factors. We will use substitutions which result in bivariate polynomials. This is a better idea since over every field there are irreducible bivariate polynomials, e.g., $x - y^2$.

9.1.2 The Theorem

We use the following substitution rule: for $f(x, y_1, y_2, \ldots, y_n) \in \mathbb{F}[x, y_1, y_2, \ldots, y_n]$ and $\hat{a}, \hat{b} \in \mathbb{F}^n$ we consider the bivariate polynomial $f(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n)$. If f is reducible, say $f = p \cdot q$, then with high probability $p(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n)$ and $q(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n)$ are not constant, and $f(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n)$ is reducible. The next theorem states that the converse is true if f is monic in x. In Section 9.1.7 we show how to check irreducibility of non-monic polynomials.

Theorem 9.1 *Let $S \subseteq \mathbb{F}$ be a finite set, and $f(x, y_1, y_2, \ldots, y_n)$ be a monic polynomial in x whose total degree is at most d. If*

$$\frac{\partial f}{\partial x} \neq 0$$

and

$$\Pr_{\hat{a}, \hat{b}} [f(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n) \textit{ is reducible }] \geq \frac{O(d^5)}{|S|},$$

where $\hat{a}$ and $\hat{b}$ are chosen uniformly and independently from S^n, then f is reducible.

The rest of this section is devoted to the proof of this theorem.

9.1.3 A Warm-up Lemma

In this section we prove that if with "high" probability $f(x, b_1, \ldots, b_n)$ is not square free then f is reducible. This lemma will be used in the sequel to assume that $f(x, 0, \ldots, 0)$ is square free. Furthermore, the proof of this lemma is a warm-up for the proof of Theorem 9.1. This lemma requires that $\frac{\partial f}{\partial x} \neq 0$, which is not really a restriction when the characteristic of $\mathbb{F}$ is zero, but is more problematic over finite fields. We show how to check irreducibility over finite fields in Section 9.1.8. We start with a simple observation.

Observation 9.2 *If the polynomial $f(x, y_1, \ldots, y_n)$ is monic in x and $g(x, y_1, \ldots, y_n)$ divides f then g is monic in x.*

Lemma 9.3 *If $\frac{\partial f}{\partial x} \neq 0$ and*

$$\Pr_{\hat{b} \in S^n} [\ f(x, b_1, \ldots, b_n) \textit{ is not square free }] > \frac{2d^2}{|S|},$$

then f is reducible.

Proof Recall that $f(x, b_1, \ldots, b_n)$ is a univariate polynomial in $\mathbb{F}[x]$. We denote the discriminant of $f(x, b_1, \ldots, b_n)$ by $\zeta_{\hat{b}}$. If $f(x, b_1, \ldots, b_n)$ is not square free for some $\hat{b}$ then

$$\gcd\left(f(x, b_1, \ldots, b_n), \frac{df(x, b_1, \ldots, b_n)}{dx}\right) \neq 1.$$

That is, the discriminant $\zeta_{\hat{b}}$ is 0.

Now, view $f(x, y_1, \ldots, y_n)$ as a polynomial in $\mathbb{F}(y_1, \ldots, y_n)[x]$, that is, as a polynomial in x whose coefficients are polynomials in $y_1, \ldots, y_n$. The discriminant of f is a polynomial in $y_1, \ldots, y_n$ which we denote $\zeta(y_1, \ldots, y_n)$. Clearly, $\zeta(b_1, \ldots, b_n) = \zeta_{\hat{b}}$. By the conditions of the lemma, with probability greater than $(2d^2)/|S|$ over the $\hat{b}$'s, the discriminant $\zeta(b_1, \ldots, b_n)$ is zero. Thus, $\zeta(y_1, \ldots, y_n)$ is a polynomial of the n variables $y_1, \ldots, y_n$ which has degree at most $2d^2$, and is zero with "high" probability. By the Schwartz-Zippel lemma, the discriminant of f, namely $\zeta(y_1, \ldots, y_n)$, is identically zero, i.e., $\gcd(f, \frac{\partial f}{\partial x}) \neq 1$. Since $\frac{\partial f}{\partial x} \neq 0$, this means that f is reducible.[7] ■

We conclude that either with high probability $f(x, b_1, \ldots, b_n)$ is not square free and by Lemma 9.3 the polynomial f is reducible, or there exists some $\hat{b}$ such that $f(x, b_1, \ldots, b_n)$ is square free. We fix such $\hat{b}$ and consider the polynomial $f(x, y_1 + b_1, \ldots, y_n + b_n)$ which is reducible if and only if $f(x, y_1 + b_1, \ldots, y_n + b_n)$ is reducible. Thus,

$$\Pr_{\hat{a}} [f(x, a_1 t + b_1, a_2 t + b_2, \ldots, a_n t + b_n) \text{ is reducible }] \geq \frac{O(d^5)}{|S|},$$

where the probability is only taken over the vectors $\hat{a}$ chosen uniformly from S^n. To simplify the notations, we "forget" the fixed $\hat{b}$ and assume that $f(x, 0, \ldots, 0)$ is square free.

9.1.4 Ideas of the proof

We next summarize the ideas of the proof of Lemma 9.3 and then we show how we use this ideas for proving the the theorem. We have shown that if $f(x, b_1, \ldots, b_n)$ is not square free then its discriminant is zero. We observe that the discriminant of $f(x, b_1, \ldots, b_n)$ is obtained from the discriminant of $f(x, y_1, \ldots, y_n)$ by substituting $y_1 = b_1, \ldots, y_n = b_n$. Thus, the discriminant of $f(x, y_1, \ldots, y_n)$ is a polynomial that is zero for "many" substitutions and therefore must be identically zero. However, if the discriminant of f is zero then f is reducible.

A similar idea is used to prove Theorem 9.1. We use Hensel's lifting lemma to find a factor of the univariate polynomial $f(x, a_1 t, \ldots, a_n t)$. If $f(x, a_1 t, \ldots, a_n t)$ is reducible (and it is with "high" probability) then we

[7] Over fields of characteristic zero we can deduce that f is not square free. This is not necessarily true over finite fields.

succeed in finding a factor (since we can assume that $f(x,0,\ldots,0)$ is square free). We use the same procedure to try and find a factor of f. However, in the latter case we do not know if the conditions that guaranteed the success for finding a factor hold. Nevertheless, we observe that we succeed if some determinant is zero. Similar to the proof of Lemma 9.3, we substitute $y_1 = a_1, \ldots, y_n = a_n$ in this determinant and observe that since we managed to factor $f(x, a_1t, \ldots, a_nt)$ then under this substitution this determinant is zero. This implies that the determinant is zero for many substitutions and therefore must be identically zero and we succeed in factoring f.

9.1.5 A Technical Claim

In this section we define some auxiliary polynomial and investigate some of their properties. First, we define

$$p(x,t,y_1,y_2,\ldots,y_n) \stackrel{\text{def}}{=} f(x, y_1t, y_2t, \ldots, y_nt).$$

Next, we define for every $\hat{a} \in \mathbb{F}^n$ a bivariate polynomial

$$f_{\hat{a}}(x,t) \stackrel{\text{def}}{=} f(x, a_1t, a_2t, \ldots, a_nt).$$

Notice that $p(x,t,a_1,a_2,\ldots,a_n) = f_{\hat{a}}(x,t)$, and $p(x,1,y_1,\ldots,y_n) = f(x,y_1,\ldots,y_n)$. We prove that it is enough to prove that p is reducible.

Claim 9.4 *If p is reducible then f is reducible.*

Proof Assume $p = \alpha \cdot \beta$ for some polynomials α and β which are not constants. Recall that f is monic in x.[8] This implies that p, α and β are monic in x.

Thus,

$$f(x,y_1,\ldots,y_n) \;=\; p(x,1,y_1,\ldots,y_n) \;=\; \alpha(x,1,y_1,\ldots,y_n)\cdot\beta(x,1,y_1,\ldots,y_n).$$

Since f is irreducible then either $\alpha(x,1,y_1,\ldots,y_n)$ is constant or $\beta(x,1,y_1,\ldots,y_n)$ is constant. W.l.o.g., assume that $\alpha(x,1,y_1,\ldots,y_n) \equiv 1$. Since α is non-constant, there is a non-zero polynomial $\alpha_1(x,t,y_1,\ldots,y_n)$ such that

$$\alpha(x,t,y_1,\ldots,y_n) = 1 + (1-t)\alpha_1(x,t,y_1,\ldots,y_n).$$

This contradicts the fact that α is monic in x. ■

9.1.6 Using Hensel's Lifting Lemma

This section is the main technical part of the proof of Theorem 9.1. We show how to use the factoring algorithm provided by Hensel's lifting lemma to prove that p is reducible. As explained in Section 9.1.3, we can assume that $f(x,0,\ldots,0)$ is square free.

Since $f(x,0,\ldots,0)$ is monic in x and $f(x,a_1,\ldots,a_n)$ factors for some $\hat{a}$, there are non-constant polynomials $g_0(x)$ and $h_0(x)$ such that

$$f(x,0,\ldots,0) = g_0(x)\cdot h_0(x). \tag{9.1}$$

It must hold that $\gcd(g_0(x)\cdot h_0(x)) = 1$ (since $f(x,0,\ldots,0)$ is square free), and we can choose g_0 and h_0 such that g_0 is monic and irreducible. Since $f(x,0,\ldots,0) = f_{\hat{a}}(x,t) \pmod t$, we get

$$f_{\hat{a}}(x,t) = g_0(x)\cdot h_0(x) \pmod t. \tag{9.2}$$

For every $\hat{a} \in \mathbb{F}^n$ we can apply Hensel's lifting lemma to $f_{\hat{a}}$ and find polynomials $g_{k,\hat{a}}(x,t)$ and $h_{k,\hat{a}}(x,t)$ such that

$$f_{\hat{a}}(x,t) = g_{k,\hat{a}}(x,t)\cdot h_{k,\hat{a}}(x,t) \pmod{t^k}. \tag{9.3}$$

Furthermore, $g_{k,\hat{a}}(x,t) = g_0(x) \pmod t$ and $h_{k,\hat{a}}(x,t) = h_0(x) \pmod t$.

[8] If f is not monic in x then p might be reducible. For example, if $f(x,y_1,y_2) = xy_1 + y_2$ then $p(x,t,y_1) = t(xy_1+y_2)$ is reducible.

Claim 9.5 *There exists a polynomial $g_k(x,t,y_1,\ldots,y_n)$ such that for every $\hat{a} \in \mathbb{F}^n$*

$$g_{k,\hat{a}}(x,t) = g_k(x,t,a_1,\ldots,a_n).$$

Proof Equation (9.1) implies that

$$f(x,y_1,\ldots,y_n) = g_0(x)\cdot h_0(x) \pmod{y_1,\ldots,y_n} \tag{9.4}$$

(since $f(x,0,\ldots,0) = f(x,y_1,\ldots,y_n) \pmod{y_1,\ldots,y_n}$). Thus, we can apply Hensel's lifting lemma to f and find polynomials $g'_k(x,y_1,\ldots,y_n)$ and $h'_k(x,y_1,\ldots,y_n)$ such that

$$f(x,y_1,\ldots,y_n) = g'_k(x,y_1,\ldots,y_n)\cdot h'_k(x,y_1,\ldots,y_n) \pmod{(y_1,\ldots,y_n)^k}. \tag{9.5}$$

Now, define

$$g_k(x,t,y_1,\ldots,y_n) \stackrel{\text{def}}{=} g'_k(x,y_1t,\ldots,y_nt),$$

and

$$h_k(x,t,y_1,\ldots,y_n) \stackrel{\text{def}}{=} h'_k(x,y_1t,\ldots,y_nt).$$

It holds that

$$\begin{aligned} p(x,t,y_1,\ldots,y_n) &= f(x,y_1t,\ldots,y_nt) \\ &= g'_k(x,y_1t,\ldots,y_nt)\cdot h'_k(x,y_1t,\ldots,y_nt) \pmod{(y_1,\ldots,y_n)^k} \\ &= g_k(x,t,y_1,\ldots,y_n)\cdot h_k(x,t,y_1,\ldots,y_n) \pmod{t^k} \end{aligned}$$

(the last equality holds since in any monomial in g_k and h_k the degree of t is the sum of the degrees of the y_i's in the monomial). In particular,

$$f_{\hat{a}}(x,t) = p(x,t,a_1,\ldots,a_n) = g_k(x,t,a_1,\ldots,a_n)\cdot h_k(x,t,a_1,\ldots,a_n) \pmod{t^k}.$$

By the uniqueness guaranteed by Hensel's lemma we deduce $g_k(x,t,a_1,\ldots,a_n) = g_{k,\hat{a}}(x,t)$. ■

As we proved in Lectures 7, if $f_{\hat{a}}$ is reducible, g_0 and h_0 have no common factors, and g_0 is irreducible then for $k = O(d^2)$ there are polynomials $g_{\hat{a}}(x,t), \ell_{k,\hat{a}}(x,t) \neq 0$ such that

$$\begin{aligned} g_{\hat{a}}(x,t) &= g_{k,\hat{a}}(x,t)\cdot \ell_{k,\hat{a}}(x,t) \pmod{t^k}, && (9.6)\\ \deg_x(g_{\hat{a}}(x,t)) &< \deg_x(f_{\hat{a}}(x,t)), \text{ and } \deg_t(g_{\hat{a}}(x,t)) \leq \deg_t(f_{\hat{a}}(x,t)). && (9.7) \end{aligned}$$

We next prove that there are polynomials g, ℓ_k satisfying similar properties with respect to p.

Claim 9.6 *There are polynomials $g(x,t,y_1,\ldots,y_n), \ell_k(x,t,y_1,\ldots,y_n) \neq 0$ such that*

$$\begin{aligned} g(x,t,y_1,\ldots,y_n) &= g_k(x,t,y_1,\ldots,y_n)\cdot \ell_k(x,t,y_1,\ldots,y_n) \pmod{t^k}, && (9.8)\\ \deg_x(g(x,t,y_1,\ldots,y_n)) &< \deg_x(p(x,t,y_1,\ldots,y_n)), && (9.9)\\ \deg_t(g(x,t,y_1,\ldots,y_n)) &\leq \deg_x(p(x,t,y_1,\ldots,y_n)), && (9.10)\\ \sum_{i=1}^{n} \deg_{y_i}(g(x,t,y_1,\ldots,y_n)) &= O(k^5). && (9.11) \end{aligned}$$

Proof Using Claim 9.5 and Equation (9.6) can get

$$g_{\hat{a}}(x,t) = g_k(x,t,a_1,\ldots,a_n)\cdot \ell_{k,\hat{a}}(x,t) \pmod{t^k}. \tag{9.12}$$

This is a homogeneous linear system, where the unknowns are the coefficients of $g_{\hat{a}}$ and $\ell_{\hat{a},k}$. The degree of x in all these polynomials is at most d (where d is the total degree of f) and the degree of t in all these polynomials is less than $k = O(d^2)$. Thus, the number of unknowns in the system, denoted m is $O(d^3)$. If $f_{\hat{a}}$ is reducible, then this system has a solution in which the degrees of x and t in $g_{\hat{a}}$ are as specified in (9.7).

These degree restrictions are expressed as additional linear equations which state that some coefficients are equal to zero.

Now, consider the following homogeneous linear system

$$g(x,t,y_1,\ldots,y_n) = g_k(x,t,y_1,\ldots,y_n)\cdot \ell_k(x,t,y_1,\ldots,y_n) \pmod{t^k}.$$

We view g, g_k and ℓ_k as polynomials in $\mathbb{F}(y_1,\ldots,y_n)[x,t]$, and this system also has m unknowns. If this system has a no solution (in the field $\mathbb{F}(y_1,\ldots,y_n)$) where the degrees of x and t in g are "small", the linear system defines an $m\times m$ square submatrix that has a non-zero determinant. This determinant is a polynomial $D(y_1,\ldots,y_n)$ whose total degree is at most $mk = O(d^5)$ (since every entry in the matrix is a polynomial of total degree less than k). However, for every $\hat{a}$ such that $f_{\hat{a}}$ is reducible the determinant $D(a_1,\ldots,a_n)$ is zero (since (9.12) has a solution). By the Schwartz-Zippel lemma $D(y_1,\ldots,y_n)\equiv 0$, thus we can solve the linear system and find g and ℓ_k. Since this is a homogeneous system, we can always find a solution in $\mathbb{F}[y_1,\ldots,y_n]$ (and not just in $\mathbb{F}(y_1,\ldots,y_n)$). Furthermore, using Gaussian elimination, we obtain a solution in which the total degree of each coefficient is $O(k^5)$. ■

As proved in previous lectures, every polynomial satisfying (9.8) – (9.10) can be used to find a non-trivial factor of p. The problem is that this factor is in the field $\mathbb{F}(y_1,\ldots,y_n)$. However, we know that for every $\hat{a}\in\mathbb{F}^n$ the polynomials $g(x,t,a_1,\ldots,a_n)$ and $\ell_k(x,t,a_1,\ldots,a_n)$ are a solution of (9.6) and (9.7). Thus, as proved in Lectures 7 and 8,

$$\gcd_x(p(x,t,a_1,\ldots,a_n), g(x,t,a_1,\ldots,a_n)) \neq 1, \tag{9.13}$$

which implies that the resultant of $p(x,t,a_1,\ldots,a_n)$ and $g(x,t,a_1,\ldots,a_n)$ is zero. That is, the resultant of $p(x,t,y_1,\ldots,y_n)$ and $g(x,t,y_1,\ldots,y_n)$ is zero for every substitution of $y_1,\ldots,y_n$, i.e., the resultant is identically zero. Hence, p has a non-trivial factor, and the polynomials p and f are reducible. This completes the proof of Theorem 9.1.

9.1.7 Testing Irreducibility of Non-monic Polynomials

In Theorem 9.1 we assumed that f is monic in x. We show how to remove this assumption. Given a polynomial $r(x,y_1,\ldots,y_n)$ which has degree at least 1 in x, we construct a polynomial $f(x,y_1,\ldots,y_n)$ that is monic in x such that f is reducible then r is reducible, and if r has a "good" factorization (as described in Claim 9.8) then f is reducible. Furthermore, from the factorization of f, one can easily construct the factorization of r.

We next present the construction. Let $r(x,y_1,\ldots,y_n)=\sum_{i=0}^{e} r_i(y_1,\ldots,y_n)x^i$ where e is the degree of x in r, that is, $r_e\neq 0$. We define

$$f(x,y_1,\ldots,y_n)\stackrel{\text{def}}{=}(r_e(y_1,\ldots,y_n))^{e-1}\cdot r\left(\frac{x}{r_e(y_1,\ldots,y_n)}, y_1,\ldots,y_n\right). \tag{9.14}$$

Claim 9.7 *The function f is a polynomial in $\mathbb{F}[x,y_1,\ldots,y_n]$ which is monic in x. Furthermore, if f is reducible then r is reducible.*

Proof By definition of f it holds that

$$\begin{aligned} f(x,y_1,\ldots,y_n) &= \sum_{i=0}^{e}(r_e(y_1,\ldots,y_n))^{e-1-i} r_i(y_1,\ldots,y_n)x^i \\ &= x^e + \sum_{i=0}^{e-1}(r_e(y_1,\ldots,y_n))^{e-1-i} r_i(y_1,\ldots,y_n)x^i. \end{aligned}$$

Thus, f is a polynomial in $\mathbb{F}[x,y_1,\ldots,y_n]$ that is monic in x.

Furthermore,

$$r(x,y_1,\ldots,y_n)\cdot(r_e(y_1,\ldots,y_n))^{e-1} = f\left(x\cdot r_e(y_1,\ldots,y_n), y_1,\ldots,y_n\right).$$

Assume that f is reducible, say $f = g \cdot h$. Then,

$$r(x, y_1, \ldots, y_n) \cdot (r_e(y_1, \ldots, y_n))^{e-1} \quad = \quad g(x \cdot r_e(y_1, \ldots, y_n), y_1, \ldots, y_n) \cdot h(x \cdot r_e(y_1, \ldots, y_n), y_1, \ldots, y_n).$$

Thus, there is some integer $e_1 \geq 0$ and polynomials g' and h' such that

$$g(x \cdot r_e(y_1, \ldots, y_n), y_1, \ldots, y_n) = r_e(y_1, \ldots, y_n)^{e_1} g'(x, y_1, \ldots, y_n)$$

and

$$h(x \cdot r_e(y_1, \ldots, y_n), y_1, \ldots, y_n) = r_e(y_1, \ldots, y_n)^{e-e_1-1} h'(x, y_1, \ldots, y_n).$$

Therefore, $r(x, y_1, \ldots, y_n) = g'(x, y_1, \ldots, y_n) \cdot h'(x, y_1, \ldots, y_n)$. Since f is monic in x, the polynomials g and h are monic in x as well, and it is not possible that g' or h' are constant. Thus, we get a factorization of r. ■

We next claim that if r is reducible and has a "good" factorization then f is reducible. The proof is similar to the proof of Claim 9.7 and therefore omitted.

Claim 9.8 *If $r = g(x, y_1, \ldots, y_n) \cdot h(x, y_1, \ldots, y_n)$ for some polynomials g and h such that $\deg_x(g) > 0$ and $\deg_x(h) > 0$ then f is reducible.*

Therefore, we get an efficient procedure to check if a polynomial $r(x, y_1, \ldots, y_n)$ is irreducible.

- If $\gcd(r_0(y_1, \ldots, y_n), r_1(y_1, \ldots, y_n), \ldots, r_e(y_1, \ldots, y_n)) \neq 1$ then r is reducible.
- Otherwise, construct f as described in (9.14).
- Test, using Hilbert's irreducibility theorem, if f is reducible. That is, choose $\hat{a}$ and $\hat{b}$ at random from S^n. If $f(x, ta_1 + b_1, \ldots, ta_n + b_n)$ is reducible, output "r is reducible".

Notice that if the degree of r is d then the degree of f is at most d^2, thus to use Hilbert's irreducibility theorem we need a field $\mathbb{F}$ with at least $\Omega(d^{10})$ elements.

9.1.8 Checking Irreducibility over Finite Fields

Theorem 9.1 requires that $\frac{\partial f}{\partial x} \neq 0$. Over fields with characteristic zero this only means that we have to choose a variable that appears in the polynomial. However, over finite fields there are irreducible polynomials r whose degree in x_1 is at least 1 but $\frac{\partial r(x_1, \ldots, x_n)}{\partial x_1} = 0$ (e.g., consider the irreducible polynomial $x_1^6 + x_2$ over $GF(2)$). Nevertheless, if $\frac{\partial r(x_1, \ldots, x_n)}{\partial x_i} = 0$ for every i then the degree of each variable in each monomial is divisible by p – the characteristic of the field. Thus, if r is a non-constant polynomial and $\mathbb{F}$ is a finite field then r is a p-th power of some polynomial.

We are left with the case that $\frac{\partial r(x_1, \ldots, x_{n+1})}{\partial x_i} \neq 0$ for some i. In this case if we covert r to a function f that is monic in x_i as described in Section 9.1.7, then $\frac{\partial f(y_1, \ldots, x_n)}{\partial x_i} \neq 0$. Thus, we can use the test of Section 9.1.7 to check if r is reducible.

9.2 Black-box Factorization of Multivariate Polynomials

The task of black-box factorization of multivariate polynomials is defined as follows. Given a black-box access to a polynomial f, construct black-boxes for its factors where each black-box has access to the black-box for f. We emphasize that we do not know the number of factors in advance and we have to compute it. This is done in a preprocessing stage, in which we also "identify" each black-box with a factor of f. Later on, we are given an assignment and an index i and we need to return the value of the i-th factor on this assignment. Both the preprocessing stage and the computing stage use the black-box for f.

We use Hilbert's irreducibility theorem as well as an algorithm for factoring polynomials with 3 variables for the black-box factorization. The preprocessing stage we describe is randomized and has a small probability of failing. However, if the preprocessing stage succeeded then we always compute the correct value of the factor (provided that we manage to factor some polynomial with 3 variables). In the rest of this section we assume that we are working over a field that is big enough, f is monic in x, and $\frac{\partial f}{\partial x} \neq 0$.[9]

9.2.1 Preprocessing Stage

Our first task is to determine the number of factors of f. Assume that f has $n+1$ variables $x, y_1, \ldots, y_n$, it is monic in x, the derivative $\frac{\partial f}{\partial x}$ is non-zero, and the total degree of f is at most d. Furthermore, let $S \subseteq \mathbb{F}$ be a set of size $\Omega(d^6/\epsilon)$ (for some $\epsilon < 1$). To compute the number of factors we use the following simple algorithm:

- Chose $\hat{a}, \hat{b}$ at random from S^n.
- Factor $f_{\hat{a},\hat{b}}(x,t) \stackrel{\text{def}}{=} f(x, a_1t_1+b_1, \ldots, a_nt_n+b_n)$. Let ℓ be the number of factors and $\tilde{f}^1(x,t), \ldots, \tilde{f}^\ell(x,t)$ be the factors.

Assume that f has ℓ' irreducible factors $f^1(x, y_1, \ldots, y_n), \ldots, f^{\ell'}(x, y_1, \ldots, y_n)$. Since f is monic in x, all the factors are monic in x. For a fixed i, the probability that the polynomial $f^i_{\hat{a},\hat{b}}(x,t) \stackrel{\text{def}}{=} f^i(x, a_1t_1+b_1, \ldots, a_nt_n+b_n)$ is irreducible is at most $O(d^5/|S|)$. Since $\ell' \leq d$, the probability that there exists some f^i that is reducible is at most $O(d^6/|S|) \leq \epsilon$. Each $f^i_{\hat{a},\hat{b}}(x,t)$ is not constant since f^i is monic in x. These two facts imply that $\ell' = \ell$ with probability at least $1-\epsilon$. By renaming $f^1, \ldots, f^\ell$ we can assume that $\tilde{f}^i(x,t) = f^i_{\hat{a},\hat{b}}(x,t)$. We define the i-th black-box as computing the factor f^i of f.

We should remark that to factor $f_{\hat{a},\hat{b}}(x,t)$ we need to have its representation as a bivariate polynomial while we only have a black-box to it. However, we can interpolate and find the $\binom{d+1}{2}$ coefficients of $f_{\hat{a},\hat{b}}$.

9.2.2 Computing the Value of Box B_i

Given $\alpha, \beta_1, \ldots, \beta_n$ we want to compute $f^i(\alpha, \beta_1, \ldots, \beta_n)$. For this we define the function

$$g(x, t_1, t_2) \stackrel{\text{def}}{=} f(x, a_1t_1 + b_1 + (\beta_1 - b_1)t_2, \ldots, a_nt_n + b_n + (\beta_n - b_n)t_2).$$

The function g is a restriction of f to a plane that contains both the point $\alpha, \beta_1, \ldots, \beta_n$ and the line $a_1t+b_1, \ldots, a_nt+b_n$. That is, $g(\alpha, 0, 1) = f(\alpha, \beta_1, \ldots, \beta_n)$ and $g(x, t, 0) = f_{\hat{a},\hat{b}}(x,t)$. The following algorithm computes $f^i(\alpha, \beta_1, \ldots, \beta_n)$.

1. Factor $g(x, t_1, t_2)$ into factors $g^1, \ldots, g^{\ell''}$.
2. Find an index j such that $g^j(x, t, 0) = \tilde{f}^i$.
3. Output $g^j(\alpha, 0, 1)$.

Lemma 9.9 *If the preprocessing stage succeeded and the factorization of g is correct, then the above algorithm is correct.*

Proof We will prove that we can always find an index j in Step 2 of the algorithm, and

$$g^j(x, t_1, t_2) = f^i(x, a_1t_1 + b_1 + (\beta_1 - b_1)t_2, \ldots, a_nt_n + b_n + (\beta_n - b_n)t_2).$$

Thus, $g^j(\alpha, 0, 1) = f^i(\alpha, \beta, \ldots, \beta_n)$ as required.

First,

$$g(x, t_1, t_2) = \prod_{i=1}^{\ell} f^i(x, a_1t_1 + b_1 + (\beta_1 - b_1)t_2, \ldots, a_nt_n + b_n + (\beta_n - b_n)t_2). \tag{9.15}$$

[9] In the black box scenario, we cannot use the construction of Section 9.1.7 since we do not know what the coefficient of the largest power of x is.

Since f is monic in x, every factor f^i is monic in x and $f^i(x, a_1t_1+b_1+(\beta_1-b_1)t_2, \ldots, a_nt_n+b_n+(\beta_n-b_n)t_2)$ is not constant. Thus, g has at least ℓ irreducible factors. Now,

$$f_{\hat{a},\hat{b}}(x,t) = g(x,t,0) = \prod_{j=1}^{\ell''} g^j(x,t,0).$$

However, g is monic in x and therefore $g^j(x,t_1,t_2)$ is monic in x and $g^j(x,t,0)$ is non-constant. Since $f_{\hat{a},\hat{b}}$ has ℓ irreducible factors, g has at most ℓ factors. Thus, g has exactly ℓ factors which are described in Equation (9.15). Since $\tilde{f}^i(x,t) = f^i_{\hat{a},\hat{b}}(x,t)$, the lemma follows. ■

6.966 Algebra and Computation	October 13, 1998
Lecture 10	
Lecturer: Madhu Sudan	*Scribe: Prahladh Harsha*

10.1 Factoring Univariate Polynomials over Integers

In this lecture, we shall discuss factorisation of univariate polynomials over integers. Factoring polynomials over the rationals can be reduced to this case by a clearing of denominators. This factorisation can be further extended to factoring multivariate polynomials as discussed in the previous 3 lectures.

Given a polynomial $f(x) = \sum_{i=0}^{n} a_i x^i \in \mathbb{Z}[x], (a_n \neq 0)$, factoring f involves finding irreducible polynomials $f_1, f_2, \ldots f_k \in \mathbb{Z}[x]$ and $c \in \mathbb{Z}$ such that $f(x) = cf_1(x)f_2(x)\ldots f_k(x)$. As integer factorisation is hard, we will relax the uniqueness restrictions to just the following: each of the f_i's is an irreducible polynomial over $\mathbb{Z}$ of degree greater than 1 and $c \in \mathbb{Z}$. In this lecture, we shall present a deterministic algorithm which on input a polynomial $f \in \mathbb{Z}[x]$, determines whether f is irreducible or not and if not outputs a non-trivial factorisation of f and which runs in polynomial time in terms of the length of the coefficients of f.

One possible direction we could proceed along to factorise polynomials would be to consider the factorisation of $f \pmod p$ for several choices of primes p and find out whether these factorisations reveal anything about the factorisation of f over $\mathbb{Z}$. However, not much information is obtained in this line as there do exist ireducible polynomials $f \in \mathbb{Z}[x]$ such that $f \pmod p$ factors for every prime p.

Another line of approach could be to consider the factorisation of f modulo a suitable prime p and then employ Hensel's Lemma to lift this factorisation to a factorisation modulo p^k. We shall adopt this method which runs parallel to the bivariate polynomial factorisation discussed in Lecture 7.

10.2 Factoring Algorithm

The factoring algorithm is outlined below. We shall expand on the individual steps later in this lecture.
Input: A polynomial $f(x) = \sum_{i=0}^{n} a_i x^i \in \mathbb{Z}[x]$ with $|a_i| \approx l$-bits for each i.
Output: A non-trivial factorisation of f if f is reducible over $\mathbb{Z}[x]$.

1. Preprocessing of f: Firstly perform Square Free Factorisation and reduce the problem to the case of factoring a square free polynomial f. Then obtain a suitable "small" prime p such that $a_n \neq 0 \pmod p$. (Such a prime p polynomial in terms of l and n exists and can be found out easily.)

2. (a) Factor $f(x) \pmod p$ as $f(x) = g_0(x)h_0(x) \pmod p$ where g_0 is an irreducible monic polynomial relatively prime to h_0. (Note such a g_0 exists as f is square free)

   (b) Iteratively lift this factorisation using Hensel's Lemma to obtain
   $$f(x) = g_k(x)h_k(x) \pmod{p^k}$$

3. Find $\tilde{g}(x), l_k(x)$ such that
   $$\begin{aligned} \tilde{g}(x) &= g_k(x)l_k(x) \pmod{p^k} \\ \deg(\tilde{g}), \deg(l_k) &\leq \deg(f) \end{aligned} \tag{10.1}$$
   and $\tilde{g}$ has "small" coefficients.

4. Compute $h = \gcd(\tilde{g}, f)$. If h is non-trivial output $h, \frac{f}{h}$ as a non-trivial factorisation of f, else output that f is irreducible.

Before dwelling into the details of this algorithm, let us first make an observation regarding the size of the coefficients of any factor of f.

Claim 10.1 *For any factor g of f in $\mathbb{Z}[x]$ where $f = \sum_{i=0}^{n} a_i x^i$ and $|a_i| \approx l-$bits, there is a polynomial p in l and n such that the coefficients of g are not more than $p(l, n)$ bits long.*

Proof We shall first show that the roots (complex ones including) of f are not very large compared to the size of the coefficients of f. In fact, for all α such that $f(\alpha) = 0$, we have $|\alpha| \leq n \max_i |a_i|$.

Suppose the contrary,(ie., $|\alpha| > n \max_i |a_i|$), then

$$\begin{aligned} |a_n \alpha^n| &\geq |\alpha|^n \\ |\sum_{i=0}^{n-1} a_i \alpha^i| &\leq n \max_i |a_i| |\alpha|^{n-1} \\ \text{Hence } |f(\alpha)| &= |a_n \alpha^n + \sum_{i=0}^{n-1} a_i \alpha^i| \\ &\geq |\alpha|^n - n \max_i |a_i| |\alpha|^{n-1} \\ &= |\alpha|^{n-1}(|\alpha| - n \max_i |a_i|) \\ &> 0 \end{aligned}$$

This contradicts the fact that α is a root of f. If $S = \{\alpha_1, \ldots, \alpha_n\}$ is the set of roots of f then any factor g of f is of the form $g(x) = \Pi_{\alpha_i \in S'}(x - \alpha_i)$ for some $S' \subseteq S$. Hence the coefficients of g are polynomials in $\alpha_1, \ldots, \alpha_n$ of degree atmost n. This implies that there exists a polynomial p in l and n such that the coefficients of g are smaller than $2^{p(l,n)}$. ■

The "lifting" of the factorisation performed in step 2(b) is exactly similar to that performed in the bivariate polynomial factorisation (ref: Lecture 7). We recall Hensel's Lemma

Lemma 10.2 (Hensel's lifting) *Let R be a UFD and $I \subset R$ be an ideal. (Here $I = (p^k)$) For any $f \in R$ and for any factorisation$f = gh(\bmod I)$ of f and there exist a, b such that $ag + bh = 1(\bmod I)$, then there exist g^*, h^*, a^*, b^* such that*

$$f = g^* h^* (\bmod I^2)$$
$$g^* = g(\bmod I)$$
$$h^* = h(\bmod I)$$
$$a^* g^* + b^* h^* = 1(\bmod I^2)$$

and the solution g^*, h^* is unique in the sense: If any other g', h' also satisy these properties, then

$$g' = g^*(1 + u)(\bmod I^2)$$
$$h' = h^*(1 - u)(\bmod I^2)$$

for some $u \in I$.

Coming to steps 3 and 4 of the algorithm, as before we have the following questions:

- How do we know that $\tilde{g}$ and l_k satisfying (1) exist?
- How do we find such a pair $\tilde{g}$ and l_k?
- Why does the existence of $\tilde{g}$ and l_k yield anything in step 4?

The answers to the first two questions mentioned above are precisely the same as those discusse in the bivariate polynomial factorisation. The set of $(\tilde{g}, l_k)$ is a linear space and thus obtaining $\tilde{g}$ and l_k is essentially solving a linear system (with unknowns being the coefficents of $\tilde{g}$ and l_k)

Now, we need to see why the gcd of f and $\tilde{g}$ gives a non-trivial factor of f, if f were reducible.

Claim 10.3 *Let $\tilde{g}, l_k$ be any pair of polynomials satisfying (1). If the coefficients of both f and $\tilde{g}$ are not more than l bits long, and $p^k > (2n)!2^{2nl}$ then* $\gcd(f, \tilde{g})$ *is non-trivial.*

Proof We clearly have $\gcd(f, \tilde{g}) \neq f$ as $\deg(\tilde{g}) < \deg(f)$. Suppose for contradiction that $\gcd(f, \tilde{g}) = 1$ over $\mathbb{Q}[x]$. Then there exists polynomials $u, v \in \mathbb{Z}[x]$ and $N \in \mathbb{Z} - \{0\}$ such that $N < Res(f, \tilde{g})$ and

$$u(x)f(x) + v(x)\tilde{g}(x) = N$$

Recall that the resultant of 2 polynomials of degree atmost n and coefficients not larger than l bits is atmost $(2n)!2^{2nl}$. (see Lecture 4). Now, using the fact that $p^k > (2n)!2^{2nl}$, we have $N \neq 0(\mod p)$. Thus reducing the above equation modulo p^k, we have

$$\begin{aligned} N &= u(x)f(x) + v(x)\tilde{g}(x)(\mod p^k) \\ &= u(x)g_k(x)h_k(x) + v(x)g_k(x)l_k(x)(\mod p^k) \\ &= g_k(x)(u(x)h_k(x) + v(x)l_k(x))(\mod p^k) \end{aligned}$$

But this is a contradiction. As $g_k(x)$ is a polynomial of degree greater than 1. On the RHS, we have either the constant 0 or polynomial in x depending on whether $u(x)h_k(x) + v(x)l_k(x)$ is identically equal to 0 or not. However on the LHS we have a non-zero constant N. Thus, we should have $\gcd(f, \tilde{g}) \neq 1$.■

Thus step 4 computes a non-trivial factor of f if the coefficients of $\tilde{g}$ are not very large. We know from Claim 1 that the coefficients of any factor g of f are not very large. Thus while solving the linear system in step 3(b) of the algorithm, we should obtain a $\tilde{g}$ with small sized coefficients. It is not necessary that we obtain the $\tilde{g}$ with the smallest coefficients, but one close enough would do. Let us look at the linear system once again.

Suppose g_k has degree $d_0 < n(= \deg(f))$. Let

$$\tilde{g}(x) = \sum_{i=0}^{d} c_i x^i \text{ where } d = n - 1, c_i \in \mathbb{Z}$$

$$l_k(x) = \sum_{i=0}^{d-d_0} \alpha_i x^i, \alpha_i \in \mathbb{Z}$$

The linear system is then given by the equation:

$$\sum_{i=0}^{d} c_i x^i = \sum_{i=0}^{d-d_0} \alpha_i g_k(x) x^i + \sum_{i=0}^{d} \beta_i p^k x^i, \beta_i \in \mathbb{Z} \text{ where } c_i, \alpha_i, \beta_i \text{ are the unknowns.}$$

Obtaining a $\tilde{g}$ with small coefficients thus reduces to the following problem

Given $b_1 \ldots, b_m \in \mathbb{Z}^n$, find $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}$, such that the vector $\sum \gamma_i b_i$ has small norm

10.3 Small Vectors in a Lattice

For any vectors $b_1, \ldots b_n \in \mathbb{Z}^n$, let lattice $L(b_1, \ldots b_n) = \{\sum \alpha_i b_i | \alpha_i \in \mathbb{Z}, 1 \leq i \leq n\}$. By small we mean small with respect to any suitable L_p norm[10]. Given a set of vectors $b_1, \ldots, b_m \in \mathbb{Z}^n$, it is sufficient if we look at a linearly independent subset of the vectors which also spans the lattice. We can obtain such a subset by the following reductions.

Let each vector b_i be given by the column vector $(b_{i1}, \ldots, b_{in})^T$. Consider the matrix given by $B = (b_1, \ldots, b_m)$ (ie.,

$$B = \begin{pmatrix} b_{11} & b_{21} & \ldots & b_{m1} \\ b_{12} & b_{22} & \ldots & b_{m2} \\ \vdots & \vdots & & \vdots \\ b_{1n} & b_{2n} & \ldots & b_{mn} \end{pmatrix}$$

[10] L_p norm for a vector $\overline{x} = (x_1, \ldots, x_n)$ is given by $||\overline{x}||_p = (\sum x_i^p)^{\frac{1}{p}}$

1. (a) Compute $g = \gcd(b_{11}, b_{21}, \ldots, b_{m1})$ and integers $a_{11}, a_{21}, \ldots, a_{m1}$ such that $\sum_{i=1}^{m} a_{i1} b_{i1} = g$
   (b) Construct a new basis $B' = (b'_1, \ldots, b'_m)$ as follows

$$b'_1 = \sum_{i=1}^{m} a_{i1} b_i$$
$$b'_k = b_k - \frac{b_{k1}}{g} b'_1, k \neq 1$$

   [Note that B' also spans $L(b_1, \ldots, b_m)$ and that $b'_{11} = g$ and $b'_{k1} = 0, k \neq 1$]

2. Repeat step 1 for $\tilde{B}$ given by

$$\tilde{B} = \begin{pmatrix} b'_{22} & \ldots & b'_{m2} \\ \vdots & & \vdots \\ b'_{2n} & \ldots & b'_{mn} \end{pmatrix}$$

   Repeat this process till B is reduced to a matrix of the form $\left(\frac{A}{C}\ O\right)$ where A is a $m' \times m'$ lower triangular matrix ($\in \mathbb{Z}_{m' \times m'}$), C is a $(n - m') \times m'$ matrix ($\in \mathbb{Z}_{(n-m') \times m'}$) and O is the $n \times (m - m')$ zero matrix.

Clearly the basis $\overline{B} = \left(\frac{A}{C}\right)$ also spans $L(b_1, \ldots, b_m)$ and it is also a linearly independent set. With this the problem of finding a small vector reduces to:

Given a linearly independent set of vectors $\{b_1 \ldots, b_m\} \in \mathbb{Z}^n$, find a small vector in $L(b_1, \ldots b_n)$

For any lattice $L = L(b_1, \ldots, b_n)$ in $\mathbb{Z}^n$, the determinant of the lattice L (denoted as $\det(L)$) is defined as:

$$\det(L) = \det \begin{pmatrix} | & | & & | \\ b_1 & b_2 & \ldots & b_n \\ | & | & & | \end{pmatrix}$$

Though this definition seems to be dependent of the choice of the basis, we shall show that $\det(L)$ (modulo $\{\pm 1\}$) is independent of the basis.

Claim 10.4 *For any two bases $B = (b_1, \ldots, b_n)$, $B' = (b'_1, \ldots, b'_n)$ of a lattice L,* $\det(B) = \pm \det(B')$.

Proof As B is a basis for the lattice L and $b'_i \in L$, there exists a $n \times n$ matrix $A \in \mathbb{Z}_{n \times n}$ such that $B' = BA$. Similarly there exists a $A' \in \mathbb{Z}_{n \times n}$ such that $B = B'A'$. hence we have

$$\begin{aligned} & AA' = I \\ \Rightarrow\quad & \det(AA') = 1 \\ \Rightarrow\quad & \det(A)\det(A') = 1 \\ \Rightarrow\quad & \det(A) = \det(A') = \pm 1 \text{ since } A, A' \in \mathbb{Z}_{n \times n} \\ \Rightarrow\quad & \det(B) = \det(B'A') = \det(B')\det(A') = \pm \det(B') \blacksquare \end{aligned}$$

To obtain a small vector, we shall make a series of moves from one basis to another.(In fact the moves we adopt to do so leave the sign of the determinant unaltered) We move from one basis to another by a sequence of (i, j, q) moves. An (i, j, q) move from a basis $(b_1, \ldots, b_n)$ to $(b'_1, \ldots, b'_n)$ is defined as follows:

$$b'_i = b_i - q b_j$$
$$b'_k = b_k, k \neq i$$

We choose q such that the b'_i is the smallest such vector. For the case $n = 2$, a sequence of such moves leads us to a basis (a, b) where $||a|| \leq ||b|| \leq ||b + qa||$ for all $q \in \mathbb{Z}$. In fact for this case with L_2 norm, a is indeed the smallest vector in the lattice.

Claim 10.5 *If $a, b \in \mathbb{Z}^2$ and $||a||_2 \leq ||b||_2 \leq ||b + qa||_2$ for all $q \in \mathbb{Z}$, then for all $z \in L(a,b) - \{(0,0)\}$, $||a||_2 \leq ||z||_2$.*

Proof Let $z = ua + vb$ for some $u, v \in \mathbb{Z}$ (u, v not both 0). If either u or v is 0, then by hypothesis $||z||_2 \geq ||a||_2$. If both $u \neq 0$ and $v \neq 0$

Case $(i) : u > v$

We first observe that

$$
\begin{aligned}
(a+b)\cdot(a+b) &\geq b \cdot b \\
\Rightarrow 2a \cdot b &\geq -a \cdot a \\
||z||_2^2 &= (u^2 a \cdot a + v^2 b \cdot b + 2uva \cdot b) \\
&\geq (u^2 a \cdot a + v^2 b \cdot b - uva \cdot a) \\
&\geq (u(u-v)a \cdot a + v^2 b \cdot b) \\
&\geq (u(u-v)a \cdot a) \\
&\geq ||a||_2^2
\end{aligned}
$$

Case $(i) : u \leq v$

We now observe that

$$
\begin{aligned}
(a+b)\cdot(a+b) &\geq a \cdot a \\
\Rightarrow 2a \cdot b &\geq -b \cdot b \\
||z||_2^2 &\geq (u^2 a \cdot a + v(v-u)b \cdot b) \\
&\geq (u^2 a \cdot a) \\
&\geq ||a||_2^2 \qquad \blacksquare
\end{aligned}
$$

For the case $n > 2$, we employ the LLL basis reduction to find a small vector. The LLL basis reduction will be discussed in the next lecture.

6.966 Algebra and Computation	October, 21 1998
Lecture 11	
Lecturer: Madhu Sudan	*Scribe: Adam Klivans*

11.1 Introduction

In the last lecture, we saw how finding a short vector in a lattice plays an important part in a polynomial time algorithm for factoring polynomials with rational coefficients. Here, we present the well known basis reduction algorithm of Lenstra, Lenstra, and Lovasz. We can use this algorithm to find a relatively short vector in a lattice. Many of the details in this set of notes are based on L. Lovasz's book "An Algorithmic Theory of Numbers, Graphs, and Convexity" and R. Kannan's "Algorithmic Geometry of Numbers." Most details can be also be read from the original source, i.e., the paper "Factoring polynomials with rational coefficients" by A. K. Lenstra, H. W. Lenstra Jr. and L. Lovasz.

11.1.1 Preliminaries

We would like to point out that without loss of generality, all of our proofs will concern lattices that are specified by a set of basis vectors with integral entries (A lattice specified by basis vectors with rational entries can be transformed into one with integral entries via multiplication by the least common multiple).

11.2 Orthogonalization

The LLL algorithm makes use of some important relationships between orthogonal bases and short vectors. To begin, we review the Gram-Schmidt orthogonalization procedure. This algorithm takes in as input a set of basis vectors for R^n and produces an orthogonal basis. The algorithm works by iterating over all basis vectors and, at step i, takes basis vector i and subtracts from basis vector i its projection onto the already constructed orthogonal basis vectors.

GS-Orthogonalization Algorithm

Input: A set of basis vectors $(a_1, \ldots, a_n)$ for R^n.
Output: A set of orthogonal basis vectors $(a_1^*, \ldots, a_n^*)$ for R^n.

1 Let $a_1^* = a_1$

2 Compute $a_i^* = a_i - \sum_{j=1}^{i-1} \frac{(a_i, a_j^*)}{\|a_j^*\|^2} a_j^*$.

(Note we repeat step (2) for all $1 \leq i \leq n$.)
Furthermore, by expanding the resulting expressions, we can write $a_i = \sum_{j=1}^{i} \mu_{ij} a_j^*$.

Note that in the above algorithm, we started with a basis for R^n and retrieved a new basis. For the rest of this lecture, we will take a basis for a lattice and retrieve its orthogonalization. This orthogonalization is not necessarily a basis for the lattice. Given a basis we can, however, still refer to its orthogonalization as the output of the Gram-Schmidt process. Now that we have seen how to compute an orthogonalization from a set of basis vectors, we show what relationship the orthogonal vectors have to the shortest vector in the lattice. The following important lemma (that we will use later) shows that the size of the shortest vector in an orthogonalization computed via the Gram-Schmidt procedure is always a lower bound for the shortest vector in the lattice.

Lemma 11.1 *Let $(b_1, \ldots, b_n)$ be a basis for a lattice L and let $(b_1^*, \ldots, b_n^*)$ be the vectors obtained from its orthogonalization. Then for any $b \in L$, $||b|| \geq \min_i\{||b_i^*||\}$.*

Proof: Let $b \in L$. Then $b = \sum_{i=1}^{k} \lambda_i b_i$ where $1 \leq k \leq n$ and $\lambda_i \neq 0$. Then we can substitute in for each b_i its orthogonal representation. I.e., $b_i = \sum_{j=1}^{i} \mu_{ij} b_j^*$. So, we can rewrite $b = \sum_{i=1}^{k} \lambda_i' b_i^*$ and since $u_{ii} = 1$ for all i, $\lambda_k' = \lambda_k$, a non-zero integer. So

$$||b||^2 = \sum_{i=1}^{k} ||\lambda_i'|^2 ||b_i^*||^2 \geq |\lambda_k|^2 ||b_i^*||^2 \geq ||b_k^*||^2$$

□

11.2.1 Gauss' Reduction Algorithm

The LLL algorithm can be viewed as a generalization of a simple basis reduction algorithm in two dimensions due to Gauss. The idea is, given u and v, try to subtract off as much of u's projection onto v from u as possible. It is a discrete form of an orthogonalization. We formalize this as follows:

Lemma 11.2 *Given two lattice "basis" vectors u and v we can in polynomial time output a new lattice basis u' and v' such that the acute angle between u' and v' is at least 60 degrees.*

Proof: The algorithm is simple: suppose without loss of generality that $||u|| \leq ||v||$, and replace v with v' equal to the shortest vector of the form $v - mu$ for some $m \in \mathbb{Z}$. Notice that choosing the nearest integer to $\frac{v \cdot u}{u \cdot u}$ suffices. Also notice the length of the component of v' in the direction of u, i.e., $\frac{v \cdot u}{u \cdot u}u$, is less than $||\frac{1}{2}u||$. Now if $||v'|| \geq ||u||$ stop. Otherwise swap u and v' and repeat.
□

Note that by modifying Gauss' algorithm to stop only if $||v'|| \geq (1-\epsilon)||u||$ then the length of the shortest basis vector decreases by a factor of at least $(1-\epsilon)$ after each iteration.

11.3 The LLL Algorithm

Before giving the LLL algorithm, we state formally what it means for a basis to be reduced. The definition may seem somewhat arbitrary, but it will be useful for finding a short vector in a lattice.

Definition 11.3 *We say that a basis $(b_1, \ldots, b_n)$ is reduced if each μ_{ij} corresponding to its orthogonalization is less than $\frac{1}{2}$ and, for each i, $||b_i^*||^2 \leq \frac{4}{3}||b_{i+1}(i)||^2$ where $b_{i+1}(i)$ is the component of b_{i+1} orthogonal to $b_1, \ldots, b_{i-1}$.*

Here is an outline of the LLL basis reduction algorithm: The algorithm keeps a list of the vectors corresponding to our candidate short basis. Initially the list is simply our input vectors. We will sort the vectors in increasing length. Then, we will subtract small vectors from large vectors (a la Lemma 11.2) until one vector's magnitude decreases significantly and should be moved around in the list in order to maintain our length increasing invariant. We then repeat these two steps. For clarity, we separate the two steps and describe them in more detail.

11.3.1 Step 1– Applying Gauss' Reduction

Lemma 11.4 *Given a basis $(b_1, \ldots, b_n)$ with orthogonalization $(b_1^*, \ldots, b_n^*)$ we can, in polynomial time, find a new basis $(b_1', \ldots, b_n')$ where $b_i' = \sum_{j=1}^{i} \mu_{ij}' b_j^*$ with $|\mu_{ij}'| \leq \frac{1}{2}$.*

Proof and Algorithm for Step 1: Assume that $(b_1, \ldots, b_n)$ does not have the desired property. Then there must be indices ij such that $|\mu_{ij}| \geq \frac{1}{2}$. Choose the indices ij where j is maximized. Let m be the closest integer to $|\mu_{ij}|$. Replace b_i with $b_i' = b_i - mb_j$. It should be clear that in our updated basis $(b_1, \ldots, b_{i-1}, b_i', \ldots, b_n)$, μ_{ij} (now μ_{ij}') is less than $\frac{1}{2}$ as we have performed one iteration of Gauss' algorithm above. Repeating this at most $O(n^2)$ times (one time for each pair i, j) we will achieve the desired basis.
□

11.3.2 Step 2– Swapping Vectors

Find an index i such that such that $||b_i^*||^2 > \frac{4}{3}||b_{i+1}(i)||^2$ and swap b_i with b_{i+1}.

11.3.3 Proof of Termination

Certainly if we cannot carry out step 1 or step 2 we have found a reduced basis. We now prove that we only need to repeat steps 1 and 2 a polynomial number of times.

Proof of termination: Consider the quantity $D(b_1, \ldots, b_n) = \prod_{i=1}^{n} ||b_i^*||^{n-i}$. We will show that Step 1 does not change the quantity $D(b_1, \ldots, b_n)$ and that Step 2 only decreases the quantity $D(b_1, \ldots, b_n)$. Since initially $D(b_1, \ldots, b_n)$ is at most exponentially large in the size of the largest vector,and $D(b_1, \ldots, b_n)$ is always larger than 1, we can only decrease $D(b_1, \ldots, b_n)$ by a constant factor a polynomial number of times. This will prove that our algorithm terminates in polynomial time.

- Clearly Step 1 leaves the quantity $D(b_1, \ldots, b_n)$ unchanged.
- In Step 2, we are putting a bigger lattice vector farther down our list. Thus the quantity $D(b_1, \ldots, b_n)$ must be decreasing by at least a factor of $\frac{2}{\sqrt{3}}$.

As a crude upper bound we see that $D(b_1, \ldots, b_n) \leq (\max_i ||b_i||)^{n^2}$.

Now we need to show $D(b_1, \ldots, b_n)$ is always greater than 1. Let C_k be the $k \times k$ matrix whose ith column is b_i and R_k matrix be the matrix whose ith row is b_i. Notice that for any k, $det(R_k C_k) = det(R_k)det(C_k) \geq 1$ (recall our basis vectors have integer entries, and $\det(A) = \det(B)$ for any bases A and B). But, $det(R_k C_k) = ||b_1^*||^2 \cdots ||b_k^*||^2$ since the determinant of the matrix consisting of orthogonal basis vectors is the product of the norms of the vectors. Thus, $D(b_1, \ldots, b_n) \geq 1$.

Since $D(b_1, \ldots, b_n)$ is only a single exponential in the size of the input and is always greater than 1, we can only repeat steps 1 and 2 a polynomial number of times.

□

11.3.4 Retrieving a Short Vector

Here we show how finding a short basis immediately yields a relatively "short" vector.

Theorem 11.5 *Let $(b_1, \ldots, b_n)$ be a reduced basis of the lattice L. Then $||b_1|| \leq 2^{\frac{n-1}{2}}\lambda(L)$ where $\lambda(L)$ is the length of the shortest vector in the lattice.*

Proof: (taken directly from Lovasz's An Algorithmic Theory of Numbers, Graphs, and Convexity)

$$\begin{aligned} ||b_i^*||^2 &= ||b_i(i)||^2 &\leq \frac{4}{3}||b_{i+1}(i)||^2 \\ & &= \frac{4}{3}||b_{i+1}^* + \mu_{i+1,i}||b_i^*||^2 \\ & &= \frac{4}{3}||b_{i+1}^*||^2 \frac{4}{3}\mu_{i+1,i}^2||b_i^*||^2 \\ & &\leq \frac{4}{3}||b_{i+1}^*||^2 + \frac{1}{3}||b_i^*||^2 \end{aligned}$$

Thus, $||b_{i+1}^*||^2 \geq \frac{1}{2}||b_i^*||^2$ and inductively we see that $||b_i^*||^2 \geq 2^{1-i}||b_1^*||^2 = 2^{1-i}||b_1||^2$. By Lemma 11.1, $||b_1||^2 \leq \min_i\{2^{i-1}||b_i^*||^2\} \leq 2^{n-1}\min_i ||b_i^*||^2 \leq 2^{n-1}\lambda(L)^2$ which proves the theorem.

□

11.4 Root Finding and Factorization

In this course we've covered a number of different factorization algorithms. One category of factorization algorithms we don't cover is factorization over the reals. Here we briefly mention one important concept that arises in this study: namely, the method of Sturm sequences. We illustrate this concept to describe its role in (approximately) finding real roots of a real polynomial. ' Observe that in many cases, root-finding seems much easier than factorization. For example, if we have a bivariate polynomial $f(x, y)$ and we know that $y - p(x)$ divides $f(x, y)$ for some polynomial $p(x)$, then we know that for any polynomial $r(x)$, $f(x, r(x))$ will have $r(x) - p(x)$ as an irreducible factor.

11.4.1 Sturm sequences

Sturm sequences are used to determine the number of real roots that a given polynomial over the reals has in some interval (a, b). Given f, we compute f', its derivative, and we define a sequence of polynomials h_i inductively as follows:

$$h_{i-2} = q_{i-1} h_{i-1} - h_i$$

Now let $Var_a(h_0(a), \ldots, h_n(a))$ = the number of sign changes of this sequence. The interesting fact about these polynomials is captured in the following theorem:

Theorem 11.6 *For a given polynomial f, the number of distinct zero locations in an interval (a, b) is equal to $Var_a - Var_b$.*

One can find a proof for this in B. Mishra's book "Algorithmic Algebra" published by Springer-Verlag.

6.966 Algebra and Computation	October 19, 1998
Lecture 12	
Lecturer: Madhu Sudan	*Scribe: Rocco Servedio*

12.1 Introduction

This was the first lecture on the second main course topic, the complexity of solving multivariate polynomial equations. In this lecture we outlined the general family of problems we will be interested in and began to discuss the ideal membership problem. We gave an algorithm for "dividing" a fixed polynomial by several polynomials and introduced the notion of a Groebner basis. A useful reference for this material is the book "Ideals, Varieties and Algorithms" by Cox, Little and O'Shea.

12.2 The general question

In its most general form, the question we will be concerned with for this portion of the course is the following: let $f_1(x_1, \ldots, x_n), \ldots, f_s(x_1, \ldots, x_n)$ be a collection of s polynomials, where for each polynomial the degree in x_i is bounded by d_i. Let $Q_1, \ldots, Q_n$ be some sequence of quantifiers, i.e. each Q_i is either "$\exists$" or "$\forall$." What is the computational complexity of determining the truth value of the following statement:

$$\text{“}Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \text{ such that } f_1(x_1, \ldots, x_n) = \cdots = f_s(x_1, \ldots, x_n) = 0\text{”?}$$

A moment's thought shows that this is a very general question (and that the computational complexity can be quite high). If we allow an arbitrary sequence of quantifiers, then we can express PSPACE-hard problems in the above format. We noted back in Lecture 1 that even if all quantifiers are restricted to be existential and $d_i = 2$, we can still express the NP-hard problem 3SAT. Similarly, if we take all quantifiers to be "$\forall$" we can capture co-NP; and by bounding the number of alternations between "$\exists$" and "$\forall$" we obtain the different levels of the polynomial hierarchy.

Today we will focus on the "NP" version of this problem in which all quantifiers are existential. More specifically, we consider the following question: let K be a field and let $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ be polynomials each of degree at most d_i in x_i. Does there exist $(a_1, \ldots, a_n) \in K^n$ such that

$$f_1(a_1, \ldots, a_n) = \cdots = f_s(a_1, \ldots, a_n) = 0 \text{ ?}$$

12.3 Varieties and Ideals

The question we have asked leads naturally to the following definition:

Definition 12.1 *Let $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$. The* variety *of $(f_1, \ldots, f_s)$ is the set*

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in K^n \mid \forall i \in \{1, \ldots, s\} f_i(a_1, \ldots, a_n) = 0\}.$$

So the variety of a set of polynomials is the set of their common zeros. The question we are interested in could thus be succinctly written as "is $V(f_1, \ldots, f_s)$ nonempty?" This problem has an NP flavor to it because if the answer is yes, a short proof can be given which can be easily verified – the proof is simply a point $(a_1, \ldots, a_n) \in K^n$, and verification consists of checking that it is a zero for each polynomial f_i.

Now let $f_1, f_2 \in K[x_1, \ldots, x_n]$, and suppose that $(a_1, \ldots, a_n)$ is such that $f_1(a_1, \ldots, a_n) = f_2(a_1, \ldots, a_n) = 0$. Then it is clear that for any $q_1, q_2 \in K[x_1, \ldots, x_n]$, we will have

$$q_1(a_1, \ldots, a_n) f_1(a_1, \ldots, a_n) + q_2(a_1, \ldots, a_n) f_2(a_1, \ldots, a_n) = 0.$$

This observation motivates the following definition:

Definition 12.2 *Let $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$. The* ideal generated by $f_1, \ldots, f_s$ *is the following set of polynomials:*

$$I(f_1, \ldots, f_s) = \{\sum_{i=1}^{s} q_i(x_1, \ldots, x_n) f_i(x_1, \ldots, x_n) \mid \forall i \in \{1, \ldots, s\}\ q_i \in K[x_1, \ldots, x_n]\}.$$

So the ideal generated by a nontrivial set of polynomials is an infinite set of polynomials which includes the original set. Note that if $f_1, \ldots, f_s$ all vanish at a point $(a_1, \ldots, a_n)$, then so does every polynomial in $I(f_1, \ldots, f_s)$. This leads to the following simple claim:

Claim 12.3 *If $1 \in I(f_1, \ldots, f_s)$, then $\nexists (a_1, \ldots, a_n) \in K^n$ such that $\forall i\ f_i(a_1, \ldots, a_n) = 0$.*

Proof Suppose to the contrary that $(a_1, \ldots, a_n) \in K^n$ is a zero for each f_i. Since $1 \in I(f_1, \ldots, f_s)$, there exist $q_i \in K[x_1, \ldots, x_n]$ such that $1 = \sum_{i=1}^{s} q_i f_i$; evaluating at the point $(a_1, \ldots, a_n)$, we obtain

$$1 = \sum_{i=1}^{s} q_i(a_1, \ldots, a_n) f_i(a_1, \ldots, a_n) = \sum_{i=1}^{s} 0 = 0$$

which is a contradiction. ■

An equivalent form of Claim 3 is the following: if $1 \in I(f_1, \ldots, f_s)$, then for any $(a_1, \ldots, a_n) \in K^n$, there is some i such that $f_i(a_1, \ldots, a_n) \neq 0$. Later in the course we will prove the following partial converse, which is a weak version of Hilbert's Nullstellensatz ("null space theorem"):

Fact 12.4 *If K is an algebraically closed field and $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$, then*

$$1 \in I(f_1, \ldots, f_s) \iff \forall (a_1, \ldots, a_n) \in K^n\ \exists j \in \{1, \ldots, s\} \text{ such that } f_j(a_1, \ldots, a_n) \neq 0.$$

This means that over algebraically closed fields,

$$\exists q_1, \ldots, q_s \in K[x_1, \ldots, x_n]\ \sum_{i=1}^{s} q_i f_i = 1 \iff \forall (a_1, \ldots, a_n) \in K^n\ \exists j \in \{1, \ldots, s\}\ f_j(a_1, \ldots, a_n) \neq 0.$$

At first glance this equivalence seems to relate NP- and co-NP-type problems. However, as we will see in later lectures, the degrees of the polynomials q_i can be doubly exponential in n; hence even if the answer is "yes" there may not be a short proof which can be verified quickly.

12.4 The Ideal Membership Problem

An instance of the *ideal membership problem* consists of a distinguished polynomial $f \in K[x_1, \ldots, x_n]$ and a finite set of polynomials $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$. The problem is to determine whether $f \in I(f_1, \ldots, f_s)$. Since ideals are infinite objects, this can be viewed as an effective method of computing the ideal generated by $f_1, \ldots, f_s$. The discussion at the end of the last section shows that this is an interesting question even when $f = 1$.

Recall that in the simple case when $s = 1$ and f, f_1 are univariate, we can solve the ideal membership problem by simply diving f by f_1 to obtain a quotient q and a remainder r; we have that $f(x) = q(x) f_1(x) + r(x)$, and $f \in I(f_1)$ if and only if $r(x) = 0$. In the more complicated scenario where f, f_i are multivariate polynomials, we would like to take the same approach and divide f by $(f_1, \ldots, f_s)$ by expressing f as $f = \sum_{i=1}^{s} q_i f_i + r$ where r is a remainder term which we try to make as small as possible. If we can achieve $r = 0$ then f belongs to $I(f_1, \ldots, f_s)$.

However, in order to make this precise we need a criterion for deciding which of two possible remainders is "smaller." For instance, if $f = x^2$ and $f_1 = x^2 - y^2$, we could express f as $0 \cdot f_1 + (x^2)$, in which case the remainder would be x^2, or we could express f as $1 \cdot f_1 + (y^2)$, in which case the remainder would be y^2.

In the next section we define an ordering on monomials in $K[x_1, \ldots, x_n]$; this will enable us to meaningfully discuss the "size" of a remainder. Then we will give an algorithm for "dividing" f by several polynomials $f_1, \ldots, f_s$ and computing the remainder. As we will see, though, this division algorithm will not have all of the properties we want. Ultimately we will develop the machinery of Groebner bases to come up with an effective (but not efficient) algorithm to solve the ideal membership problem.

12.5 Orderings on Monomials

For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$ we write x^α to denote the monomial $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$.

Definition 12.5 *A relation $\succeq$ on the set of monomials in $K[x_1, \ldots, x_n]$ is an* admissible monomial ordering *if it satisfies the following properties:*

1. *$\succeq$ is a total ordering, i.e. for any $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$, either $\alpha \succ \beta, \alpha = \beta$, or $\alpha \prec \beta$.*
2. *for all $\alpha, \beta, \gamma \in \mathbb{Z}^n_{\geq 0}$, if $x^\alpha \succeq x^\beta$ then $x^{\alpha+\gamma} \succeq x^{\beta+\gamma}$.*
3. *for all $\alpha \in \mathbb{Z}^n_{\geq 0}$, it holds that $x^\alpha \succeq 1$.*

We will prove the following fact in Section 12.8

Fact 12.6 *Let $\succeq$ be an admissible monomial ordering. Then any sequence of monomials which is strictly decreasing under $\succeq$, i.e.*

$$x^{\alpha(1)} \succ x^{\alpha(2)} \succ x^{\alpha(3)} \succ \cdots$$

must be finite.

Here are some examples of admissible monomial orderings:

Example 1: lexcographic (lex) ordering. To compare x^α and x^β under lex ordering, let j be such that $\alpha_i = \beta_i$ for $0 \leq i < j$ and $\alpha_j \neq \beta_j$. Then if $\alpha_j > \beta_j$ we have $x^\alpha \succ x^\beta$, and if $\alpha_j < \beta_j$ we have $x^\alpha \prec x^\beta$.

We note that under lex ordering, the "earlier" variables $x_1, x_2, \ldots$ are more important than the "later" variables which have higher indices. As a result, lex ordering is useful in applications such as the following: let $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$ and suppose $m < n$. Consider the following statement:

$$\text{“}\exists x_1, \ldots, x_m \; f_1(x_1, \ldots, x_n) = \cdots = f_s(x_1, \ldots, x_n) = 0.\text{”}$$

The truth value of this statement will be a function of $x_{m+1}, \ldots, x_n$; we would like to find this function. As we will see later in the course, we will eliminate the variables $x_1, \ldots, x_m$ to obtain new functions $g_1(x_{m+1}, \ldots, x_n), \ldots, g_t(x_{m+1}, \ldots, x_n)$ which depend only on the variables $x_{m+1}, \ldots, x_n$. The lexicographic ordering with $x_i \succ x_j$ for $i > j$ will facilitate this elimination.

Example 2: graded lexicographic (grlex) ordering. We write $|\alpha|$ to denote $\sum_{i=1}^n \alpha_i$. Graded lexicographic ordering ranks monomials by their total degree and breaks ties by using lexicographic ordering; so if $|\alpha| > |\beta|$ then $x^\alpha \succ x^\beta$. Otherwise, if $|\alpha| = |\beta|$, the ranking of x^α and x^β is determined by the lexicographic ordering of x^α and x^β.

Revisiting our earlier example where $f = x^2$ and $f_1 = x^2 - y^2$, we see that under the lexicographic ordering on monomials over $\{x, y\}$ with $x \succ y$, the remainder y^2 is smaller than x^2. In the next section we will further refine our intuitive notion of what constitutes a good remainder.

12.6 Division by several polynomials

Example 3: Let $f = x^2y + x + y^2$, let $f_1 = x^2$, let $f_2 = y^2 - 3$, and let $\succeq$ denote lex order on monomials over $\{x, y\}$ with $x \succ y$. One attempt to divide f by f_1 and f_2 would be to express f as $y \cdot f_1 + (x + y^2)$, leaving a remainder of $x + y^2$. Since any remainder must contain x, this expression has succeeded in minimizing the highest-degree term; yet this does not seem like the best possible remainder because it still contains a y^2 term. A better remainder can be obtained by writing $f = y \cdot f_1 + 1 \cdot f_2 + (x + 3)$. This remainder is such that no term in the remainder is divisible by the leading term of any polynomial which we are dividing by.

Example 4: Let $f = x^2y + xy^2 + y^2$, let $f_1 = xy - 1$, and let $f_2 = y^2 - 1$. If we first extract the largest possible multiple of f_1 and then extract the largest possible multiple of f_2 from what's left, we obtain $f = (x + y) \cdot f_1 + 1 \cdot f_2 + (x + y + 1)$, yielding a remainder of $x + y + 1$. On the other hand, if we were to first extract the largest multiple of f_2 and then f_1, we would get $f = (x+1)\cdot f_2 + x\cdot f_1 + (2x+1)$ for a remainder of $2x+1$. Yet another remainder can be obtained by writing $f = (2y+x)\cdot f_1 + (-x+1)\cdot f_2 + (2y+1)$.

In order to state our division algorithm, we require the following definitions. Let $\succeq$ be an admissible monomial ordering over the variables $x_1, \ldots, x_n$ and let $f \in K[x_1, \ldots, x_n]$.

- The *leading monomial* of f, written $LM(f)$, is the monomial which is ranked highest under $\succeq$ of all monomials which have nonzero coefficients in f.

- The *leading coefficient* of f, written $LC(f)$, is the coefficient of $LM(f)$ in f.

- The *leading term* of f, written $LT(f)$, is $LC(f)\cdot LM(f)$. Note that if f is a nonconstant polynomial, then $LM(f) \succ LM(f - LT(f))$.

- The *degree* of f, written $\deg(f)$, is α such that $x^\alpha = LM(f)$.

The algorithm to divide a polynomial f by an ordered s-tuple of polynomials $f_1, \ldots, f_s$ is as follows. The initial invocation should be $\textsc{Divide}(f; 0; f_1, \ldots, f_s; 0, \ldots, 0)$, and the return value will be $(r; q_1, \ldots, q_s)$ such that $f = q_1 f_1 + \cdots + q_s f_s + r$.

$\texttt{Divide}(f; r; f_1, \ldots, f_s; q_1, \ldots, q_s)$
1. IF $f = 0$ THEN RETURN $(r; q_1, \ldots, q_s)$.
2. LET i BE LEAST INDEX SUCH THAT $LM(f_i)$ DIVIDES $LM(f)$.
3. IF SUCH AN i EXISTS, LET $q_i' = LT(f)/LT(f_i)$ AND RETURN

$$\texttt{Divide}(f - q_i f_i; r; f_1, \ldots, f_s; q_1, \ldots, q_{i-1}, q_i + q_i', q_{i+1}, \ldots, q_s).$$

4. IF NO SUCH i EXISTS, RETURN $\texttt{Divide}(f - LT(f), r + LT(f), f_1, \ldots, f_s, q_1, \ldots, q_s)$.

We can immediately make several observations about the algorithm. Perhaps the most important properties of the algorithm, both of which can be easily verified, are the following: if the algorithm outputs $(r; q_1, \ldots, q_s)$, then the identity $f = q_1 f_1 + \cdots + q_s f_s + r$ must indeed hold, and the remainder polynomial r has the property that no monomial term of r is divisible by any of $LT(f_1), \ldots, LT(f_s)$. We say that a remainder which satisfies this property is *legitimate.*

Another important property is that the algorithm must eventually terminate on any input. To see this, note that the degree of f drops with each invocation of `Divide` within the algorithm. Thus, the sequence of leading monomials of f which successively occur as the algorithm progresses form a strictly decreasing sequence under $\succeq$, and by Fact 12.6 such a sequence can be only finitely long.

Unfortunately, we cannot say that this algorithm runs in polynomial time in any meaningful sense. If each of $f, f_1, \ldots, f_s$ have at most t nonzero coefficients, the algorithm's runtime may exceed any fixed polynomial in s and t. If d_i is the maximum degree in each variable across all the polynomials, however, it can be shown that the run time will be $O(s \cdot \Pi_{i=1}^n (d_i + 1))$. This function grows extremely rapidly; even if each d_i is 1 it will be exponential in n.

Another less-than-desirable property of the algorithm is that it is sensitive to the order of its inputs $f_1, \ldots, f_s$; hence using it to divide f by (f_1, f_2, f_3) can produce different results than dividing f by (f_2, f_1, f_3). Thus, it is possible for two different polynomials r_1, r_2 to both be legitimate remainders.

If in some sense we could divide the polynomial f by the *infinite* collection of polynomials $I(f_1, \ldots, f_s)$, though, then there would be a unique legitimate remainder. To see this, suppose that $f = g_1 + r_1 = g_2 + r_2$ where $g_1, g_2 \in I(f_1, \ldots, f_s)$ and r_1, r_2 are both legitimate remainders, i.e. no term of r_i is divisible by $LT(h)$ for any $h \in I(f_1, \ldots, f_s)$. It follows that no term of $r_2 - r_1$ can be divisible by $LT(h)$ for any h in $I(f_1, \ldots, f_s)$; but since $r_2 - r_1 = g_1 - g_2 \in I(f_1, \ldots, f_s)$, this means that $LT(r_2 - r_1)$ is not divisible by $LT(r_2 - r_1)$, which is a contradiction.

12.7 Groebner bases

Of course, our division algorithm cannot be applied to divide f by an infinite collection of polynomials. We might hope, though, to find a finite collection $g_1, \ldots, g_t$ of polynomials which would be such that using our division algorithm to divide f by $(g_1, \ldots, g_t)$ would somehow give us the remainder that we would like to obtain if we could divide f by $I(f_1, \ldots, f_s)$. It is clear that the leading terms $LT(g_1, \ldots, g_t)$ of such a collection might have to contain more information than the leading terms of the polynomials $f_1, \ldots, f_s$. For example, under lex ordering with $x \succ y \succ z$, the leading terms of the polynomials $f_1 = x^2 - y$ and $f_2 = x^2 - z$ yield no information whatsoever about y and z.

With this motivation, we make the following important definition:

Definition 12.7 *Under a fixed monomial ordering $\succeq$, a finite subset $(g_1, \ldots, g_t)$ of an ideal J forms a* Groebner basis *for J if*

1. *$I(g_1, \ldots, g_t) = J$ and*
2. *$I(LT(g_1), \ldots, LT(g_t)) = I(LT(J))$, where $LT(J) = \{LT(h) : h \in J\}$.*

Returning to the above example, we can immediately see that $\{f_1, f_2\}$ does not constitute a Groebner basis for $I(f_1, f_2)$, since $y \in I(LT(I(f_1, f_2)))$ but $y \notin I(x^2, x^2)$. Later in the course we will prove that given any ideal $J \subset k[x_1, \ldots, x_n]$ and any admissible monomial ordering $\succeq$, there is a Groebner basis for J under $\succeq$. As a corollary we will obtain Hilbert's basis theorem, which states that every ideal has a finite basis. We will also show that our division algorithm, when applied to a Groebner basis, yields an effective solution to the ideal membership problem. (Unfortunately, it will not be the case that our division algorithm applied to Groebner bases yields an efficient solution to the ideal membership problem, but there is strong evidence that no efficient algorithm exists for this problem – Mayr and Meyer have showed that the the ideal membership problem is EXPSPACE-hard.) We will also define a minimal Groebner basis and show that the minimal Groebner basis for an ideal is unique.

12.8 Monomial Ideals

We turn first to the study of monomial ideals. In this section we prove that every monomial ideal is finitely generated; this is a special case of Hilbert's basis theorem.

Definition 12.8 *An ideal $J \subseteq K[x_1, \ldots, x_n]$ is a* monomial ideal *if there is some set A (possibly infinite) of monomials such that*

$$J = \{ \sum_{x^\alpha \in A} q_\alpha x^\alpha \mid q_\alpha \in K[x_1, \ldots, x_n]\}.$$

We can immediately state the following claim:

Claim 12.9 *Let $J = I(x^\alpha : x^\alpha \in A)$ be a monomial ideal. A monomial x^β belongs to J iff x^β is divisible by x^α for some $x^\alpha \in A$.*

Proof Clearly if $x^\beta = x^\alpha \cdot x^\gamma$ for some $x^\alpha \in A$, then $x^\beta \in J$. For the other direction, suppose that $x^\beta \in J$; then $x^\beta = q_{\alpha_1} x^{\alpha(1)} + \cdots + q_{\alpha(s)} x^{\alpha(s)}$ for some $x^{\alpha(1)}, \ldots, x^{\alpha(s)} \in A$. The right hand side is a sum of monomials each of which is divisible by some element of A, and the left hand side must have the same property. ■

As an example, consider the ideal $I(x^2y^5, x^4y^3, x^5y) \subset K[x, y]$. The monomials in I correspond to the integer grid points which fall in the shaded region of Figure 1.

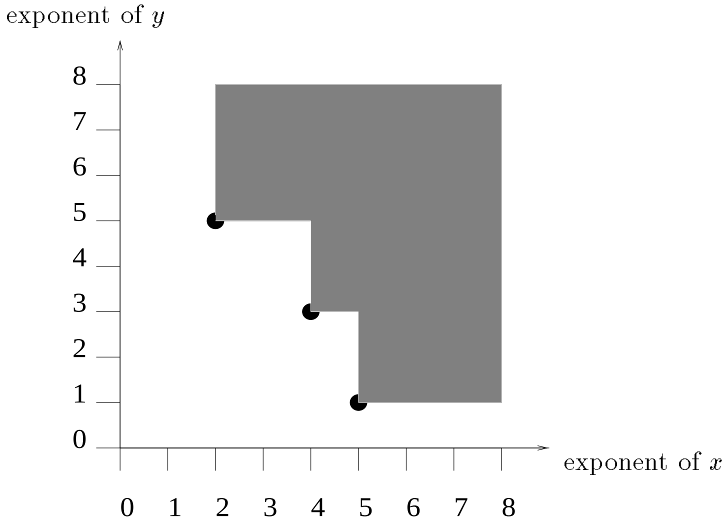


Figure 1

Now we prove that every monomial ideal has a finite basis:

Lemma 12.10 (Dickson's Lemma) *Let $J = I(x^\alpha : x^\alpha \in A) \subseteq K[x_1, \ldots, x_n]$. Then there is some finite set of monomials $\{x^{\alpha(1)}, \ldots, x^{\alpha(s)}\} \subseteq A$ such that $J = I(x^{\alpha(1)}, \ldots, x^{\alpha(s)})$.*

Proof The proof is by induction on n, the number of variables. If $n = 1$, then let $\beta = \min\{\alpha : x^\alpha \in A\}$; we have that x^β divides x^α for all $x^\alpha \in A$, and hence $J = I(x^\beta)$.

Now suppose the claim holds for $1, \ldots, n-1$, and let $J \subseteq K[x_1, \ldots, x_{n-1}, y]$ be a monomial ideal over n variables. We write a monomial in $K[x_1, \ldots, x_{n-1}, y]$ as $x^\alpha y^m$ where $\alpha \in \mathbb{Z}^{n-1}_{\geq 0}$ and $m \in \mathbb{Z}_{\geq 0}$. Let $J' \subseteq K[x_1, \ldots, x_{n-1}]$ be the monomial ideal generated by those monomials x^α which are such that $x^\alpha y^m \in J$ for some $m \geq 0$. Clearly J' is a monomial ideal; by the induction hypothesis, we have that J' is finitely generated, i.e. $J' = I(x^{\alpha(1)}, \ldots, x^{\alpha(s)})$.

Let m be the smallest integer such that $x^{\alpha(i)} y^m \in J$ for $i = 1, \ldots, s$. For each $k \in \{0, \ldots, k-1\}$ let $J'_k \subseteq K[x_1, \ldots, x_{n-1}]$ be the ideal generated by those monomials x^β which are such that $x^\beta y^k \in J$. Using the inductive hypothesis again, each J'_k is finitely generated, i.e. $J'_k = I(x^{\alpha_k(1)}, \ldots, x^{\alpha_k(s_k)})$.

Now consider the following list of monomials over $\{x_1, \ldots, x_{n-1}, y\}$:

$$\begin{array}{rcc}
\text{from } J' & : & x^{\alpha(1)}, \ldots, x^{\alpha(s)} \\
\text{from } J'_0 & : & x^{\alpha_0(1)}, \ldots, x^{\alpha_0(s_0)} \\
 & \vdots & \\
\text{from } J'_k & : & x^{\alpha_k(1)}, \ldots, x^{\alpha_k(s_k)} \\
 & \vdots & \\
\text{from } J'_{m-1} & : & x^{\alpha_{m-1}(1)}, \ldots, x^{\alpha_{m-1}(s_{m-1})}
\end{array}$$

Let A' be the (finite) set of all monomials which occur in this list; we claim that $J = I(A')$. Clearly $I(A') \subseteq J$, we must show that $J \subseteq I(A')$. Let $x^\alpha y^\ell$ be an arbitrary monomial which belongs to J. If $\ell \geq m$, then by our construction of J' we have that $x^\alpha y^\ell$ must be divisible by some $x^{\alpha(i)} y^m$. Otherwise, if $\ell < m$, then $x^\alpha y^\ell$ must be divisible by some $x_k^\alpha(i) y^\ell$ by the construction of J'_ℓ. By Claim 12.9, any monomial in J must also belong to $I(A')$, and it is easy to verify that this implies that $J \subseteq I(A')$, and hence $J = I(A')$.

With a little additional work one can show that the finite set of monomials which generate J can actually be chosen from A (exercise for the reader). ■

As a corollary of Dickson's Lemma we can prove the following fact about admissible monomial orderings, which we used earlier to show that our division algorithm will always run in finite time.

Fact 12.6 *Let $\succeq$ be an admissible monomial ordering. Then any sequence of monomials which is strictly decreasing under $\succeq$, i.e.*

$$x^{\alpha(1)} \succ x^{\alpha(2)} \succ x^{\alpha(3)} \succ \cdots$$

must be finite.

Proof Let $x^{\alpha(1)} \succ x^{\alpha(2)} \succ x^{\alpha(3)} \succ \cdots$ be a sequence of monomials which is strictly decreasing under $\succeq$. Let $A = \{x^{\alpha(1)}, x^{\alpha(2)}, \ldots\}$ and let $J = I(x^{\alpha} : x^{\alpha} \in A)$. Since J is a monomial ideal, Dickson's Lemma implies that for some $x^{\alpha(i_1)} \succ \cdots \succ x^{\alpha(i_s)} \in A$, we have that $J = I(x^{\alpha(i_1)}, \ldots, x^{\alpha(i_s)})$. We claim now that $x^{\alpha(i_s)}$ must be the smallest element of A, and thus that A must be finite. To see this, let x^{α} be any element of A. Since $x^{\alpha} \in J = I(x^{\alpha(i_1)}, \ldots, x^{\alpha(i_s)})$, it follows from Claim 12.9 that $x^{\alpha} = x^{\alpha(i_j)} \cdot x^{\gamma}$ for some $j \in \{1, \ldots, s\}$ and some $\gamma \in \mathbb{Z}^n_{\geq 0}$. Since $\succeq$ is an admissible monomial ordering, though, we have that $x^{\alpha} = x^{\alpha(i_j)} \cdot x^{\gamma} \succeq x^{\alpha(i_j)} \succeq x^{\alpha(i_s)}$ and hence $x^{\alpha(i_s)}$ is indeed the smallest element of A. ■

6.966 Algebra and Computation October 21, 1998

Lecture 13

Lecturer: Madhu Sudan *Scribe: Amit Sahai*

13.1 Introduction

In the last lecture, we discussed a division algorithm for polynomials in many variables. The algorithm, while efficient, had several drawbacks, the primary one being a lack of uniqueness of the remainder when dividing a polynomial f by several polynomials $p_1, \ldots, p_n$. We decided that what we really wanted was a way to "divide" f by the *ideal* $\mathcal{J}$ generated by $p_1, \ldots p_n$ – *i.e.* find the residue of f in the ring modulo $\mathcal{J}$. The tool we introduced for accomplishing this task is the Groebner basis of an ideal. We claimed (and will show in this lecture) that if we divide f by a Groebner basis for the ideal $\mathcal{J}$, then the remainder will have the uniqueness properties we desire.

The main result of this lecture is that every ideal in a polynomial ring over a field in finitely many variables has a finite Groebner basis. A useful reference for this material is the book "Ideals, Varieties and Algorithms" by Cox, Little and O'Shea. We begin by reviewing some definitions we will need from the last lecture:

13.2 Defintions

For $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$ we write x^α to denote the monomial $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$.

Definition 13.1 *A relation $\succeq$ on the set of monomials in $K[x_1, \ldots, x_n]$ is an* admissible monomial ordering *if it satisfies the following properties:*

1. *$\succeq$ is a total ordering, i.e. for any $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$, either $\alpha \succ \beta, \alpha = \beta$, or $\alpha \prec \beta$.*

2. *for all $\alpha, \beta, \gamma \in \mathbb{Z}^n_{\geq 0}$, if $x^\alpha \succeq x^\beta$ then $x^{\alpha+\gamma} \succeq x^{\beta+\gamma}$.*

3. *for all $\alpha \in \mathbb{Z}^n_{\geq 0}$, it holds that $x^\alpha \succeq 1$.*

Let $\succeq$ be an admissible monomial ordering over the variables $x_1, \ldots, x_n$ and let $f \in K[x_1, \ldots, x_n]$.

- The *leading monomial* of f, written $LM(f)$, is the monomial which is ranked highest under $\succeq$ of all monomials which have nonzero coefficients in f.

- The *leading coefficient* of f, written $LC(f)$, is the coefficient of $LM(f)$ in f.

- The *leading term* of f, written $LT(f)$, is $LC(f) \cdot LM(f)$. Note that if f is a nonconstant polynomial, then $LM(f) \succ LM(f - LT(f))$.

- The *degree* of f, written $\deg(f)$, is α such that $x^\alpha = LM(f)$.

Note that since we work in a field, we will often abuse notation by speaking of $LT(f)$ as if it were a monomial. This should not cause concern since any ideal that contains $LT(f)$ must also contain $LM(f)$, and vice versa.

Remainder Property. We first recall the fundamental property of our division algorithm `Divide` is that after dividing a polynomial f by a sequence of polynomials $p_1, \ldots, p_n$, *no term in the remainder r is divisible by any of $LT(p_1), \ldots, LT(p_n)$*. Unfortunately, this property does not suffice to guarantee uniqueness, and indeed can fail dramatically to do "what we want." (*e.g.* consider dividing $y-z$ by $p_1 = x-y$ and $p_2 = x-z$, under the lexicographic ordering where $x > y > z$. Here, $y-z$ is still a remainder satisfying the Remainder Property even though $y - z = p_2 - p_1$.) Intuitively what we wanted was to find a remainder that satisfies the Remainder Property not only for the given polynomials but for all polynomials in the ideal generated by them. This motivates the definition of the notion of a Groebner basis:

Definition 13.2 *Under a fixed monomial ordering $\succeq$, a finite subset $\{g_1, \ldots, g_t\}$ of an ideal $\mathcal{J}$ forms a* Groebner basis *for $\mathcal{J}$ if*

1. *$\mathcal{I}(g_1, \ldots, g_t) = \mathcal{J}$ and*

2. *$\mathcal{I}(LT(g_1), \ldots, LT(g_t)) = \mathcal{I}(LT(\mathcal{J}))$, where $LT(\mathcal{J}) = \{LT(h) : h \in \mathcal{J}\}$.*

13.3 Existence of finite Groebner bases

In this section, we prove that every ideal in a polynomial ring over finitely many variables has a finite Groebner basis. Suppose $\mathcal{J}$ is an ideal in $K[x_1, \ldots, x_n]$. We will build our Groebner basis by first finding a set of polynomials satisfying the second Groebner basis condition, and then conclude that this set must also be a basis for $\mathcal{J}$.

Consider the ideal $\mathcal{L} = \mathcal{I}(LT(\mathcal{J}))$. By Dickson's Lemma, which we proved last time, we know that every monomial ideal – one that can be generated by a set of monomials – has a finite basis of monomials. So in particular this means that $\mathcal{L}$ has a finite basis of monomials $\{m_1, \ldots, m_t\}$. In the last lecture, we showed:

Claim 13.3 *Let h be a monomial and M be a set of monomials in $K[x_1, \ldots, x_n]$. Then $h \in \mathcal{I}(M)$ iff m divides h for some $m \in M$.*

Thus, since $\mathcal{L}$ is generated by the leading terms of polynomials in $\mathcal{J}$, for each m_i there must exist some $g_i \in \mathcal{J}$ such that m_i is divisible by $LT(g_i)$. Hence, we have immediately that

$$\mathcal{I}(LT(g_1), \ldots, LT(g_t)) = \mathcal{I}(m_1, \ldots, m_t) = \mathcal{L} = \mathcal{I}(LT(\mathcal{J})).$$

All that remains in showing that $\{g_1, \ldots, g_t\}$ is a Groebner basis for $\mathcal{J}$ is to show that it does actually generate all of $\mathcal{J}$. Suppose $f \in \mathcal{J}$. Consider what happens when we use our Division Algorithm to divide f by $g_1, \ldots, g_t$. Suppose the algorithm outputs remainder r and quotients $q_1, \ldots, q_t$. If the remainder that is output is 0, then this implies $f \in \mathcal{I}(g_1, \ldots, g_t)$. We claim that this must be the case:

Lemma 13.4 *Let $\mathcal{J} \subset K[x_1, \ldots, x_n]$ be any polynomial ideal. Let $g_1, \ldots, g_t \in \mathcal{J}$ be any set of polynomials satisfying $\mathcal{I}(LT(g_1), \ldots, LT(g_t)) = \mathcal{I}(LT(\mathcal{J}))$. Let* `Divide` *be any division algorithm satisfying the Remainder Propery. Then for any $f \in \mathcal{J}$, the remainder output by* `Divide`$(f;\ g_1, \ldots, g_t)$ *is 0.*

Proof Suppose that the remainder $r \neq 0$. Since $r = f - q_1 g_1 - \cdots - q_t g_t$ for the quotients q_i output by the division algorithm, it must be that $r \in \mathcal{J}$. Hence $LT(r) \in \mathcal{I}(LT(\mathcal{J})) = \mathcal{I}(LT(g_1), \ldots, LT(g_n))$. Since $\mathcal{I}(LT(\mathcal{J}))$ is a monomial ideal, Claim 13.3 implies that $LT(r)$ must be divisible by $LT(g_i)$ for some i. But the Remainder Property of the Division Algorithm states that $LT(r)$ cannot be divisible by any of $LT(g_1), \ldots, LT(g_t)$, a contradiction. ■

Thus we have that all $f \in \mathcal{J}$ are also in $\mathcal{I}(g_1, \ldots, g_t)$. Hence $\mathcal{I}(g_1, \ldots, g_t) = \mathcal{J}$, and we have shown that every polynomial ideal over a field with finitely many variables has a finite Groebner basis. Note that in particular, this shows that every such ideal has a finite basis, which is the Hilbet Basis Theorem.

13.4 Division using Groebner bases

We first show that Groebner bases allow for a robust notion of division by polynomial ideals. Lemma 13.4 already shows that any polynomial in an ideal will yield a remainder of 0 when divided by any Groebner basis for the ideal. Using this basic fact, we can show that division by Groebner bases allows for a strong notion of uniqueness of remainder.

Lemma 13.5 *For any polynomial ideal $\mathcal{J} \subset K[x_1, \ldots, x_n]$ and any $f \in K[x_1, \ldots, x_n]$, the remainder r produced by* `Divide`$(f; g_1, \ldots, g_t)$ *will be unique for any Groebner basis $g_1, \ldots, g_t$ of $\mathcal{J}$ and any division algorithm* `Divide` *satisfying the Remainder Property.*

Proof Suppose that r_1 and r_2 were two remainders produced by possibly different division algorithms using Groebner bases $g_1, \ldots, g_s$ and $h_1, \ldots, h_t$, respectively. By the Remainder Property, we know that no term of r_1 is divisible by $LT(g_i)$ for any i, and no term of r_2 is divisible by $LT(h_j)$ for any j. Since the leading terms of any Groebner basis for $\mathcal{J}$ generate $\mathcal{I}(LT(\mathcal{J}))$, by Claim 13.3, we have that no terms of r_1 or r_2 are in $\mathcal{I}(LT(\mathcal{J}))$. Now, for suitable polynomials $p_1, \ldots, p_s$ and $q_1, \ldots, q_t$, $r_1 = f - p_1 g_1 - \cdots - p_s g_s$ and $r_2 = f - q_1 h_1 - \cdots - q_t h_t$. Hence, $r_1 - r_2 \in \mathcal{J}$, and so $LT(r_1 - r_2) \in \mathcal{I}(LT(\mathcal{J}))$. However, the leading monomial of $r_1 - r_2$ must be either 0 or be a monomial of either r_1 or r_2. But we know that no terms of r_1 or r_2 are in $\mathcal{I}(LT(\mathcal{J}))$, so $r_1 - r_2 = 0$. ■

13.5 More Groebner Basis Tricks

One consequence of the existence of finite Groebner Bases (actually just the existence of finite bases) for every ideal in $K[x_1, \ldots, x_n]$ is the Ascending Chain Condition:

Theorem 13.6 *For every infinite ascending chain of ideals $\mathcal{I}_1 \subset \mathcal{I}_2 \subset \cdots \subset K[x_1, \ldots, x_n]$, there exists an integer N_0 such that for all $n \geq N_0$, $\mathcal{I}_n = \mathcal{I}_{N_0}$.*

Rings satisfying this condition are called Noetherian rings. We show that $K[x_1, \ldots, x_n]$ is Noetherian:
Proof Let $\mathcal{J} = \bigcup_{i=1}^{\infty} \mathcal{I}_i$. Let $g_1, \ldots, g_t$ be a finite basis for $\mathcal{J}$. For each g_i, since $g_i \in \mathcal{J}$, there must exist some $\mu(i)$ such that $g_i \in \mathcal{I}_{\mu(i)}$. Let $N_0 = \max_i(\mu(i))$. We must have that $g_1, \ldots, g_t \in \mathcal{I}_{N_0}$. Hence $\mathcal{J} = \mathcal{I}_{N_0}$, and the theorem is proved. ■

Now we examine the question of uniqueness for Groebner Bases. Certainly Groebner bases need not be unique, since one can always add more elements of the ideal to a Groebner Basis. Suppose therefore we restrict ourselves to *minimal* Groebner bases, *i.e.* a Groebner basis $g_1, \ldots, g_t$ such that for all i, $LT(g_i) \notin \mathcal{I}(LT(g_1), \ldots, LT(g_{i-1}), LT(g_{i+1}), \ldots, LT(g_t))$. Note that if we start with a Groebner basis and eliminate all polynomials that do not satisfy the minimality constraint, by Lemma 13.4, what remains is still a Groebner basis. Since we are working in a field, we also adopt a convention requiring that the leading coefficients of polynomials in our bases be 1. This is still not enough to guarantee uniqueness, as $\{x + y, y\}$ and $\{x, y\}$ are both minimal Groebner bases for $\mathcal{I}(x, y)$. This leads us to the definition of a reduced Groebner basis:

Definition 13.7 *A* reduced Groebner basis *$g_1, \ldots, g_t$ is a minimal Groebner Basis that also satisfies the property that for all pairs i, j with $i \neq j$, we have that no term of g_i is divisible by $LT(g_j)$.*

Now we can prove uniqueness:

Theorem 13.8 *Every polynomial ideal $\mathcal{J} \subset K[x_1, \ldots, x_n]$ has a unique reduced Groebner basis.*

Proof First, we note that by definition, the leading terms of any minimal Groebner basis must be distinct. Moreover, we have the following:

Claim 13.9 *For any two minimal Groebner bases G and G', the set $LT(G)$ of leading terms of G must equal the set $LT(G')$ of leading terms of G'.*

Proof Suppose not. Let m be the minimal monomial such that m is in one set of leading terms but not the other. Without loss of generality, assume $m \in LT(G)$. Then $m \in \mathcal{I}(LT(G)) = \mathcal{I}(LT(G'))$, and hence by Claim 13.3, there must be some monomial a in $LT(G')$ such that a divides m. But since a divides m, by the minimality of G, it cannot be that $a \in LT(G)$. Hence a is in $LT(G')$ but not in $LT(G)$, contradicting the minimality of m. ■

In particular, this claim implies that all minimal Groebner bases of an ideal $\mathcal{J}$ have the same cardinality N, and in fact any Groebner basis of cardinality N must be a minimal Groebner basis. Now we can easily show existence of a reduced Groebner basis by the following algorithm:

Let $g_1, \ldots, g_t$ be any minimal Groebner basis. For $i = 1$ to t, let g_i' be the remainder when dividing g_i by $g_1', \ldots, g_{i-1}', g_{i+1}, \ldots, g_t$. Output $g_1', \ldots, g_t'$.

Since at each stage g_i' is set equal to $g_i - q_1 g_1' - \cdots - q_{i-1} g_{i-1}' - q_{i+1} g_{i+1} - \cdots - q_t g_t$ for some $q_1 \ldots q_t$, yet by the Remainder Property no leading term of $g_1', \ldots, g_{i-1}', g_{i+1}, \ldots, g_t$ can divide the leading term of g_i', it must be that $LT(g_i') = LT(g_i)$. Also, the span of $g_1', \ldots, g_i', g_{i+1}, \ldots, g_t$ is the same as the span of $g_1', \ldots, g_{i-1}', g_i, g_{i+1}, \ldots, g_t$. Hence $g_1', \ldots, g_t'$ remain a basis for $\mathcal{J}$, and since the leading terms do not change, it remains a minimal Groebner basis. However, now also, since the leading terms do not change, by the Remainder Property, no term of g_i' is divisible by the leading term of g_j for all $j \neq i$, so the basis $g_1', \ldots, g_t'$ is reduced.

We will now show that any two reduced bases must be the same. Suppose $g_1, \ldots, g_t$ and $g_1', \ldots, g_t'$ were two reduced bases. Since they are minimal, their set of leading terms must be the same, so we may assume that $LT(g_i) = LT(g_i')$ for all i. Suppose $g_i \neq g_i'$ for some i. Now, $g_i - g_i' \in \mathcal{J}$. But on the other hand, by the reduced property, $LT(g_i - g_i')$ cannot be divisible by any $LT(g_j)$, and hence by Claim 13.3 and the definition of a Groebner basis, $LT(g_i - g_i')$ cannot be in the ideal $\mathcal{I}(LT(\mathcal{J}))$, a contradiction. ■

13.6 An Algorithm for computing Groebner Bases

In this section we will give a simple condition which allows us to check if a given generating set is a Groebner basis. First we define the Syzygy polynomial $S(f, g)$:

Definition 13.10 *For two monomials $m_1 = x^\alpha$ and $m_2 = x^\beta$, we define $LCM(m_1, m_2) := x^\gamma$, where $\gamma_i = \max(\alpha_i, \beta_i)$.*

Definition 13.11 *For two polynomials f and g, we define the* Syzygy polynomial $S(f, g)$ of f and g *to be* $\frac{LC(g)\cdot LCM(LM(f),LM(g))}{LM(f)} \cdot f - \frac{LC(f)\cdot LCM(LM(f),LM(g))}{LM(g)} \cdot g$.

Note that the definition of $S(f, g)$ implies that $LM(S(f, g)) < LCM(LM(f), LM(g))$, since the highest order monomials will cancel out.

Given this definition, we can establish the following theorem:

Theorem 13.12 *$G = \{g_1, \ldots, g_t\}$ form a Groebner basis for $\mathcal{J} = \mathcal{I}(g_1, \ldots, g_t)$ iff for all pairs i, j, the remainder after executing* `Divide`$(S(g_i, g_j);\ g_1, \ldots, g_t)$ *is 0.*

Before we proceed with the proof of the theorem, we first examine our algorithm `Divide` more closely. We simply make the observation:

Claim 13.13 *If* `Divide`$(f;\ g_1, \ldots, g_s)$ *outputs a remainder r and quotients $q_1, \ldots, q_s$, then we always have that $LM(r) \leq LM(f)$ and $LM(q_i g_i) \leq LM(f)$ for each i.*

Proof Obvious from description of algorithm. ■

We also observe:

Claim 13.14 *Let $i > j$ and suppose the remainder after executing* `Divide`$(S(g_i, g_j);\ g_1, \ldots, g_t)$ *is 0 and leads to quotients $q_1, \ldots, q_t$.*

Then executing `Divide`$(\frac{LC(g_j)\cdot LCM(LM(g_i), LM(g_j))}{LM(g_i)} \cdot g_i;\ g_1, \ldots, g_t)$ *will lead to a remainder of 0 and quotients $q'_1, \ldots, q'_t$ where $q'_k = q_k$ for all k except $q'_j = q_j + \frac{LC(g_i)\cdot LCM(LM(g_i), LM(g_j))}{LM(g_j)}$.*

Proof This is immediate from the description of the divison algorithm and the fact that $LM(S(g_i, g_j)) < LM(\frac{LC(g_j)\cdot LCM(LM(g_i), LM(g_j))}{LM(g_i)} \cdot g_i)) = LCM(LM(g_i), LM(g_j))$. ■

Now we begin the proof of the theorem:

Proof (of Theorem 13.12) Clearly, $S(g_i, g_j) \in \mathcal{J}$, so by Lemma 13.4, if G is a Groebner basis, then all the remainders will be 0.

Suppose now that for each i, j the remainder after executing `Divide`$(S(g_i, g_j);\ g_1, \ldots, g_t)$ is 0. To show that G is a Groebner basis, we must show that for all $f \in \mathcal{J}$, we have that $LM(f)$ is divisible by $LM(g_i)$ for some i. Let us write $f = \sum m_{i,k} g_i$ for monomials $m_{i,k}$. We say that a term mg_i is *reducible* if we can write it as $mg_i = hg_j + \sum \mu_k g_k$ where h and μ_k are monomials, and $j < i$, and $LM(hg_j) = LM(mg_i)$ while $LM(\mu_k g_k) < LM(mg_i)$. Here we call $hg_j + \sum \mu_k g_k$ the *reduction* of mg_i. Now, we take our representation of f as $\sum m_{i,k} g_i$ and replace each reducible term $m_{i,k} g_i$ with its reduction, to obtain a new representation of f as $\sum h_{i,k} g_i$, where now no term is reducible. Now, we consider the terms $h_{i,k} g_i$ such that $LM(h_{i,k} g_i)$ has maximum degree. If there is only one such term, then it must be that $LM(f) = LM(h_{i,k} g_i)$, and hence $LM(g_i)$ divides $LM(f)$. Otherwise, suppose there are at least two terms hg_i and $h'g_j$ with $j < i$ such that their leading monomials are of the same degree. So we have that $LM(h)LM(g_i) = LM(h')LM(g_j)$. This implies that $LM(h)LM(g_i) = w \cdot LCM(LM(g_i), LM(g_j))$ for some monomial w. Hence, $LM(h) = w \cdot \frac{LCM(LM(g_i), LM(g_j))}{LM(g_i)}$. Let the leading coefficient of h be written as $c \cdot LC(g_j)$. Hence, finally we have that

$$hg_i = cw \cdot \frac{LC(g_j) \cdot LCM(LM(g_i), LM(g_j))}{LM(g_i)} \cdot g_i$$

By Claims 13.14 and 13.13 and our assumption, we have that hg_i is in fact reducible, a contradiction. Hence, after reducing all terms, we must have a single term with the highest degree monomial, implying that $LM(f) \in \mathcal{I}(LM(G))$, and completing the proof. ■

This characterization of Groebner basis immediately implies that the following algorithm will eventually terminate and produce a Groebner basis.

`Groebner − Basis`$(g_1, \ldots, g_t)$:

1. For each i, j execute `Divide`$(S(g_i, g_j);\ g_1, \ldots, g_t)$.
2. If all remainders are 0, return $g_1, \ldots, g_t$.
3. Otherwise, return `Groebner − Basis`$(g_1, \ldots, g_t, r)$, where r is one of the non-zero remainders found in Step 1.

A remainder is guaranteed to be such that $LM(r)$ is not in the ideal generated by $LM(g_1), \ldots, LM(g_t)$, so each recursion causes the ideal generated by the leading terms of the basis to become strictly larger. On the other hand, by Theorem 13.6, this ascending chain of ideals must eventually stop growing, and at that point, by Theorem 13.12 the algorithm must terminate with a Groebner basis.

Unfortunately, we have no bound, except for finiteness, on the running time of this algorithm or on the Groebner basis produced.

6.966 Algebra and Computation	October 19, 1998
Lecture 14	
Lecturer: Madhu Sudan	*Scribe: Salil Vadhan*

Explicit Bounds for Ideal Membership

In this lecture, we will prove a result due to Mayr and Meyer, stating that the ideal membership problem can be solved in exponential space. Recall that this is simply the following decision problem:

Ideal Membership (over any fixed field K)

Input Polynomials $g, f_1, \ldots, f_t \in K[x_1, \ldots, x_n]$.

Question Is $g \in (f_1, \ldots, f_t)$? That is, are there polynomials $q_1, \ldots, q_t \in K[x_1, \ldots, x_n]$ such that

$$g = \sum_{j=1}^{t} q_j f_j? \tag{14.1}$$

The main step in showing that this problem can be solved in exponential space will be to show that the q_i's can be taken to be of "reasonable" (doubly-exponential) degree; such degree bounds date back to the work of G. Hermann in 1926. We have already seen one algorithm for solving the ideal membership problem, namely the Groebner basis algorithm. The exponential-space algorithm we'll be seeing will not be based on Groebner bases, though the literature suggests that there are ways of using the same ideas to implement the Groebner basis algorithm in exponential space.

The algorithm we give will be based on viewing Equation (14.1) as a *linear* system over K. This can be done as follows. Say that d is an upper bound on the (total) degree of the f_j's and g and that $D = D(n, d, t)$ is an upper bound on the (total) degree of the q_j's. Then for every $j \in [t]$ and $\overline{\alpha} \in \{0, \ldots, D\}^n$, we let $q_{j,\overline{\alpha}}$ be an unknown (taking values in K) corresponding to the coefficient of $x^{\overline{\alpha}}$ in q_j and $f_{j,\overline{\alpha}}$ the coefficient of $x^{\overline{\alpha}}$ in f_j. Then Equation (14.1) can be rewritten as the following linear system:

$$\sum_{j=1}^{t} \sum_{\overline{\beta} \leq \overline{\alpha}} q_{j,\overline{\beta}} \cdot f_{j,\overline{\alpha}-\overline{\beta}} = g_{\overline{\alpha}} \qquad \forall\, \overline{\alpha} \in \{0, \ldots, D\}^n, \tag{14.2}$$

where $\overline{\beta} \leq \overline{\alpha}$ means that $\beta_k \leq \alpha_k$ for every $k \in [n]$.

Since linear systems can be solved in polylogarithmic space, the above system (with $t \cdot D^n$ unknowns and D^n equations) can be solved in space $\mathrm{poly}(n \log D, \log t)$. Thus, to show that the system can be solved in exponential (in n, t, and d) space, it suffices to prove a doubly exponential bound on D. We will obtain this bound by passing back and forth between the formulation given in Equation (14.1) (one linear equation over $K[x_1, \ldots, x_n]$) and the formulation given in System of Equations (14.2) (many linear equations over K). To facilitate this, we consider the following common generalization of both of these formulations:

Linear System of Polynomials (over any fixed field K)

Input Polynomials $\{f_{ij}\}$ and $\{g_i\}$ in $K[x_1, \ldots, x_n]$ for $i = 1, \ldots, s$ and $j = 1, \ldots, t$.

Question Do there exist $q_1, \ldots, q_t \in K[x_1, \ldots, x_n]$ such that

$$g_i = \sum_{j=1}^{t} q_j f_{ij} \qquad \forall\, i \in [s] \tag{14.3}$$

Actually, this problem is not much more general than the ideal membership problem if we allow ourselves to introduce new variables, as System of Equations (14.3)can be rewritten as the single equation

$$\sum_{i=1}^{s} g_i \cdot y_i = \sum_{i=1}^{s} \left(\sum_{j=1}^{t} q_j \cdot f_{ij} \right) \cdot y_i,$$

over $K[x_1, \ldots, x_n, y_1, \ldots, y_s]$.

However, it is convenient to state and prove our result for the many-equations version.

Theorem 14.1 (Mayr–Meyer, Hermann) *If* LINEAR SYSTEM OF POLYNOMIALS *has a solution, then it has a solution in which every q_i has total degree at most $D = (t \cdot d)^{2^n}$.*

Applying this to the special case of IDEAL MEMBERSHIP and reducing to solving a linear system as described earlier, we get

Corollary 14.2 IDEAL MEMBERSHIP *(for $K = \mathbb{Q}$ or $K =$ any fixed finite field) in* EXPSPACE.

Before proceeding to the proof, we observe that, in proving Theorem 14.1, we may assume that the field K is infinite without loss of generality. This is because the existence of a solution using polynomials of a given degree can be expressed as a linear system with coefficients in K (as in Equation (14.2)). We know that if such a linear system with coefficients in K has a solution in any field L containing K, then it also has a solution in K itself. So we can take L to be any infinite field containing L and prove the theorem for L. Thus, in the following two sections we will assume that K is infinite.

A special case: univariate polynomials

In this section, we treat the special case of Theorem 14.1 when the polynomials are all univariate, i.e. $n = 1$. We're given a $s \times t$ matrix $M = (f_{ij})$ with entries in $K[x]$ and a $g \in K[x]^s$ such that there exists a solution $q \in K[x]^t$ to $Mq = g$ and we want to prove that there exists a solution in which all entries of q have small degree.

First, let us see that, without loss of generality, we may assume that M is of full row rank (i.e., $rank(M) = s$), over $K(x)$. If this is not the case, there is some linear relationship among the rows $M_i = h_1 M_{i_1} + \cdots + h_k M_{i_k}$, where the h_ℓ's are in $K(x)$ and $i \notin \{i_1, \ldots, i_k\}$. We must also have $g_i = h_1 g_{i_1} + \cdots + h_k g_{i_k}$ or else the system $Mq = g$ would have no solution even over $K(x)$. From these two linear relationships it follows that removing row i from the system does not change the set of solutions, and hence we can keep removing rows until M has full row rank.

So, now let us assume that M has full row rank and hence $s \le t$. If $s = t$, then M is a nonsingular square matrix and the unique solution is $q = M^{-1} g$, where M^{-1} is the inverse of M (computed in $K(x)$). In general, however, s can be less than t and the system will have many solutions. To deal with this possibility, we will obtain a description of *all* solutions in $K(x)$. Since M has rank s, some subset of its columns of size s are linearly independent. By permuting the columns, we may assume that these are the first s columns, so we can describe M as a nonsingular $s \times s$ matrix A together with $r = t - s$ column vectors $v_1, \ldots, v_r$:

$$M = \left[\begin{array}{c|cccc} & | & | & & | \\ A & v_1 & v_2 & \cdots & v_r \\ & | & | & & | \end{array} \right] \tag{14.4}$$

Now, if we write $q = (z, a_1, \ldots, a_r)$, where $z \in K(x)^s$ and $a_1, \ldots, a_r \in K(x)$, then $Mq = Az + \sum_j a_j v_j$. So, q is a solution to $Mq = g$ iff $z = A^{-1}(g - \sum_j a_j v_j)$. We are interested in solutions where all the entries of q are polynomials (rather than rational functions); that is, where $a_1, \ldots, a_r$ and z are all in $K[x]$. However, understanding polynomial solutions are complicated by the fact that the entries of A^{-1} are rational functions. Here Cramer's Rule comes to the rescue — we can write $A^{-1} = [adj(A)]/\det(A)$, where $adj(A)$ is the $s \times s$ matrix whose ij'th entry is $(-1)^{i+j}$ times the determinant of A with row i and column j removed.

The key point is that the entries of $adj(A)$ are polynomials (of degree at most $(s-1)d$) and $\det(A)$ is a polynomial (of degree at most sd). Now, let $q = (z, a_1, \ldots, a_r)$ be any *polynomial* solution to $Mq = g$ (which exists by hypothesis). Then, for any $b_1, \ldots, b_r \in K[x]$, $q' = (z', a'_1, \ldots, a'_r)$ is also a polynomial solution for $a'_j \triangleq a_j - b_j \det(A)$ and

$$z' \triangleq A^{-1}\left(g - \sum_j a'_j v_j\right) = z + \sum_j b_j[adj(A)v_j].$$

By choosing the b_j's appropriately, we may assume that all the a'_j's are reduced modulo $\det(A)$ and hence have degree smaller than $\ell \triangleq \deg(\det(A)) \le sd$. Then, defining the degree of a matrix or vector to be the maximum degree of its entries, we have

$$\begin{aligned}
\deg(z') &\le \deg\left(A^{-1}\right) + \deg\left(g - \sum_j a'_j v_j\right) \\
&\le \deg(adj(A)) - \deg(\det(A)) + \max\left(\deg(g), \deg\left(\sum_j a'_j v_j\right)\right) \\
&\le (s-1)d - \ell + \max(d, \ell - 1 + d) \\
&\le sd
\end{aligned}$$

Thus, all the entries of q' have degree at most $sd \le td$. This completes the proof of the univariate case of Theorem 14.1.

Extending to multivariate polynomials

To prove Theorem 14.1 for multivariate polynomials (i.e. arbitrary n), we will work one variable at a time, viewing the multivariate polynomials in $x_1, \ldots, x_n$ as univariate polynomials in x_1 with coefficients in $K[x_2, \ldots, x_n]$. Then, almost everything we did in the last section will work, except the step when we reduced the a_i's modulo $\det(A)$. We can only reduce a polynomial modulo $\det(A)$ (with respect to variable x_1) only if $\det(A)$ is *monic* in x_1 (i.e., its maximum power of x_1 has a coefficient in K rather than $K[x_2, \ldots, x_n]$). Fortunately, it is easy to guarantee that this holds by using a random linear transformation to change variables.

Lemma 14.3 *Let $r_2, \ldots, r_n$ be chosen uniformly and independently from some subset $U \subset K$ of size u. For every polynomial $P \in K[x_1, \ldots, x_n]$, define $\tilde{P} \in K[y_1, \ldots, y_n]$ by $\tilde{P}(y_1, \ldots, y_n) = P(y_1, r_2y_1 + y_2, \ldots, r_ny_1 + y_n)$. Then,*

1. *For every choice of $r_2, \ldots, r_n$, $P \mapsto \tilde{P}$ is a ring isomorphism which preserves degree. That is,*
   (a) *The map $P \mapsto \tilde{P}$ is a bijection.*
   (b) *For any polynomials P and Q, $\widetilde{P+Q} = \tilde{P} + \tilde{Q}$ and $\widetilde{PQ} = \tilde{P}\tilde{Q}$.*
   (c) $\deg(\tilde{P}) = \deg(\tilde{Q})$.
2. *For any P of degree m, $\tilde{P}(\overline{y})$ is monic in y_1 with probability $\ge 1 - \frac{m}{u}$.*

Proof To see that $P \mapsto \tilde{P}$ is a bijection, note that it can be inverted by the rule $P(x_1, \ldots, x_n) = \tilde{P}(x_1, x_2 - r_2x_1, \ldots, x_n - r_nx_1)$. The remaining parts of Item 1 are straightforward. For Item 2, it suffices to show that the coefficient of y_1^m in $\tilde{P}(\overline{y})$ vanishes with probability at most m/u, as $\tilde{P}$ is of degree m. A straightforward calculation shows that the coefficient of y_1^m in $\tilde{P}(\overline{y})$ is exactly $P_m(1, r_2, \ldots, r_n)$, where P_m is the homogeneous degree m part of P. This is a nonzero polynomial of degree at most m in $r_2, \ldots, r_n$, so, by the Schwartz–Zippel Lemma, it vanishes with probability at most m/u. ■

Remark. In class, we did the change of variables using a random linear transformation chosen from the set of all $n \times n$ matrices over K. Above, we selected from a smaller set of linear transformations to avoid having to deal with the possibility that the transformation is not invertible.

Now we proceed with the proof of Theorem 14.1. We're given an $s \times t$ matrix $M = (f_{ij})$ with entries in $K[\overline{x}] \triangleq K[x_1, \ldots, x_n]$ and $g \in K[\overline{x}]^s$ such that there exists a polynomial solution $q \in K[\overline{x}]^s$ to $Mq = g$. As in the univariate case, we may assume that M is of full row rank over $K(\overline{x}) \triangleq K(x_1, \ldots, x_n)$ and can be written as a nonsingular $s \times s$ matrix A together with $r = t - s$ column vectors $v_1, \ldots, v_r$ (as in Equation (14.4)). Now suppose we choose $r_2, \ldots, r_n \in K$ randomly from subset $U \subset K$ of size $u = sd + 1$ and transform every polynomial as in Lemma 14.3 (and hence also transform every vector and matrix with polynomial entries). By Lemma 14.3, $\widetilde{\det(A)}$ is monic in y_1 with nonzero probability (as $\deg(\det(A)) \leq sd$). Hence, we may fix $r_2, \ldots, r_n$ such that $\widetilde{\det(A)}$ is monic in y_1.

Now we have the matrix $\tilde{M} = (\tilde{f}_{ij}) = (\tilde{A}|\tilde{v}_1 \cdots \tilde{v}_r)$ and vectors $\tilde{g}$ and $\tilde{q} \triangleq (\tilde{z}, \tilde{a}_1, \ldots, \tilde{a}_r)$. Since the transformation is a ring isomorphism, polynomial solutions q' to $Mq' = g$ of a given degree are in 1–1 correspondence with polynomial solutions $\tilde{q}'$ to $\tilde{M}\tilde{q}' = \tilde{g}$. In particular, we have $\tilde{M}\tilde{q} = \tilde{g}$ and our problem is reduced to finding another polynomial solution $\tilde{q}'$ to $\tilde{M}\tilde{q}' = \tilde{g}$ such that $\tilde{q}'$ is of low degree. As in the univariate case, for any $\tilde{b}_1, \ldots, \tilde{b}_r \in K[\overline{y}]$, $\tilde{q}' = (\tilde{z}', \tilde{a}'_1, \ldots, \tilde{a}'_r)$ is also a polynomial solution for $\tilde{a}'_j = \tilde{a}_j - \tilde{b}_j \det(\tilde{A})$ and $\tilde{z}' = \tilde{A}^{-1}(g - \sum_j \tilde{a}'_j \tilde{v}_j)$. Since the transformation is an isomorphism, $\det(\tilde{A}) = \widetilde{\det(A)}$, which is monic in y_1. Thus, by choosing $\tilde{b}_1, \ldots, \tilde{b}_r$ appropriately, we may assume that all the $\tilde{a}'_j$'s are of degree smaller than $\ell \triangleq \deg_{y_1}(\det(\tilde{A})) \leq sd$ in y_1. As in the univariate case, it then follows that $\tilde{z}'$ is of degree at most sd in y_1.

We have argued that the system $\tilde{M}\tilde{q}' = \tilde{g}$ has a solution $\tilde{q}'$ in which all the entries are of degree at most sd in y_1. Hence, we can restrict to such solutions and expand everything in terms of y_1 as follows: $\tilde{q}' = \tilde{q}'_0 + \tilde{q}'_1 y_1 + \tilde{q}'_2 y_1^2 + \cdots + \tilde{q}'_{sd} y_1^{sd}$, $\tilde{M} = \tilde{M}_0 + \tilde{M}_1 y_1 + \cdots + \tilde{M}_d y_1^d$, and $\tilde{g} = \tilde{g}_0 + \tilde{g}_1 y_1 + \cdots + \tilde{g}_d y_1^d$. Equating terms of the same degree in y_1 in $\tilde{M}\tilde{q}' = \tilde{g}$, we obtain a system of $(sd + d)s$ equations over $K[y_2, \ldots, y_n]$ with $(sd)t \leq t^2 d$ unknowns in which all coefficients still have degree at most d. Moreover, any solution to the new system of total degree k yields a solution to the original system of total degree at most $k + sd \leq k + td$.

Now we proceed inductively. A system with $t_0 = t$ unknowns in n variables yields a system $t_1 = t_0^2 d$ unknowns in $n - 1$ variables, which in turn yields $t_2 = t_1^2 d = t_0^4 d^3$ unknowns in $n - 2$ variables, and so on with $t_i = t_0^{2^i} d^{2^i - 1}$ unknowns in $n - i$ variables. Since passing from $n - i$ variables to $n - i - 1$ variables contributes degree at most $t_i d$ to the total degree of the solution, we end up with a solution of total degree $\sum_{i=0}^{n-1} t_i d < (td)^{2^n}$.

EXPSPACE-completeness

Because an exponential space is far from "efficient" by our usual standards, one might wonder whether there's a better algorithm for Ideal Membership. Unfortunately, we cannot hope to do better.

Theorem 14.4 (Mayr–Meyer) *Ideal Membership is EXPSPACE-complete.*

To show that Ideal Membership is EXPSPACE-hard, Mayr and Meyer reduce from the following EXPSPACE-hard problem.

Input $(\Sigma, R, \alpha, \beta)$, where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is a finite alphabet, $\alpha, \beta \in \Sigma^*$, and S is a set of rewrite rules of the form "$\gamma = \delta$" for $\gamma, \delta \in \Sigma^*$ such that S also contains all rules of the form $\sigma_i = \sigma_j$.

Output Can $\alpha = \beta$ be derived from the rewrite rules in S?

To reduce this problem to Ideal Membership, Mayr and Meyer define polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$ where $n = |\Sigma|$. To every string $\gamma \in \Sigma^*$, they associate the monomial $x^\gamma \triangleq x_1^{d_1} \cdots x_n^{d_n}$, where d_i is the number of times the symbol σ_i occurs in γ. To a rewrite rule $\gamma = \delta$, they associate the polynomial $f_{\gamma,\delta} = x^\gamma - x^\delta$. Then, one can show that $f_{\alpha,\beta}$ is in the ideal generated by $\{f_{\gamma,\delta} \colon \text{"}\gamma = \delta\text{"} \in S\}$ iff $\alpha = \beta$ can be derived from S.

What about the Nullstellensatz?

Recall that our initial motivation for looking at the ideal membership problem was the problem of deciding whether a system of polynomial equations $\{f_i = 0\}$ has a solution (in, say, some algebraically closed field). By Hilbert's (Weak) Nullstellensatz (which we will prove in upcoming lectures), the latter problem is equivalent to deciding whether 1 is the ideal $(f_1, \ldots, f_i)$. Today we studied a more general problem, testing whether an arbitrary polynomial is in an ideal. Although one might naively guess that the two problems are equally hard, testing whether 1 is in an ideal is in fact provably easier. Specifically, this problem is in PSPACE ("polynomial space"), which is strictly contained in EXPSPACE by the space hierarchy theorem. In fact, Koiran has shown that it is even in the Polynomial-Time Hierarchy if the Extended Riemann Hypothesis is true, as we'll see several lectures from now.

The PSPACE result is proven by showing *singly* exponential degree bounds on the solutions $\{q_i\}$ to $1 = \sum_i q_i f_i$ (when they exist). Such bounds were exhibited by Brownawell in 1986, and decidability in polynomial space follows immediately by viewing it as a linear system, as described at the beginning of this lecture. A generalization of this was given by Kollar in 1988, who showed a singly exponential degree bounds on solutions $\{q_i\}$ to the question "$\exists k\ f^k = \sum_j q_j f_j$" (also known as testing membership in the "radical" of an ideal).

Despite the impractical size of the degree bounds we proved today, what we did to obtain these bounds is in fact useful. It allows us to eliminate a constant number of variables (in a linear system of polynomials) with only a polynomial blowup in degree. Eliminating variables (and quantifiers) "efficiently" in a more general setting will be the focus of the next couple of lectures.

6.966 Algebra and Computation	October 28, 1998
Lecture 15	
Lecturer: Madhu Sudan	*Scribe: Ted Allison*

15.1 Variable Elimination

In this section we will examine what happens when variables are eliminated from a set of polynomials and how this affects the common zeros of the polynomials. To do this we will look at a new way of creating an ideal.

Definition 15.1 *Let $I = \langle f_1, \ldots, f_s \rangle$ be an ideal in $k[x_1, \ldots, x_n]$. Then define $I_T = I \cap k[x_1, \ldots, x_n - T]$, where $T \subset \{x_1, \ldots, x_n\}$. If $T = \{x_1, \ldots, x_l\}$, then I_T is called the l^{th}* **elimination ideal** *of I and is denoted I_l.*

This naturally raises the usual questions for an ideal: what is a basis for I_l? how do we determine membership in I_l? These questions are addressed by the elimination theorem. A related question is answered by the extension theorem: given a point $(a_2, \ldots, a_n) \in \mathbf{V}(I_{\{x_1\}})$, when can we find a value a_1 such that $(a_1, \ldots, a_n) \in \mathbf{V}(I)$? In other words, given a partial solution to a system, when does a complete solution exist? The concept of elimination ideals will be useful in solving this problem.

Theorem 15.2 (Elimination Theorem) *Let G be a Groebner basis for an ideal I under lexicographic order, $x_1 > x_2 > \cdots > x_n$. Then $G_l = G \cap k[x_{l+1}, \ldots, x_n]$ is a Groebner basis for $I_l = I \cap k[x_{l+1}, \ldots, x_n]$, the l^{th} elimination ideal of I.*

Proof: Since $G_l \subset I_l$, it suffices to show that $\langle \mathtt{LT}(G_l) \rangle = \langle \mathtt{LT}(I_l) \rangle$, and one direction is trivial.

To show $\langle \mathtt{LT}(I_l) \rangle \subset \langle \mathtt{LT}(G_l) \rangle$, choose a polynomial $f \in I_l$. Because G is a Groebner basis of I, $\mathtt{LT}(f)$ is divisible by $\mathtt{LT}(g)$ for some $g \in G$. But $\mathtt{LT}(f)$ contains none of the variables $x_1, \ldots, x_l$, so neither does $\mathtt{LT}(g)$. Then by the monomial ordering, neither does any other term of g. Thus $g \in G_l$, which shows that $\mathtt{LT}(f) \in \langle \mathtt{LT} G_l) \rangle$, and the desired inclusion is proved. ■

Theorem 15.3 (Extension Theorem) *Given a set of polynomials $f_i(x, y_1, y_2, \ldots, y_n), 1 \leq i \leq s$ over an algebraically closed field k, write each f_i in the form $f_i(x, y_1, \ldots, y_n) = g_i(y_1, \ldots, y_n)x^{d_i} +$ [terms of lower degree in x], and let $I = \langle f_1, \ldots, f_s \rangle$. If $(a_1, \ldots, a_n) \in \mathbf{V}(I_{\{x\}})$ and $(a_1, \ldots, a_n) \notin \mathbf{V}(g_1, \ldots, g_s)$, then there is some $a \in k$ such that $(a, a_1, \ldots, a_n) \in \mathbf{V}(I)$.*

Before proving the theorem, let us look at some examples of how extensions of partial might not exist if one of the hypotheses fails to hold.

First consider $\langle x - y^2 \rangle$ with the monomial ordering $y > x$. It is easy to see that I_y is the trivial ideal $\{0\}$, which would seem to imply that for any value of y, there is a value for x which yields a complete solution. But this is obviously not true over a field such as the real numbers which is not algebraically closed.

Next consider $I = \langle xy - 1, xz - 1 \rangle$ using the ordering $x > y > z$ over an algebraically closed field. Then $z - y = y(xz - 1) - z(xy - 1) \in I_x$, and in fact $I_x = \langle z - y \rangle$. Clearly $(0, 0) \in \mathbf{V}(I_x)$, but there is no value for x which extends to a complete solution $(x, 0, 0)$ for the polynomials in I. On closer inspection we see that the polynomials g_i defined in the theorem are

$$g_1 = y$$

and

$$g_2 = z.$$

Then we have $\mathbf{V}(g_1, g_2) = \mathbf{V}(y, z) = \{(0, 0)\}$, and so no extension was expected for this point.

Recall also remember the resultant function. Given two polynomials $h_1(x) = c_0 + c_1x + \cdots + c_nx^n$ and $h_2(x) = d_0 + d_1x + \cdots + d_mx^m$, with $c_n \neq 0$ and $d_m \neq 0$, the question arises, can we find two lower degree polynomials $u(x) = u_0 + u_1x + \cdots + u_{m-1}x^{m-1}$ and $v(x) = v_0 + v_1x + \cdots + v_{n-1}x^{n-1}$ such that $u(x)h_1(x) + v(x)h_2(x) = 1$? This reduces to a system of linear equations in the coefficients of x represented by the matrix:

$$\begin{bmatrix} c_0 & & & & d_0 & & & \\ c_1 & c_0 & & & d_1 & d_0 & & \\ & c_1 & \ddots & & & d_1 & \ddots & \\ \vdots & & \ddots & c_0 & \vdots & & \ddots & d_0 \\ c_n & \vdots & & c_1 & d_m & \vdots & & d_1 \\ & c_n & & & & d_m & & \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & c_n & & & & d_m \end{bmatrix}$$

The determinant of this matrix is the resultant of h_1 and h_2 with respect to x, $Res(h_1, h_2, x)$. Of course, if the coefficients c_i and d_j are polynomials in variables y_k, then the resultant will also be a polynomial in these variables.

Now we are ready to prove the extension theorem.

Proof: We will begin with the special case $n = 0, s = 2$. This is equivalent to asking, "Do the polynomials $f_1(x)$ and $f_2(x)$ have a common root?" Symbolically, this becomes

$$\begin{aligned} \exists a : f_1(a) = f_2(a) = 0 \quad & \Leftrightarrow \quad gcd(f_1, f_2) \neq 1 \\ & \Leftrightarrow \quad \forall u(x)\forall v(x), uf_1 + vf_2 \neq 1 \\ & \Leftrightarrow \quad 1 \notin \langle f_1, f_2 \rangle \\ & \Leftrightarrow \quad \langle 1 \rangle \neq I_x \end{aligned}$$

Note that I_x is either $\{0\}$ or $\langle 1 \rangle$, and these two cases are equivalent to the existence and nonexistence of a solution, respectively.

The next case to consider is again $s = 2$, but with $n > 0$. Let the two polynomials be $f_1(x, y_1, \ldots, y_n)$ and $f_2(x, y_1, \ldots, y_n)$ and let $I = \langle f_1, f_2 \rangle$. Let $p(y_1, \ldots, y_n) = Res(f_1, f_2, x)$, and note that $p \in I_x$. Let $(a_1, \ldots, a_n) \in \mathbf{V}(I_x)$. Then $p(a_1, \ldots, a_n) = 0$. It must now be shown that there is some a such that $f_1(a, a_1, \ldots, a_n) = f_2(a, a_1, \ldots, a_n) = 0$ if $(a_1, \ldots, a_n) \notin \mathbf{V}(g_1, g_2)$.

Define $h_i(x) = f_i(x, a_1, \ldots, a_n)$. Assuming $(a_1, \ldots, a_n) \notin \mathbf{V}(g_1, g_2)$, it cannot be that both $g_1(a_1, \ldots, a_n)$ and $g_2(a_1, \ldots, a_n)$ are zero. If neither are zero, then

$$\begin{aligned} Res(h_1, h_2, x) &= Res(f_1, f_2, x)(a_1, \ldots, a_n) \\ &= p(a_1, \ldots, a_n) \\ &= 0. \end{aligned}$$

But this means that there are no polynomials $u(x), v(x)$ such that $u(x)h_1(x) + v(x)h_2(x) = 1$, which from above implies that h_1 and h_2 have a common root a. This root provides an extension to the partial solution such that $f_1(a, a_1, \ldots, a_n) = f_2(a, a_1, \ldots, a_n) = 0$.

If one is zero, say $g_2(a_1, \ldots, a_n)$, then it is not true that $Res(h_1, h_2, x) = Res(f_1, f_2, x)(a_1, \ldots, a_n)$, since the definition of the resultant requires nonzero leading coefficients. But consider the ideal $\langle f_1, f_2 + x^r f_1 \rangle$. It is easy to see that this ideal is identical to $\langle f_1, f_2 \rangle$, so a solution to one pair of functions is a solution to the other. Furthermore, it is always possible to choose r high enough that, viewing f_1 and f_2 as polynomials in x with coefficients which are polynomials in y, the leading coefficient of $f_2 + x^r f_1$ is the same as the leading coefficient of f_1. Then the leading coefficients of both functions are nonzero at the partial solution $(a_1, \ldots, a_n)$, and the existence of an extension to a complete solution is guaranteed by the previous argument.

Now the proof is complete in the case where $s = 2$. In the case where $s > 2$, we will reduce to the proven case by trading off excess polynomials for additional variables.

Let $f_1, \ldots, f_s$ be given, and let $(a_1, \ldots, a_n) \in \mathbf{V}(I_x(f_1, \ldots, f_s))$ be a partial solution. Order the f_i so that $g_1(a_1, \ldots, a_n) \neq 0$.

Define $U(x, y_1, \ldots, y_n, u_2, \ldots, u_n) = \sum_{i=2}^{s} u_i f_i(x, y_1, \ldots, y_n)$. Consider the resultant of U and f_1 with respect to x written as a sum of monomials in the new variables u_i with coefficients which are polynomials in y_i. Let the coefficients be h_α; then $Res(f_1, U, x) = \sum_\alpha h_\alpha(y_1, \ldots, y_n) u^\alpha$. It can be shown that $h_\alpha \in \langle f_1, \ldots, f_s \rangle_{\{x\}}$. This means that $h_\alpha(a_1, \ldots, a_n) = 0$, so $Res(f_1, U, x)(a_1, \ldots, a_n, u_2, \ldots u_s) = 0$.

Now let $\tilde{f}_1(x) = f_1(x, a_1, \ldots, a_n)$ and $\tilde{U}(x, u_2, \ldots, u_s) = U(x, a_1, \ldots, a_n, u_2, \ldots, u_s)$. As in the previous case, we have, possibly after replacing $\tilde{U}$ with $\tilde{U} + x^r f_1$, $Res(\tilde{f}_1, \tilde{U}, x) = 0$. This implies that $\tilde{f}_1(x)$ and $\tilde{U}(x, u_2, \ldots, u_s)$ have a common factor, which must be of the form $(x - a)$ for some a since $\tilde{f}_1$ does not contain any of the u_i. But then $\tilde{U}(a, u_2, \ldots, u_s) = 0$, from which it is easy to see that $f_i(a, a_1, \ldots, a_n) = 0$ for all i. Thus an extension exists in this final case, and the theorem is proved. ■

15.2 Hilbert's Nullstellensatz

Now that we have answered the question of when we can find a point in a variety of a set of polynomials, it is logical to ask when we cannot. In other words, given a set of polynomials $f_1, \ldots, f_s$, how can we claim that $\forall a_1, \ldots, a_n, \exists i : f_i(a_1, \ldots, a_n) \neq 0$, or, equivalently, $\mathbf{V}(f_1, \ldots, f_s) = \emptyset$?

Consider a random invertible linear transform R and define $(y_1, \ldots, y_n)^T = R^{-1}(x_1, \ldots, x_n)^T$. Set $\tilde{f}_i(y_1, \ldots, y_n) = f_i(R(y_1, \ldots, y_n))$. The new functions $\tilde{f}_i$ will have a common root if and only if the original f_i do. Note that because R has no effect on constant polynomials, $1 \in \tilde{I} = \langle \tilde{f}_1, \ldots, \tilde{f}_s \rangle \Rightarrow \langle f_1, \ldots, f_s \rangle$.

There is some transform R such that the new polynomials $\tilde{f}_i$ are monic in y_1. By the extension theorem, if $\tilde{I}_1$ has a common root, then it extends to a common root of $\tilde{I}$. Assuming then that there is no root, it can be shown by induction that $1 \in \tilde{I}_1$, which implies that $1 \in \tilde{I}$, which agin implies that $1 \in \langle f_1, \ldots, f_s \rangle$.

This argument shows that the Extension Theorem leads to the following result, due to Hilbert.

Theorem 15.4 (Weak Nullstellensatz) *Given a set of polynomials $f_1, \ldots, f_s$ over an algebraically closed field, $1 \in \langle f_1, \ldots, f_s \rangle$ if and only if $\mathbf{V}(f_1, \ldots, f_s) = \emptyset$.*

6.966 Algebra and Computation November 2, 1998

Lecture 16

Lecturer: Madhu Sudan *Scribe: Leonid Reyzin*

Nullstellensatz

To review, we've been considering the following two questions, given s polynomials $f_1, \ldots, f_s$ in n variables over a field K of maximum total degree d.

1. Do they have a common zero? That is, is their variety non-empty?
2. Is a given polynomial f in the ideal $I(f_1, \ldots, f_s)$? That is, do there exist polynomials $q_1, \ldots, q_s \in K[x_1, \ldots, x_n]$ such that $f = \sum q_i f_i$.

Two lectures ago we proved a doubly exponential upper bound on the q_i's for question 2. One lecture ago we proved a relationship between the two questions (for $f = 1$) with the theorem known as "Hilbert's Nullstellensatz." We review the main idea of the proof below.

In order to find the answer to the first question, we can try to find a point $(a_1, \ldots, a_n) \in V(f_1, \ldots, f_s)$. We can reduce the number of dimensions by considering

$$\Pi_1 = \{(a_2, \ldots, a_n) | \exists a_1 : (a_1, a_2, \ldots, a_n) \in V(f_1, \ldots, f_s)\}.$$

Then we can first look for a point in Π_1 and then extend it to a point in $V(f_1, \ldots, f_s)$ by looking for a common zero of the s univariate polynomials $f_i(x_1, a_2, \ldots, a_n)$ for $1 \le i \le s$.

It seems that, at least in principle, finding a common zero should not be hard. The hard part is looking for the point in Π_1. The hope is that Π_1 is itself a variety : $\exists g_1, \ldots, g_t \in K[x_2, \ldots, x_n]$ such that $\Pi_1 = V(g_1, \ldots, g_t)$.

We weren't quite able to prove that. We were able to prove, however, that Π_1 contains a variety if K is algebraically closed and $f_1, \ldots, f_s$ are monic in x_1 (this is the Extension Theorem). More specifically, let

$$f_i = x_1^{d_i} g_i(x_2, \ldots, x_n) + x_1^{d_i - 1} \ldots$$

and let

$$\begin{aligned} I_1 &= I(f_1, \ldots, f_s) \cap K[x_2, \ldots, x_n] \\ J &= I(g_1, \ldots, g_s). \end{aligned}$$

For a point $(a_2, \ldots, a_n) \in V(I_1) - V(J)$, there exists an extension a_1 such that $(a_1, a_2, \ldots, a_n) \in V(I)$. So, we get that $V(I_1) - V(J) \subseteq \Pi_1$. This was enough to prove the following theorem.

Theorem 16.1 *(Hilbert's Weak Nullstellensatz.) If K is algebraically closed, then $(f_1, \ldots, f_s)$ have a common zero if and only if $1 \notin I(f_1, \ldots, f_s)$.*

We will now work towards a strong version of Nullstellensatz.

So far, we have been considering the case of $V = \emptyset$ or $V \neq \emptyset$. But varieties have more geometric meaning, and we'd like to look more closely at the ideals when $V \neq \emptyset$.

Let V be a set of points in K^n. Define the ideal defining V as $I(V) = \{f \in K[x_1, \ldots, x_n] | f(x) = 0 \ \forall x \in V\}$.

To better understand the definition, consider the following examples. Let $f_1 = x^2 - y^2$ and $f_2 = x + y$. Then $V(f_1, f_2) = \{(x, y) | x + y = 0\}$. Therefore, $I(V(f_1, f_2)) = \{(x + y)p(x, y) | \forall p(x, y) \in K[x, y]\} = I(x + y) = I(f_1, f_2)$. Now let $f_1 = (x + y)^2(x - y)^2$ and $f_2 = (x + y)^2$. Note that $V(f_1, f_2)$ is still the same, and therefore $I(V(f_1, f_2)) = I(x + y)$, even though $I(f_1, f_2) = I((x + y)^2)$.

Definition 16.2 *A radical for an ideal I is* $\mathrm{Rad}(I) = \{f \mid \exists m > 0 : f^m \in I\}$.

$\mathrm{Rad}(I)$ is an ideal. The proof is left as an exercise to the reader.

This definition together with the following theorem explain what happened in the example above.

Theorem 16.3 *(Strong Nullstellensatz.) If K is algebraically closed, and $f_1, \dots, f_s \in K[x_1, \dots, x_n]$, then* $\mathrm{Rad}(I(f_1, \dots, f_s)) = I(V(I(f_1, \dots, f_s)))$.

First, we'd like to note that Strong Nullstellensatz easily implies Weak Nullstellensatz, as expected. Indeed, if $V(f_1, \dots, f_s) = \emptyset$, then $1 \in I(V(f_1, \dots, f_s))$ (by definition of the ideal defining a variety), so $1 \in \mathrm{Rad}(I(f_1, \dots, f_s))$ (by Strong Nullstellensatz), so $1 \in I(f_1, \dots, f_s)$. The converse doesn't even need Strong Nullstellensatz: if $1 \in I(f_1, \dots, f_s)$, then $1 = \sum q_i f_i$; if all the f_i's have a common zero, evaluate at it to get $1 = 0$.

We will now use the Weak Nullstellensatz to prove the Strong Nullstellensatz. Note that we are not involved in circular reasoning here, because we originally proved the Weak Nullstellensatz without the Strong Nullstellensatz.

Proof Let $I = I(f_1, \dots, f_s)$ and $V = V(I)$. Suppose $f \in \mathrm{Rad}(I)$. We need to show that $f \in I(V)$, i.e., that f vanishes on every point in V. Since $f \in \mathrm{Rad}(I)$, $f^m \in I$, so f^m vanishes on every point in V. Therefore, so does f.

Now, suppose $f \in I(V)$. We need to show that $f \in \mathrm{Rad}(I)$. If $f = 0$, we are done. So assume $f \neq 0$. We will use Rabinowitsch's trick. Consider the polynomials $f_1, \dots, f_s, 1 - yf$, for a new variable y. Note that if for some point $a = (a_1, \dots, a_n, b)$ we have that $f_1(a) = \dots = f_s(a) = 0$, then $f(a) = 0$ (because $(a_1, \dots, a_n) \in V$ and f vanishes on V), so $1 - yf \neq 0$. Therefore, $V(f_1, \dots, f_s, 1 - yf) = \emptyset$. Therefore, by the Weak Nullstellensatz, $1 \in I(f_1, \dots, f_s, 1 - yf)$. Therefore, for some $q_1, \dots, q_s$ and $p \in K[x_1, \dots, x_n, y]$ we have that

$$1 = \left(\sum_{i=1}^{s} q_i(x_1, \dots, x_n, y) f_i(x_1, \dots, x_n) \right) + (1 - yf(x_1, \dots, x_n))\, p(x_1, \dots, x_n).$$

Note that this identity holds over $K[x_1, \dots, x_n, y]$ and therefore also over $K(x_1, \dots, x_n)[y]$. Now substitute $y = \frac{1}{f(x_1, \dots, x_n)}$ ($f \neq 0$, so this is allowed). Then

$$1 = \sum_{i=1}^{s} q_i \left(x_1, \dots, x_n, \frac{1}{f} \right) f_i(x_1, \dots, x_n).$$

If we let m be the maximum degree of q_i in y, then multiplying through by f^m we get rid of the denominators to get

$$f^m = \sum_{i=1}^{s} \left(f^m q_i \left(x_1, \dots, x_n, \frac{1}{f} \right) \right) f_i(x_1, \dots, x_n).$$

Since f is a polynomial only in $x_1, \dots, x_n$, this identity holds in $K[x_1, \dots, x_n]$. Therefore, we expressed f^m as a linear combination of $f_1, \dots, f_s$, so $f^m \in I$, so $f \in \mathrm{Rad}(I)$. ■

The above proof shows that $f \in \mathrm{Rad}(I(f_1, \dots, f_s))$ if $1 \in I(f_1, \dots, f_s, 1 - yf)$. The converse also holds: if $1 \notin I(f_1, \dots, f_s, 1 - yf)$, then by Nullstellensatz some point $(a_1, \dots, a_n, b)$ is a common zero for $(f_1, \dots, f_s, 1 - yf)$. Note that $f(a_1, \dots, a_n) \neq 0$, because otherwise the last polynomial would 1. Therefore, f is not zero at some point where $f_1, \dots, f_s$ are zero; therefore, $f^m \notin I(f_1, \dots, f_s)$ for any m, and hence $f \notin \mathrm{Rad}(I(f_1, \dots, f_s))$.

Thus, to check if $f \in \mathrm{Rad}(I(f_1, \dots, f_s))$ one can just check if $1 \in I(f_1, \dots, f_s, 1 - yf)$. This is a PSPACE question (because, by some recent results, checking if 1 is a member of an ideal is in PSPACE). On the other hand, to check if $f \in I(f_1, \dots, f_s)$ is EXPSPACE-complete.

Quantifier Elimination

Suppose we are given a boolean formula φ in s variables. Let $f_1, \ldots, f_s$ be n-variate polynomials of maximum total degree d over an algebraically closed field K. Let $Q_1, \ldots, Q_w$ be quantifiers ($Q_i \in \{\forall, \exists\}$). And finally, let $x_1, \ldots x_n$ be n variables divided into $w+1$ blocks $x^{[0]}, x^{[1]}, \ldots, x^{[w]}$. Note that whether $f_i(x_1, \ldots, x_n) = 0$ is a boolean variable whose value depends on $x_1, \ldots, x_n$. This boolean variable can be used as input to φ. Then we can consider the following quantified formula:

$$F(x^{[0]}) = Q_1 x^{[1]} \ldots Q_w x^{[w]} \varphi(f_1(x_1, \ldots, x_n) = 0, \ldots, f_s(x_1, \ldots, x_n) = 0),$$

whose boolean value depends on the variables in the block $x^{[0]}$. Let m be the number of variables in $x^{[0]}$.

(Note that if $f_i = 1 - x_i$, then deciding the value of this formula is a complete problem for a level of the polynomial hierarchy (if w is bounded) or for PSPACE (if w is unbounded).)

We are required to output a boolean formula ψ on t variables and polynomials $g_1, \ldots, g_t$ in m variables of degree at most D such that

$$\psi(g_1(x^{[0]}) = 0, \ldots, g_t(x^{[0]}) = 0) = F(x^{[0]}).$$

Note that *a priori* it's not even clear that it is possible to eliminate quantifiers, i.e., that such ψ exists. Here and in the next lecture we will prove that they do and obtain reasonable bounds on t, D and the size of ψ.

A simple special case

Consider the case when $w = 1$, $Q_1 = \exists$, and $\varphi = \wedge$. Then

$$F(x^{[0]}) = \exists x^{[1]} \bigwedge_{i=1}^{s} (f_i(x^{[0]}, x^{[1]}) = 0).$$

By the Nullstellensatz, we know that for any particular $x^{[0]}$, $\neg F(x^{[0]}) \Leftrightarrow \exists q_1, \ldots, q_s \in K[x^{[1]}]$ s.t. $1 = \sum_{i=1}^{s} q_i f_i$.

Thus, all we need to do is to attempt to find such $q_1, \ldots, q_s$. We can do so by setting up a system of linear equations $Ay = b$, where A is a matrix and b is a vector with coefficients from $K[x^{[0]}]$. This system has a solution if and only if $\mathrm{rank}(A) \geq \mathrm{rank}(A|b)$. Thus, we can construct ψ as follows:

$$\psi(x^{[0]} = \bigwedge_k (\mathrm{rank}(A|b) < k \vee \mathrm{rank}(A) \geq k)$$

The only remaining question now is how to express for some matrix X that $\mathrm{rank}(X) < k$?

For a set S of row-numbers and T of column-numbers such that $|S| = |T| = i$, let $A_{S,T}$ denote the $i \times i$ minor of the matrix A consisting of the entries at the intersections of the rows whose numbers are in S and the columns whose numbers are in T. Consider $\det(A_{S,T})$. It is a polynomial in $x^{[0]}$ of degree about d^n. Denote it by $p_{S,T}(x^{[0]})$.

We can now express $\mathrm{rank}(A) < k$ by a statement that the determinants of all the minors of size k are 0:

$$\bigwedge_{S,T,|S|=|T|=k} (p_{S,T}(x^{[0]}) = 0).$$

This looks promising, except for one problem: there are too many elements under the $\bigwedge$, because there are too many size k minors possible. However, all we need is a small subset of all possible $p_{S,T}$: namely, those that generate an ideal that contains the rest. Since there are about d^{n^2} coefficients in each $p_{S,T}$, there can't be too many linearly independent polynomials. So, all we need to do is to find a linearly independent subset of all $p_{S,T}$ that generates the rest. Such a set can be found using Gaussian elimination. The details are left as an exercise to the reader.

In the next lecture, we will perform a series of generalizations to show how to do quantifier elimination for any formula.

6.966 Algebra and Computation November 4, 1998

Lecture 17

Lecturer: Madhu Sudan *Scribe: Anna Lysyanskaya*

Quantifier Elimination (continued)

Recall that in the last lecture we stated the problem of quantifier elimination:

We are given a boolean formula φ in s variables. Let $f_1, \ldots, f_s$ be n-variate polynomials of maximum total degree d over an algebraically closed field K. Let $Q_1, \ldots, Q_w$ be quantifiers ($Q_i \in \{\forall, \exists\}$). And finally, let $x_1, \ldots x_n$ be n variables divided into $w+1$ blocks $x^{[0]}, x^{[1]}, \ldots, x^{[w]}$. Note that $f_i(x_1, \ldots, x_n) = 0$ is a boolean variable whose value depends on $x_1, \ldots, x_n$. This boolean variable can be used as input to φ. Consider the following quantified formula:

$$F(x^{[0]}) = Q_1 x^{[1]} \ldots Q_w x^{[w]} \varphi(f_1(x_1, \ldots, x_n) = 0, \ldots, f_s(x_1, \ldots, x_n) = 0),$$

whose boolean value depends on the variables in the block $x^{[0]}$. Let m be the number of variables in $x^{[0]}$. Output a boolean formula ψ on t variables and polynomials $g_1, \ldots, g_t$ in m variables of degree at most D such that

$$\psi(g_1(x^{[0]}) = 0, \ldots, g_t(x^{[0]}) = 0) = F(x^{[0]}).$$

In the last lecture, we dealt with a special case of this problem, when $w = 1$, $Q_1 = \exists$, and $\varphi = \wedge$. Then

$$F(x^{[0]}) = \exists x^{[1]} \bigwedge_{i=1}^{s} (f_i(x^{[0]}, x^{[1]}) = 0).$$

We saw that, using the Hilbert Nullstellensatz, we can transform F into a quantifier-free formula of involving at most $(sd)^{n^2}$ polynomials in n variables and of degree at most $(sd)^n$. Our goal is to apply that result to the general case, and obtain good bounds on D, the size of the output, and the running time of the computation.

Handling Inequalities

The first generalization we are interested in is the case when we have both equalities and inequalities inside the AND. That is, $w = 1$, $Q_1 = \exists$, and $\varphi = \wedge$. Then

$$F(x^{[0]}) = \exists x^{[1]} (\bigwedge_{i=1}^{s_1} (f_i(x^{[0]}, x^{[1]}) = 0)) \wedge (\bigwedge_{i=1}^{s_2} (f'_i(x^{[0]}, x^{[1]}) \neq 0)).$$

We can reduce this to the already familiar case by using Rabinowitsch's trick. Let y be a new variable and set $f = 1 - y\Pi_{i=1}^{s_2} f_i$.

$$F(x^{[0]}) = \exists x^{[1]} (\bigwedge_{i=1}^{s_1} (f_i(x^{[0]}, x^{[1]}) = 0)) \wedge (f(x^{[0]}, x^{[1]}) = 0).$$

Now we know how to eliminate the existential quantifier from here, and so we are done. Note that the maximum degree of the polynomials $g_1 \ldots, g_t$ we get by the procedure described in the previous lecture, is still at most $(sd)^n$.

OR of ANDs with an Existential Quantifier

Now suppose the formula we are given is of the form:

$$F(x^{[0]}) = \exists x^{[1]}(\bigvee_{i=1}^{A} \varphi_i(f_1, ..., f_s))$$

where $\varphi_i(f_1, ..., f_s) = (\bigwedge_{j=1} s(f_j = 0) = A_{ij})$, where A_j is just the truth assignment to the variables corresponding to each clause.

Note that the existential quantifier and the OR can trade places, and so we get:

$$F(x^{[0]}) = (\bigvee_{i=1}^{A} \exists x^{[1]}\varphi_i(f_1, ..., f_s))$$

Now, using what we have already seen, we can transform

$$\psi_i = \exists x^{[1]}\varphi_i(f_1, ..., f_s)$$

into formulas that do not involve quantifiers, on polynomials $g_1, ..., g_t$ of maximum degree $(sd)^n$. And so we get:

$$F(x^{[0]}) = \bigvee_{i=1}^{A} \psi_i(g_1, ..., g_t)$$

Boolean Formula with One Existential Quantifier

When we are given a boolean formula on s variables with an existential quantifier, we can examine all 2^s assignments to the variables and see which ones satisfy the formula. So potentially we can replace the formula with an OR of ANDs (DNF formula), where each AND clause specifies a truth assignment to the variables. This will introduce up to 2^s clauses.

Having this many clauses was something we wanted to avoid. However, it turns out that not all 2^s assignments to the variables are always realizable, by the following theorem, which is a derivative of the famous Bézout's theorem:

Theorem 17.1 *Let $f_1, ..., f_s$ be polynomials of degree at most d in variables $(x_1, \dots, x_n)$. Let*

$$S = \{(I_1, ..., I_s) | \exists (x_1 \dots, x_n) \text{ s.t. } I_i = \text{“}f_i(x_1, \dots, x_n) = 0''\}$$

Then $|S| = O((sd)^n)$.

By this theorem, we obtain two upper bounds on the number of clauses in the boolean formula where the existential quantifier is removed: 2^s and $O((sd)^n)$.

The proof of the theorem is actually constructive and gives us an algorithm for removing the existential quantifier. We will omit it for now.

Each of the $\min(2^s, O((sd)^n))$ clauses in the quantified DNF formula we obtain is an AND of truth statements about s polynomials f_i of degree at most d. By what we have already seen, it can be replaced by a quantifier-free formula which is an OR of $\min(2^s, O((sd)^n))$ clauses on $(sd)^{n^2}$ polynomials of degree $(sd)^n$.

Quantified Boolean Formula

Theorem 17.2 *Suppose we are given a Boolean formula with w quantifiers:*

$$F(x^{[0]}) = Q_1 x^{[1]} \dots Q_w x^{[w]} \varphi(f_1(x_1, \dots, x_n) = 0, \dots, f_s(x_1, \dots, x_n) = 0)$$

We can eliminate the quntifiers to obtain a Boolean formula with $(sd)^{O(n^{2w})}$ clauses, in $\min(2^{sw}, (sd)^{O(n^{2w})})$ polynomials of degree at most $sd^{O(n^{2w})}$.

Proof We will begin by examining a formula of the form:

$$F(x_1,\ldots,x_k) = \forall z_1,\ldots,z_l \exists y_1,\ldots,y_m \varphi(f_1(x_i,y_i,z_i)=0,\ldots,f_s(x_i,y_i,z_i)=0)$$

First, we rewrite $F(x_1,\ldots,x_k)$ as

$$F(x_1,\ldots,x_k) = \neg(\exists z_1,\ldots,z_l(\neg\exists y_1,\ldots,y_m \varphi(f_1(x_i,y_i,z_i)=0,\ldots,f_s(x_i,y_i,z_i)=0)))$$

Then, we eliminate the existential quantifier inside and obtain a formula of the form:

$$F(x_1,\ldots,x_k) = \neg(\exists z_1,\ldots,z_l(\neg\varphi'(g'_1(x_i,z_i)=0,\ldots,g'_{t'}(x_i,z_i)=0)))$$

This formula has one quantifier, $O((sd)^n)$ clauses each involving $(sd)^{n^2}$ polynomials of degree at most $(sd)^n$. (The total number of polynimials is $O((sd)^{n^2+n}) = (sd)^{O(n^2)}$.) Now, we reiterate and remove the remaining quantifier, to obtain:

$$F(x_1,\ldots,x_k) = \psi(g_1,\ldots,g_t),$$

a formula that involves $((sd)^{O(n^2)} * (sd)^n)^n = (sd)^{O(n^3)}$ clauses, each on $((sd)^{O(n^2)} * (sd)^n)^{n^2} = (sd)^{O(n^4)}$ polynomials of degree $((sd)^{O(n^2)} * (sd)^n)^n = (sd)^{O(n^3)}$. By iterating this step, we obtain our bounds. ■

Thus we have seen that owing to Bezout's theorem, we can do quantifier elimination in time that is polynomial in the number of polynomials given. Next time we will talk some more about Bezout's theorem.

6.966 Algebra and Computation November 9, 1998

Lecture 18

Lecturer: Madhu Sudan *Scribe: Salil Vadhan*

Bézout's Theorem and Applications

In this lecture, we will prove Bézout's theorem for curves in the plane and describe a couple of its applications in coding theory.

Clarification on Dimension and Degree

In the last lecture, we defined the dimension and degree of ideals and varieties. There was an error in the definition we gave last time which we correct now.

Definition 18.1 *Let F be a field and I an ideal in $F[x_1, \ldots, x_n]$. Then the* Hilbert polynomial *of I is the function $\mathrm{HP}_I : \mathbb{N} \to \mathbb{N}$ given by $\mathrm{HP}_I(d)$ = the number of monomials of degree at most d* **not** *in $\mathrm{LT}(I)$.*

Fact 18.2 *For every I, HP_I is a eventually a polynomial, that is, there is a polynomial $p(x) \in \mathbb{Q}[x]$ such that for sufficiently large $d \in \mathbb{N}$, $\mathrm{HP}_I(d) = p(d)$.*

Definition 18.3 *The* dimension *of I, denoted $\dim(I)$, is the degree of HP_I. The* degree *of I, denoted $\deg(I)$, is the leading coefficient of HP_I times $(\dim(I)!)$. The* dimension *(resp.,* degree*) of a variety $V \subset F^n$ is defined to be the dimension (resp., degree) of $I(V)$.*

To motivate these definitions, it is helpful to think of monomial ideals, because $\mathrm{LT}(I)$ is equal to the set of monomials in I. One can easily verify the following examples:

Example 1 (union of two axes in F^2) *$I = (x^2 y) \subset F[x, y]$. The monomials outside of I are exactly those that have degree 1 or 0 in x or have degree 0 in y. The number of these of degree at most d is easily seen to be $3d$, so $\dim(I) = 1$ and $\deg(I) = 3 \cdot 1! = 3$. Note that $V(I)$ is is the union of the two coordinate axes, so dimension 1 agrees with to our geometric intuition.*

Example 2 (the origin in F^2) *$I = (x, y^4) \subset F[x, y]$. The only monomials outside of $\mathrm{LT}(I)$ are $\{1, y, y^2, y^3\}$ so $\mathrm{HP}_I(d) = 4$ for all $d \geq 3$, $\dim(I) = 0$ (corresponding to the fact that $V(I)$ is a point), and $\deg(I) = 4 \cdot 0! = 4$.*

Example 3 (monomial ideals in $F[x, y]$) *For a monomial ideal I in $F[x, y]$, $V(I)$ is either (i) $\emptyset$, (ii) the origin, (iii) the x-axis, (iv) the y-axis, (v) the union of the x and y axes, or (vi) the entire plane (depending on which "types" of monomials occur among the generators of I — where a monomial can be of one of the following four types: (a) 1, (b) x^i, $i > 0$ (c) y^i, $i > 0$ or (d) $x^i y^j$ $i, j > 0$). By calculating the rate of growth of HP_I in each of these cases, it is easy to see that $\dim(I)$ will be (i) $-\infty$ (by convention), (ii) 0, (iii) 1, (iv) 1, (v) 1, or (vi) 2, respectively, regardless of what monomials are used to generate the ideal. The degree, however, does depend on the particular generating monomials. If however, we take $I = I(V)$ for one of these varieties, the lowest degree generators will always be used (namely (i) (1), (ii) (x, y), (iii) (x), (iv) (y), (v) (xy), or (vi) (0), respectively) and the degree of these varieties will be (i) 0 (by convention), (ii) 1, (iii) 1, (iv) 1, (v) 2, or (vi) 1, respectively.*

Example 4 (hypersurfaces) *$I = (f) \subset F[x_1, \ldots, x_n]$. Then $\mathrm{LT}(I)$ is the set of monomials which are divisible by $\mathrm{LT}(f)$. The number of these of degree at most d is $\binom{d+n-\deg(f)}{n}$, so $\mathrm{HP}_I(d) = \binom{d+n}{n} - \binom{d+n-\deg(f)}{n}$, whose leading term is $\deg(f) \cdot d^{n-1}/(n-1)!$. Thus $\dim(I) = n - 1$ (which corresponds to the fact that $V(I)$ is a hypersurface) and $\deg(I) = \deg(f)$.*

Before doing our final computation of dimension and degree, we give another characterization of the Hilbert polynomial. Let P_d be the set of all polynomials in $F[x_1, \ldots, x_n]$ of degree at most d, and for any ideal I, let I_d be the set of polynomials of degree at most d in I. Then P_d is a vector space over F of dimension $\binom{n+d}{d}$ and I_d is a subspace of P_d.

Proposition 18.4 *The number of monomials of degree at most d in $LT(I)$ equals the dimension of I_d as a vector space over F. Equivalently, $\mathrm{H}P_I(d)$ equals the codimension*[11] *of I_d in P_d.*

Proof Suppose there are s monomials of degree at most d in $\mathrm{LT}(I)$. Then there are polynomials $p_1, \ldots, p_s$ in I whose leading terms are these monomials. These polynomials are all of degree at most d and are linearly independent because they have distinct leading monomials. Hence, the dimension of I_d is at least s.

Conversely, suppose the dimension of I_d is t. Let $p_1, \ldots, p_t$ be a basis for I_d. We will construct polynomials $q_1, \ldots, q_t$ in sequence satisfying

1. $\mathrm{S}pan(q_1, \ldots, q_i) = \mathrm{S}pan(p_1, \ldots, p_i)$ for every i, and
2. The q_i's have distinct leading monomials.

We can take q_1 to be p_1. Now inductively assume that we have found $q_1, \ldots, q_i$. If the leading monomial of p_{i+1} is different than all the leading monomials of $q_1, \ldots, q_i$, then we take $q_{i+1} = p_{i+1}$ and we're done. Otherwise, we can subtract a scalar multiple of one of the q_j's to obtain p'_{i+1} whose leading monomial is smaller (with respect to a fixed ordering on monmials) than that of p_i and still satisfies $\mathrm{S}pan(q_1, \ldots, q_i, p'_{i+1}) = \mathrm{S}pan(p_1, \ldots, p_{i+1})$. We keep eliminating the leading monomial until we obtain p^*_{i+1} whose leading monomial is distinct from all the q_j's, and then we can take $q_{i+1} = p_{i+1}$. This process must stop after finitely many steps because each time we reduce the leading monomial. Once we have finished, the leading monomials of $q_1, \ldots, q_t$ give us t distinct monomials of degree at most d in $\mathrm{LT}(I)$.

For the "equivalently" part of the proposition, note that $\mathrm{H}P_I(d)$ is defined to be $\binom{n+d}{d}$ minus the number of monomials of degree at most d in $\mathrm{LT}(I)$, and the codimension of I_d in P_d is $\binom{n+d}{d}$ minus the dimension of I_d as a vector space over F. ■

Now let $V \subset F^n$ be any variety, and let F^V denote the set of functions from V to F. For every d, there is a linear map $\psi_{V,d}$ from P_d to F^V which simply takes a polynomial $p \in P_d$ and maps it to $p|_V \in F^V$. The kernel of this map are those polynomials of degree at most d which vanish on all of V, i.e. the kernel is I_d for $I = I(V)$. Thus the dimension of the image (as a vector space over F) is exactly the codimension I_d in P_d, and by Proposition 18.4, we have

Corollary 18.5 *For $I = I(V)$, $\mathrm{H}P_I(d)$ equals the dimension of $\mathrm{Im}(\psi_{V,d})$ as a vector space over F.*

Now we can calculate the dimension and degree of finite sets.

Example 5 (finite sets) *Let V be a finite set of m distinct points in F^n. We will show that V has dimension 0 and degree m. F^V has dimension m, so $\mathrm{Im}(\psi_{V,d}) \subset F^V$ has dimension at most m. By Corollary 18.5, for $I = I(V)$, $\mathrm{H}P_I(d) \leq m$ for any d, so the dimension of V is zero and the degree of V is at most m. To show that the degree of V actually equals m, we need to show that for sufficiently large d, $\psi_{V,d}$ is surjective. To see this note that for every point $a \in V$, we can find a polynomial p_a of degree $m-1$ which is nonzero at a but vanishes on the all the $m-1$ points in $V \setminus \{a\}$. These polynomials $\{p_a|_V\}$ span the space of all functions from V to F, so $\psi_{V,d}$ is surjective for all $d \geq m-1$.*

Statements of Bézout's Theorem

The most general version of Bézout's theorem is the following:

Theorem 18.6 (Bézout) *For any two varieties V and W (satisfying some "niceness" conditions that are beyond the scope of this course) over an algebraically closed field,*

$$\deg(V \cap W) \leq \deg(V) \cdot \deg(W).$$

[11] If V is a subspace of W then the *codimension* of V in W is $\dim(W) - \dim(V)$.

Applying this to the special case of hypersurfaces and finite sets of points (Examples 4 and 5), we obtain

Theorem 18.7 (Bézout for hypersurfaces) *Let F be an algebraically closed field, and suppose $f_1, \ldots, f_k \in F[x_1, \ldots, x_n]$ are of degrees $d_1, \ldots, d_k$, respectively. Then the number of common zeroes of $f_1, f_2, \ldots, f_k$ is either infinite at most $\prod_i d_i$*

We will prove a special case of this for curves in the plane. We want a result that also holds for finite fields, so the case that the number of common zeroes is infinite will be replaced by something meaningful for finite fields.

Theorem 18.8 (Bézout in the plane) *Let F by any field (not necessarily algebraically closed) and let $f, g \in F[x, y]$ be polynomials of degree m and n, respectively. Then f and g either share a nontrivial common factor or have at most mn common zeroes in F^2*

Proof of Bézout's theorem in the plane

First, we observe that it suffices to prove Theorem 18.8 for infinite fields: If we're given two polynomials $f, g \in F[x, y]$ over a finite field F which have more than mn common zeroes in F^2, they they also have more than mn common zeroes in L^2 for any infinite field L containing F. Assuming Theorem 18.8 holds for infinite fields, we can deduce that f and g have a nontrivial common factor $h \in L[x, y]$. The degree of h in one of the variables, say x, must be greater than 0. This implies that $\mathrm{Res}_x(f, g) = 0$, where the resultant is computed in $L(y)$. But the resultant will be the same if we compute it in $F(y)$, so we conclude that f and g have a common factor in $F(y)[x]$. By Gauss' Lemma, we deduce that they have a common factor in $F[x, y]$

Now let's proceed to the proof of Theorem 18.8 for infinite fields. Suppose f and g have more than mn zeroes. Let S be any set of $mn+1$ distinct zeroes of f and g. We first perform a random linear transformation to make the second coordinates of all the elements of S distinct. Specifically, choose α uniformly at random from some subset of L of size greater than $(mn+1)^2$ and let $f'(x, y) = f(x, y + \alpha x)$, $g'(x, y) = g(x, y + \alpha x)$, $S' = \{(u, v - \alpha u) \colon (u, v) \in S\}$. Then, for every α, every element of S' is common zero of f' and g', and f' and g' have a common factor iff f and g do. Moreover, we claim that, with nonzero probability over the choice of α, no two elements of S' have the same second coordinate. (For every pair of points $(u_1, v_1), (u_2, v_2) \in S$, the probability that $v_1 - \alpha u_1$ equals $v_2 - \alpha u_2$ is smaller than $1/(mn+1)^2$. Since there are fewer than $(mn+1)^2$ pairs to consider, the claim follows by a union bound.) So fix any α for which no two elements of S' have the same second coordinate.

Now consider $p(y) = \mathrm{Res}_x(f', g')$, which is a polynomial in y. We have argued in previous lectures that p is of degree at most $\max\{m, n\}^2$. In fact, we will show later that p is of degree at most mn; assuming this fact, we continue with the proof. For every point $(u, v) \in S'$, $p(v) = \mathrm{Res}_x(f'(x, v), g'(x, v)) = 0$, because $f'(x, v)$ and $g'(x, v)$ have the common zero $x = u$. Thus p has at least $mn+1$ distinct roots, and must be the zero polynomial. In other words, $\mathrm{Res}_x(f', g') = 0$, so f' and g' have a common factor in $F(y)[x]$, implying a common factor in $F[x, y]$ by Gauss' Lemma.

This completes the proof of Theorem 18.8, except for the following fact about resultants.

Proposition 18.9 *If $f, g \in F[x, y]$ are polynomials of total degree at most m and n, respectively, then $\mathrm{Res}_x(f, g)$ is a polynomial of degree at most mn in y.*

Proof We can write $f(x, y) = a_0(y) + a_1(y)x + \cdots + a_m(y)x^m$ and $g(x, y) = b_0(y) + b_1(y)x + \cdots + b_n(y)x^n$, where a_j (resp., b_j) is a polynomial of degree at most $m - j$ (resp., $n - j$). Recall that $\mathrm{Res}_x(f, g)$ is the determinant of the $(m + n) \times (m + n)$ matrix M given by

$$M_{ij} = \begin{cases} a_{i-j}(y) & \text{if } j \leq n \\ b_{n+i-j}(y) & \text{if } j > n, \end{cases}$$

where the $a_k(y)$ and $b_k(y)$ are defined to be 0 when k is out of bounds. Thus,

$$\deg(M_{ij}) \leq \begin{cases} m - (i - j) = m - i + j & \text{if } j \leq n \\ n - (n + i - j) = j - i & \text{if } j > n, \end{cases}$$

Since $\det(M) = \sum_\sigma \left(\pm \prod_{j=1}^{m+n} M_{\sigma(j)j}\right)$, where the sum is over all permutations $\sigma\colon [m+n] \to [m+n]$, we have

$$\begin{aligned}
\deg(\det(M)) &\le \max_\sigma \left(\sum_{j=1}^{m+n} \deg(M_{\sigma(j)j}) \right) \\
&\le \max_\sigma \left(\sum_{j=1}^{n} (m + j - \sigma(j)) + \sum_{j=n+1}^{n+m} (j - \sigma(j)) \right) \\
&= \max_\sigma \left(mn + \sum_{j=1}^{n+m} j - \sum_{j=1}^{n+m} \sigma(j) \right) \\
&= mn,
\end{aligned}$$

where in the last equality we use the fact that σ is a permutation. ■

Application 1 — Goppa Codes

Bézout's theorem has many applications in complexity and coding theory. It can be viewed as a generalization of the fact that a degree d polynomial has at most d roots (take $f(x,y) = y - p(x)$ and $g(x,y) = y$ in Theorem 18.8). This leads to a natural generalization of Reed–Solomon codes, called Goppa codes.

Recall that the Reed–Solomon code identifies "messages" with polynomials p of degree at most $k-1$ over some finite field $\mathbb{F}_q$. The codeword associated with a message/polynomial p is $(p(a_1), \ldots, p(a_n))$ where $\{a_1, \ldots, a_n\}$ is a fixed subset of $\mathbb{F}_q$. n is called the *block length* of the code. The important property of RS codes is that every two distinct codewords disagree in at least $d = n - k + 1$ places; d is called the *distance* of the code. Our messages (polynomials of degree at most $k-1$) can we thought of as strings of length k over $\mathbb{F}_q$, so k is called the *rate* of the code. Thus, the relationship among the rate, distance, and block length of RS codes can be summarized as $k = n - d + 1$. The main disadvantage of RS codes is that the alphabet size q must be at least as large as the block length n. (because $\{a_1, \ldots, a_n\}$ must be a subset of $\mathbb{F}_q$.) This means that if one wants to encode long messages, one has to use a large alphabet. Large alphabets have the disadvantage that one can corrupt an entire symbol by only corrupting one bit in a binary representation of the symbol, and hence by corrupting only a $1/(\log_2 q)$ fraction of the bits in a binary representation of a codeword, every symbol in the codeword can be corrupted and error-correction is impossible.

Reed-Solomon codes can be thought of as evaluating low-degree polynomials on a line (namely, the elements of $\mathbb{F}_q$), which has only q points. An idea for increasing the block-length relative to the alphabet size would be to evaluate low-degree polynomials on a curve C with more than q points. We will still need the properties that no two distinct low-degree polynomials agree on C in many places and that there are many distinct low-degree polynomials on C — Bézout's theorem will enable us to prove these properties.

Fix an irreducible bivariate polynomial $f(x,y)$ of degree $\ell < q$ over $\mathbb{F}_q$, and let $C \subset \mathbb{F}_q^2$ be the zero-locus of f (i.e., $C = V((f))$). There exist such curves containing nearly $q\ell$ points (since f is of degree ℓ, the most we can hope for is ℓ choices for y for every value of x), and it is known how to find such curves efficiently. So let us assume that we have a curve with $n = |C| \approx q\ell$. Our messages will be interpreted as bivariate polynomials $g(x,y)$ of degree at most m (where $\ell \le m < n/\ell$) over $\mathbb{F}_q$, and the corresponding codeword will be the evaluation of g on all points of C. We need to be careful because two polynomials g_1 and g_2 can agree everywhere on C. However, this means that the polynomials $g_1 - g_2$ and f have at least $n > m\ell$ common zeroes, so by Bézout's theorem $f | (g_1 - g_2)$.[12] Thus the number of distinct codewords we obtain is the number of bivariate polynomials of degree m ($= q^{\binom{m+2}{2}}$) divided by the number of bivariate polynomials of degree m which are divisible by f ($q^{\binom{m-\ell+2}{2}}$). In other words, the dimension of the space of messages we

[12] If we were over an algebraically closed field, this would follow from the Nullstellensatz: $g_1 - g_2$ is identically zero on $C = V((f))$, so $(g_1 - g_2) \in I(V((f))) = \mathrm{Rad}((f)) = (f)$.

can encode, or *rate* of the code, is

$$k = \binom{m+2}{2} - \binom{m-\ell+2}{2} = \ell m + 1 - \binom{\ell-1}{2}.$$

Now we analyze the distance of the code. The same Bézout analysis we did above shows that no two polynomials g_1 and g_2 of degree m can agree on more than ℓm points of C unless $f|(g_1 - g_2)$ (in which case g_1 and g_2 agree everywhere on C and we consider them g_1 and g_2 to be the same codeword). Thus, the distance of the code is at least $d = n - \ell m$.

Summarizing, we relationship between rate, block length, and distance is now

$$k = n - d + 1 - \binom{\ell-1}{2}.$$

This is worse than what RS codes achieve by the defect $\binom{\ell-1}{2}$, but we have gained in that the alphabet size need only be about n/ℓ. If we take $\ell = \Theta(\sqrt[3]{n})$, q will be $\Theta(n^{2/3})$ (a substantial improvement over RS codes), and the defect is only $O(n^{2/3}) = o(n)$.

Generalizing these ideas (quite a bit further) eventually leads to *algebraic geometric codes*, which have no restriction on the alphabet size and satisfy

$$k \approx \left(1 - \frac{1}{\sqrt{q}}\right) n - d.$$

Application 2 — Decoding Reed–Solomon Codes

In general, the *decoding* problem for an error-correcting code is: Given a codeword which has been corrupted in some number of positions, find the message corresponding to the codeword (equivalently, find the uncorrupted codeword). In case there are so many errors that several messages could have produced the given string, the *list-decoding* problem is to find all possible messages that are consistent with the number of errors.

For Reed–Solomon codes, this can be reformulated as the following curve-fitting question: Given n points $(x_1, y_1), \ldots, (x_n, y_n)$, find a degree k polynomial p such that $p(x_i) = y_i$ for as many i possible. A widely used approach to solving this problem is algebraic-geometric in nature:

1. Find a low-degree algebraic curve $f(x, y) = 0$ which goes through all these points (x_i, y_i),
2. Decompose the curve into its irreducible components (by factoring f), and
3. Hope that the one of these components is the graph of the polynomial p we're looking for (i.e. $y - p(x)$ appears as a factor of f).

Let's prove (using Bézout's theorem) that this approach works for some settings of parameters. We will show that it can find any polynomial p which agrees with at least $2\sqrt{n} \cdot k$ of the points. To do this, we will look for C described by a polynomial $f(x, y)$ of total degree at most $2\sqrt{n}$. A nonzero polynomial f of this degree such that $f(x_i, y_i) = 0$ for all i always exists, simply because each pair (x_i, y_i) imposes a homogeneous linear constraint on the coefficients of f and f has at $\binom{2\sqrt{n}+2}{2} > n$ coefficients. To find f, we can just solve this linear system. Now we only need to prove that any polynomial p such that $p(x_i) = y_i$ for enough of the i's will appear in a factor of f of the form $y - p(x)$. Well, f and $y - p(x)$ are two algebraic curves of degree $2\sqrt{n}$ and k, respectively, that agree in many points. By Bézout's theorem, they share a common factor if they agree in at least $2\sqrt{n} \cdot k$ points. Since $y - p(x)$ is irreducible, this implies $y - p(x)$ will occur among the irreducible factors of f, and the algorithm works.

Note that for $k \ll \sqrt{n}$, this is list-decoding far beyond the range in which the answer is unique — there is guaranteed to be a unique decoding only when the the number of uncorrupted positions is greater than $(n+k)/2$. The best known result for this problem, due to Guruswami and Sudan (1998), allows list-decoding in polynomial-time when the number of uncorrupted positions is greater than $\sqrt{nk}$.

6.966 Algebra and Computation	November, 16 1998
Lecture 19	
Lecturer: Madhu Sudan	*Scribe: Adam Klivans*

19.1 Introduction

In this lecture we will introduce several algebraic models of computation including algebraic circuits, decision trees, branching programs and straight line programs. We will focus on three central questions:

1 How can we transfer known results in the boolean world to the algebraic setting?

2 How are lower bound techniques applicable to the algebraic case?

3 What is the relationship of algebraic complexity classes to their boolean counterparts?

Furthermore, we will prove a simple result due to Ben-Or and Cleve showing how to simulate a circuit of depth d with a straightline program of size 4^d using only 3 registers.

19.1.1 Preliminaries

We will consider non-uniform models of algebraic computation over some ring R. The model of computation computes a function $\phi\colon R^n \to R^m$. I.e. there will be n inputs and m outputs.

19.2 Algebraic Circuits

Definition 19.1 *An algebraic circuit is a directed acyclic graph. Each node, or gate, computes some ring operation (i.e., $+, -, *$) of the incoming edges. The output of each node "contains" the value of the ring operation applied to the values in the incoming edges.*

In the case where the ring where are working over is a field and we allow the division operation, our circuits correspond precisely to the set of all rational functions over the field.

In the boolean case, these circuits compute all boolean functions (assuming we allow an input to be hardwired to 1 for negations). This is not so in the general algebraic case. For example, we have no equality or inequality operator. We could allow equality gates, i.e., a gate that outputs 1 if and only if its input is equal to 0. Or, if we are working over an ordered field, we could add inequality gates, i.e., a gate that outputs 1 if and only if its input is greater than or equal to 0.

19.3 Algebraic Decision Trees

Another frequently studied model of computation is the algebraic decision tree:

Definition 19.2 *An algebraic decision tree is a binary tree with two types of nodes: query nodes and leaf nodes. Each query node is labelled with a polynomial in the input variables $p(x_1, \ldots, x_n)$. We "branch" depending upon whether or not $p(x_1, \ldots, x_n) = 0$. Each leaf node is then labelled as reject or accept. One of the nodes is designated as the start node. The decision tree accepts (rejects) an input if the corresponding path in the tree leads to an accept (reject) leaf.*

One way of measuring the complexity of a decision tree is to look at the maximum over all paths from root to leaf of the sum of the degrees of the polynomials at each node. To measure a function's algebraic decision tree complexity, we then take the minimum of this quantity over all decision trees computing the function. Notice that if each polynomial is a linear polynomial then this measure of complexity coincides with the standard notion of boolean decision tree complexity.

19.4 Straight Line Programs

Another interesting model of algebraic computation is the straight line program:

Definition 19.3 *A straight line program is a sequence of instructions using the variables (registers) $x_1, \ldots, x_n$, $A_1, \ldots, A_l, O_1, \ldots, O_m$ Each x_i corresponds to an input variable and each O_i corresponds to an output.*

Each instruction must be one of the following:

1 $A_i = A_j$ **op** A_k where **op** $\in \{+, -, *\}$

2 $A_i = A_j$ **op** x_k where **op** $\in \{+, -, *\}$

3 $A_i = r$ where r is some ring element.

4 $O_i = A_j$ for some i and j.

The **time** complexity of a straight line program corresponds to the number of instructions and the **space** complexity corresponds to the number of registers used.

Before we begin looking at straight line programs, we introduce another model of computation: the branching program.

19.5 Branching Programs

Definition 19.4 *A branching program is a directed acyclic graph where all nodes are labelled by variables, except for the output nodes which are labelled either* 1 *or* 0*. The nodes labelled by variables are called query nodes. Each query node has a label x_i and has two outgoing edges, one labelled* 1 *and one labelled* 0 *(corresponding to whether or not x_i equals* 0 *or* 1*).*

19.5.1 Branching Programs and Straight Line Programs

Here is a simple example which relates straight line programs and branching programs:

Example: Assume that we are given a straight line program over $GF(2)$ using registers $A_1, \ldots, A_\ell$ and having d instructions. Notice that there are exactly 2^ℓ different "configurations" that the registers could be in. We can simulate the straight line program by a branching program divided into d "levels." Each level corresponds to one instruction in our straight line program. Level i contains 2^ℓ nodes, each one corresponding to possible configuration of our registers. We connect node k in level i to node j in level $i+1$ if the ith instruction in our straight line program would change the configuration of our registers in node k to the configuration represented by node j given that the input in the ith instruction equals 1 or 0 (if the input is not looked at in the ith instruction then both edges will point to the same node).

Definition 19.5 *Let B be a branching program. We say B is levelled if it can be partitioned into sets of nodes $S_1, \ldots, S_n$ such that all of the edges leaving S_i enter only the nodes in S_{i+1}.*

Definition 19.6 *We say a levelled branching program has width w if the size of the largest set in the partition is less than or equal to w.*

Proposition 19.7 *From the above example we see that any straight line program using ℓ registers and having d instructions can be simulated with a levelled branching program of width 2^ℓ.*

We can use levelled branching programs to give an interesting non-uniform measure of the complexity of a function by looking at the smallest width over all levelled branching programs computing a function. For example, the parity function has a width 2 branching program:

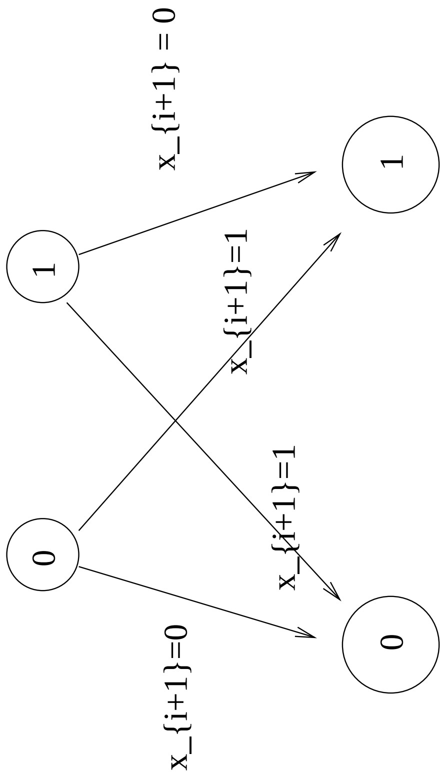


Figure 19.1: The ith level of a branching program computing Parity

Considerable effort was put into understanding what functions cannot be computed with small width, polynomial size levelled branching programs. **Put reference here** proved that the majority function on n variables requires exponential size width-3 branching programs. Finally Barrington proved that width-5 branching programs correspond precisely to non-uniform NC_1.

19.6 Algebraic Circuits and Straight Line Programs

In this section we will prove the following result of Ben-Or and Cleve:

Theorem 19.8 *If a function f is computable by an algebraic circuit of depth d with the $+, -, *$ operations then f is also computed by a straight line program of size 4^d with 3 registers*

Proof: We will assume that every instruction in our straight-line program will have the form

$$A_i \Leftarrow A_j + (A_k * x_l)$$

Let $f(x_1, \ldots, x_n)$ be some mapping from R^n to R that can be computed by a circuit of depth d. Assume for a moment that using registers A_1, A_2 and A_3 we can compute (stored in our output registers) $A_1 + f(x_1, \ldots, x_n)A_2, A_2, A_3$. Then with $A_1 = 0, A_2 = 0$, and $A_3 = 0$ given as input we would have as output $f, 1, 1$. We will prove that we can achieve this construction by induction on the depth of the circuit computing the function.

- **Base Case:** This can clearly be done for a depth 0 circuit.
- **Inductive Case:**

Assume $f = f_1 + f_2$. Then by induction we have a straight line program with three variables A_1, A_2, A_3 such that after our straight line program executes, these registers take on the values $A_1 + f_1A_2, A_2, A_3$. Then we use these values as inputs to our straight line program for f_2 and we receive as output $A_1 + f_1A_2 + f_1A_2 = A_1 + (f_1 + f_2)A_2, A_2, A_3$.

So far we have not used the full power of our inductive hypothesis. The case for multiplication will be a little more involved:

Assume $f = f_1 \times f_2$.

1 Use, via our inductive hypothesis, a straight line program which on input A_1, A_2, A_3 produces as output $A_1, A_2, A_3 + f_1A_2$.

2 Now feed these registers into our straight line program for f_2 to obtain as output $A_1 + f_2(A_3 + f_1A_2), A_2, A_3 + f_1A_2$.

3 Use the output from (2) as input to our SLP for $-f_1$ to get as output $A_1 + f_2(A_3 + f_1A_2), A_2, A_3 + f_1f_2 - f_1f_2 = A_3$.

4 Finally use the output from (3) as input to our SLP for $-f_2$ to get as output $A_1 + f_1f_2A_2, A_2, A_3$.

19.7 Uniform Models for Algebraic Computation

In the next lecture we will introduce the Blum-Shub-Smale model of algebraic computation and look at the corresponding complexity classes.

6.966 Algebra and Computation November, 18 1998

Lecture 20

Lecturer: Madhu Sudan *Scribe: Zulfikar Ramzan*

20.1 Introduction

In this lecture we continue our discussion of algebraic notions of complexity. We define the notion of a Turing Machine over an arbitrary ring, and consider the analogs of various complexity classes in this new model. We discuss some of the relevant open problems in this algebraic setting, and examine what solving them would imply in the traditional setting. We demonstrate a natural complete problem for one of the new complexity classes we consider. We also examine a slightly different way to think about computation in this setting, and using this new way of looking at things, we show that a fundamental and well studied problem from pure mathematics is in fact undecidable.

20.2 Turing Machines over Arbitrary Rings

As described in the previous lecture, we can define Turing Machines over arbitrary algebraic rings. The traditional Turing Machine model, which examines the ring Z_2, is a special case of this general model. We now define a Turing Machine over an arbitrary ring R. These general algebraic Turing Machines have a two way infinite tape where each cell contains an element of the ring R. We often refer to specific parts of the tape by the notation $Tape(index)$. These general Turing Machines also have a set of states. Some of these states specify a polynomial – and each state falls into one of 5 different categories:

1. Accept States - corresponds to the traditional definition of accept state.
2. Reject States - corresponds to the traditional definition of reject state.
3. Branch States - evaluate the polynomial specified in the definition of the state on the value $Tape(0)$. Branch to another state depending on whether or not the polynomial evaluated to 0.
4. Shift State - shift your entire tape over to the right or left; e.g. for all i: $Tape'(i) = Tape(i+1)$ or $Tape'(i) = Tape(i-1)$.
5. Computing State - perform a computation of the polynomial specified in the definition of the state on value appearing at $Tape(0)$.

Each of these states may contain two different types of information:

1. A finite number of bits describing the name of a state, and other basic information about the state.
2. An finite number of elements from the ring R. These elements actually specify the polynomial whose evaluation on some particular element determines the next action to be taken. For example, if you are in a branch state, depending on what the polynomial evaluates to, you may decide whether or not to take the branch. These polynomials are finitely specified and have a constant arity.

One thing to notice about this definition is that there is conceivably an immense amount of power such a Turing Machine could have if it can record arbitrary ring elements in its states. If we consider the ring of Real numbers, then it is possible to encode an infinitely long string into a state under this model by, for example, having a trancendental number appear as a coefficient in the polynomial specified by a state. Even though this appears to be an overwhelming amount of information, that view is deceptive. In particular, even if you have an infinitely long number encoded in a state, accessing some particular bit of the number, or some particular portion of it, may not be an easy task (because you don't explicitly have the power to determine if a number is greater than or equal to 0). And so it's possible that the task of accessing a particular digit in the number may not be achievable in polynomial time.

We know that a Turing Machine over an arbitrary ring R has a two-way infinite tape which may contain arbitrary elements of the ring R. Since we *do not* place any kinds of restrictions on these elements, it may in fact be possible to place a single element which does not have a finite representation (such as $\sqrt{2}$), on a single cell of the tape, which deceptively appears to give it an enormous amount of power. Having defined these Turing Machines over arbitrary rings, we now shift our focus to the relationships between these arbitrary machines and the classical Turing Machine (which we can think of as working over the ring Z_2). We explore the algebraic analogs of some of the classical complexity theoretic notions.

20.2.1 Algebraic Turing Machines as Universal Controllers

The first analogous notion between classical complexity theory and algebraic complexity theory that we consider is that of the *Universal Turing Machine.* It's not hard to see that we can consider the notion of a *Universal Turing Machine* even in this algebraic setting. We can design a Universal Turing Machine UTM_R over a ring R which takes as input:

1. The description of a Turing Machine M_R over the ring R.

2. An input x to this Turing Machine M_R.

The Universal Turing Machine should then simulate M_R on input x and output the same thing as M_R would have. The design of such a Universal Turing Machine is not too difficult. We now move on to consider the algebraic analog of decision problems.

20.2.2 Decision Problems

We'll be interested in decision problems which are a subset of instances of Algebraic Turing Machines. In particular consider an infinite family of functions $\{f_n\}_{n\geq 0}$ where each $f_n : R^n \rightarrow R$ is a function induced by an Algebraic Turing Machine that operates over the ring R. We can think of these functions as decision problems where we can think of x such that $f(x) = 1$ as a *yes* instance, and everything else as a *no* instance.

20.2.3 Comparison to Traditional Complexity Theoretic Models

We now compare these algebraic models and their variants with the traditional $\{0, 1\}$ model, and we show that some surprising results can occur if we make minor, yet seemingly reasonable, modifications to these models. To start with let's define the notion of a polynomial time Turing Machine over a ring R to be one that runs in time polynomial in the input size – in this case when we say input size, we mean the number of cells used in the input tape. Now, we show that adding what appears to be a tiny, yet reasonable, amount of functionality to polynomial time Algebraic Turing Machines enables them to solve undecidable problems in polynomial time!

Solving Undecidable Problems!

Suppose we allow these Turing Machines the power to tell whether or not a given number is greater than or equal to 0. Call such Turing Machines: *Turing Machines with Inequality.* It turns out that even this tiny and reasonable amount of extra functionality enables these Turing Machines to solve undecidable problems in a polynomial amount of time. Here is the idea. We know that the class $P/poly$[13] contains undecidable problems.

Now, if we use algebraic Turing Machines over the Reals, then it's possible to encode a polynomial length advice string within a single state, as a single element of the ring. If we allow the computation of ≥ 0 then accessing the $i-th$ digit of this advice string becomes easy (if we don't allow the operation $\geq$ this task may

[13] recall that $P/poly$ is the class of languages C such that L is in C if there exists a polynomial time Turing Machine M and a sequence of strings $\{s_i\}_{i\in N}$ such that $|s_i| = i^{O(1)}$ and $x \in L$ if and only if $M(x, s_{|x|})$ accepts.

be very difficult). In light of this, it turns out that the Algebraic Turing Machine defined over the Reals can solve every problem in $P/poly$, thus they can solve undecidable problems.

The Turing Machine over the Reals with Inequality (≥ 0) is not a terribly well behaved model because it can solve classically undecidable problems in polynomial time. This model may not always be interesting depending on how you look at things. One thing we can do to make this model interesting is to consider specific problems and specific problem instances and see how well these Turing Machines solve these problems.

Considering Specific Problems

If we fix certain problems of interest, we can consider various questions about how well a Turing Machine over the Reals with Inequality might handle these problems. For example, some interesting questions we can consider in this model are:

1. Is the Mandelbrodt Set Decidable by a Turing Machine over the Reals with Inequality? (We will formally define and answer this question later in the lecture)

2. Does Linear Programming have a strongly polynomial time algorithm in this same setting?

So we can fix problems of interest and ask how well these problems can be solved by such Turing Machines. In the case of Linear Programming, we might define an instance of Linear Programming as taking input $< A, b >$ and trying to find a vector x such that each element of x is greater than 0 and $Ax \leq b$.

Defining Linear Programming in this manner begs the question of what constitutes a good encoding. For example, it's possible to encode an instance of Linear Programming in the space of a few ring elements (since our ring elements may be infinitely long, and can encode a lot of information). And in general, for any problem, it might be possible to encode an instance of the problem within a single real number. The best way to handle this question is to recall that we undergo a similar dillema in the traditional 0/1 case. In particular, under some encodings, even very hard problems become trivial to solve. So, all we can do is make a qualitative assessment that a particular encoding of a problem that we're considering is "reasonable" and that it makes "the most sense." Having studied the algebraic analogs to some fundamental classical notions, we are in a position to examine the corresponding complexity classes that occur.

Defining Analogs to Traditional Complexity Classes

We now define some traditional complexity classes in this new Algebraic setting, and consider some of the interesting problems that arise. First we consider the analog of the class NP. Traditionally, one can define NP to be the set of projections of languages on two inputs that are in P. We can carry this analogy over to the setting of Algebraic Turing Machines. Consider a language L that consists of pairs (x, y) where $x \in R^n$ and $y \in R^{poly(n)}$ for some polynomial $poly(n)$, and where L is described by some polynomial time $R - TM$ called M_R. (We use $R - TM$ to denote a Turing Machine which is defined over the ring R.) We now define the projection of L

Definition 20.1 *Let L be a language. Then the Projection of L, denoted $\Pi(L)$ is defined as:*

$$\Pi(L) \equiv \{x \mid \exists\ y \text{ such that } (x, y) \in L\} \tag{20.1}$$

We now formally define the class NP_R:

Definition 20.2 $NP_R \equiv \{\Pi(L) \mid L \in P_R\}$. *Where P_R is the set of languages decidable in polynomial time by some $R - TM$.*

Now that we've defined NP_R, we can ask the natural question: $NP_R =? P_R$? Just like in the traditional scenario, this is an open problem, and later we give strong evidence for why it is believed that $NP_R \neq P_R$.

We now describe some other problems that are open – first we consider the notion of restrictions of traditional complexity classes to binary settings:

Definition 20.3 *Let C be any complexity class; i.e. C is a collection of languages $\{L\}$ where $L \in R^*$ (where $R^* \equiv \bigcup_{n\geq 0} R^n$). Then, we define $0/1 - L$ as:*

$$0/1 - L \equiv L \bigcap \{0,1\}^* \tag{20.2}$$

And we define $0/1 - C$ as:

$$0/1 - C \equiv \{0/1 - L \mid L \in C\} \tag{20.3}$$

This notion gives rise to another open question: does $0/1 - P_R$ =? P?. It's possible that $0/1 - P_R$ is stronger than than P because, for example, you can compute 2^{2^n} in n steps with an $R - TM$ (e.g. in successive steps you can compute: $2, 2^2, 2^4, 2^8, \ldots$ via repeated squaring). Can such a computation help you? It's not so obvious because even if you can perform such a computation, it's hard to extract any particular piece, such as the $i - th$ bit of the computation efficiently if you aren't given the power to perform inequality checks. It turns out that we do know one thing about $0/1 - P_R$ if R is the Ring of Complex Numbers then:

Theorem 20.4 *Let $\mathcal{C}$ denote the ring of complex numbers. Then, $0/1 - P_{\mathcal{C}} \subseteq BPP$.*

This gives evidence that $0/1 - P_{\mathcal{C}}$ may not be bigger than P. We now look specifically at the class of Turing Machines defined over the Complex Numbers.

20.2.4 Turing Machines over the Complex Numbers

We consider the class $\mathcal{C}$ of Complex Numbers, and consider various complexity classes associated with Turing Machines that are defined over the complex numbers. Let's start by considering $0/1 - NP_{\mathcal{C}}$. This is an interesting class because you can have arbitrarily high precision computation, and arbitrarily high precision witnesses! In particular, we show that Hilbert's Nullstellensatz when defined appropriately over 0/1 is $NP_{\mathcal{C}}$-complete. First let's define an appopriate decision version of Hilbert's Nullstellensatz when defined over an arbitrary field K.

0/1-Hilbert's Nullstellensatz (HN)
Input: polynomials $f_1, \ldots f_n \subseteq K[x_1, \ldots x_n]$. Where each polynomial is defined on n variables $x_1, \ldots, x_n$ and is defined over the field K. Moreover, these polynomials have degree bound d.

Question: Are there inputs $a_1, \ldots, a_n$ such that $f_i(a_1, \ldots, a_n) = 0$?

Given this formulation, one can talk about $0/1 - HN_{\mathcal{C}}$. We will later show that $0/1 - HN_{\mathcal{C}}$ is $0/1 - NP_{\mathcal{C}}$ complete. Before doing so, let us examine some of the ramifications.

Theorem 20.5 $0/1 - NP_{\mathcal{C}} \subseteq PSPACE$

Proof Will follow directly from the proof that $0/1 - HN_{\mathcal{C}}$ is $0/1 - NP_{\mathcal{C}}$-complete. ∎

Theorem 20.6 *If $0/1 - P_{\mathcal{C}} = 0/1 - NP_{\mathcal{C}}$ then $NP \subseteq BPP$.*

Proof If $0/1 - P_{\mathcal{C}} = 0/1 - NP_{\mathcal{C}}$ then $0/1 - HN_{\mathcal{C}} \in P_{\mathcal{C}}$. Now, we can think of $3 - SAT$ as a special case of $0/1 - HN_{\mathcal{C}}$. So we have that $3SAT \in 0/1 - NP_{\mathcal{C}} = 0/1 - P_{\mathcal{C}} \subseteq BPP$. ∎

Theorem 20.7 *If $0/1 - P_{\mathcal{C}} \neq 0/1 - NP_{\mathcal{C}}$ then $P \neq PSPACE$.*

Proof If $0/1-P_{\mathcal{C}} \neq 0/1-NP_{\mathcal{C}}$ then $0/1-HN_{\mathcal{C}} \notin 0/1-P_{\mathcal{C}}$. Therefore, $P \neq PSPACE$ since $0/1-HN_{\mathcal{C}} \in PSPACE$. ■

It is open whether $0/1 - P_{\mathcal{C}} \neq 0/1 - NP_{\mathcal{C}}$ would imply that $P \neq NP$. There is a recent result that $0/1 - HN_{\mathcal{C}} \subseteq PH$ under the extended Riemann Hypothesis. [14]

We are now ready to state and prove our main theorem:

Theorem 20.8 *$0/1 - HN_{\mathcal{C}}$ is $0/1 - NP_{\mathcal{C}}$-complete.*

Proof It is easy to see that $0/1-HN_{\mathcal{C}} \in 0/1-NP_{\mathcal{C}}$ because we can make our witness the vector $a_1, \dots, a_n$ that makes all the polynomials evaluate to 0 (e.g. $f_i(a_1, \dots, a_n) = 0$. Now, to show that $0/1 - HN_{\mathcal{C}}$ is $0/1 - NP_{\mathcal{C}}$-hard, we show that we can create a Cook-like tableau and do verification in the usual way.

Consider the computation of a $\mathcal{C} - \mathcal{TM}$ that decides a language in $0/1 - NP_{\mathcal{C}}$. For each time step $t = 0, t = 1, \dots, t = T$ we can consider the usual configurations on the Turing Machine tape, and ask ourselves the usual question of whether or not this computation is valid. More specifically to this case, we need to ask whether or not this computation can be expressed in terms of low degree polynomials. What kinds of information are contained in the configuration of this Turing Machine at a given time step? First there are basic bits of information like what the position of the head is, what state you're in, etc,. Next, there may be arbitrary elements of the ring R in the configuration. The first case is easy to deal with, while the latter requires more work. The goal is to do local checks over the iterations to see whether or not a particular configuration correctly follows from the previous configuration. We need to check two cases: 1) the current state is a computation state and 2) the current state is a branch state. All other cases can be handled in the standard way. If the current state is a computation state, then all the checks are easily expressible by some condition of the form $poly = 0$; i.e. does a given polynomial evaluate to 0? The only difficult case is if we're in a branch state. In this case, we may have some condition of the form: $p(a_1, a_2, a_3) =? 0?$. Where if the condition is true, then we go to state i and if the condition is false, then we should go to state j. So, we want to be able express this kind of computation. We can express it in terms of two polynomials $q(x)$ and $r(x)$ where:

$$q(x) = 0 \Longrightarrow p(a_1, a_2, a_3) = 0$$

$$r(x) = 0 \Longrightarrow p(a_1, a_2, a_3) \neq 0$$

And, using the usual manipulations, we can come up with a polynomial $P(a_1, a_2, a_3, x)$ such that if $P(a_1, a_2, a_3, x) \neq 0$ then $r(x) = 0$ and if $P(a_1, a_2, a_3, x) = 0$ then $q(x) = 0$. So, we have now defined the following scenario:

1. We have polynomials $p_1, \dots, p_l \in R[x_1, \dots, x_n]$
2. And we have polynomials $q_1, \dots, q_k \in R[x_1, \dots, x_n]$

And we need to answer the following question:

Does there exist $a_1, \dots, a_n \in R$ such that $p_i(a_1, \dots, a_n) = 0$ and $q_j(a_1, \dots, a_n) \neq 0$

Now, this is close to the Nullstellensatz – and to complete the reduction, we just have to show how to convert all the inequalities to equalities. We can use Rabinowitsch's Trick! So, given polynomials $p_1, \dots, p_l$ and polynomials $q_1, \dots, q_k$, we can define polynomials $p_1, \dots, p_l$ and r where:

$$r(x_1, \dots, x_n, y) \equiv (1 - y \cdot \Pi_{i=1}^{k} q_j(x_1, \dots, x_n))$$

Now we can make all equality checks! And so we've transformed any input instance into one of the 0/1 Hilbert Nullstellensatz. Therefore, $0/1 - HN_{\mathcal{C}}$ is $0/1 - NP_{\mathcal{C}}$-complete. ■

One thing to note about this proof is that Rabinowitsch's trick requires that you work in a field (because you need inverses which may not exist if you work over a ring). Consequently, this result only holds over arbitrary fields but does not necessarily hold over arbitrary rings. Having examined one way to view the computation of these Algebraic Turing Machines, we now shift to a slightly different point of view: the computation tree.

[14] The extended Riemann Hypothesis conjectures that the first quadratic non-residue mod p of a number is always less than $2(\ln p)^2$. This implies something along the lines of: every prime number has a generator from a small initial set.

20.2.5 Other Ways to View Computation

We consider other ways to view our new model of Algebraic computation. The fundamental question we are interested in is: given fixed inputs $x_1, \ldots, x_m$ to an $R - TM$ M_R, how can you reason about this computation? In particular, we will consider the *computation trees* of $R - TMs$ and use them to better understand computation in this Algebraic setting. Observe that we can view most of the computation in terms of some sufficiently high degree polynomial. Suppose that the first k steps of the computation in an $R - TM$ don't involve branches. Then, you can find polynomials $P_1, \ldots, P_k$ (each of which represents one step of the computation) where:

$$y_1 = P_1(x_1, \ldots, x_n)$$

$$y_2 = P_2(x_1, \ldots, x_n, y_1)$$

$$y_3 = P_3(x_1, \ldots, x_n, y_2)$$

$$\vdots$$

$$y_k = P_k(x_1, \ldots, x_n, y_{k-1})$$

And with a little work you can find polynomials $P_1', \ldots, P_k'$ such that $y_1 = P_1'(x_1, \ldots, x_n), \ldots, y_k = P_k'(x_1, \ldots x_k)$ (though the P_i' may have higher degree than the corresponding P_i). So, in particular, all non branch steps can be compressed and represented in terms of a single polynomial. This polynomial will have degree at most 2^{2^k} where k is the number of steps in your non-branching block of code. The bound 2^{2^k} is derived from the fact that you start with degree 2 polynomials, and at each step your polynomials can at most double in degree.

Given this formulation, if we add on branching states, then we can view all of the computation in terms of a "compressed" tree. Where each node contains a polynomial that represents either one of the above mentioned compressed non-branching polynomials, or a branching polynomial. Each branching node will then have pointers to two nodes (depending on whether or not the branch is taken). This alternative way of looking at computation enables us to attack a well known problem from pure mathematics: the decidability of the Mandelbrodt Set.

20.2.6 Is the Mandelbrodt Set Decidable?

We now consider the decidability of the Mandelbrot set under Algebraic Turing Machines over the complex numbers. The Mandelbrodt Decidability problem is formally defined as follows:
Input: A single complex number c.
Question: Consider the function $P_c : z \mapsto z^2 + c$. Is the following sequence bounded:

$$P_c(0), P_c(P_c(0)), P_c(P_c(P_c(0))), \ldots ?$$

(i.e. is c such that the iteration remains bounded?)
Traditional mathematics has indicated to us that this problem is very hard to solve – it is difficult to get a precise form on the value c. Now we have the tools to reason about this in a natural fashion – this is a computational problem that can be addressed using Turing Machines over the Reals. We can phrase this problem in the following equivalent fashion. If we define $Z_i \equiv P_c^i(0)$ then we can ask if there is a value k such that $|Z_k| > 2$. This condition is equivalent to asking whether or not the sequence is bounded because if $|Z_k| > 2$ then it's not difficult to work out that the sequence will diverge, and conversely if the sequence is unbounded, then it trivially follows that at some point $|Z_k| > 2$.

Now, if the sequence in unbounded – then k such that $|Z_k| > 2$ can be found in finite time. However, if the condition is not true, then this becomes more difficult. In fact, it turns out that this problem is *undecidable*. We outline the proof below:

Theorem 20.9 *The language $\{c \mid$ the sequence $Z_1, Z_2, \ldots$ converges$\}$ is undecidable.*

Proof (Sketch) Acceptance conditions by a Real Turing Machine at time T can be given by a collection of polynomial inequalities:

$$\begin{gathered} f_1(x_1,\dots,x_n) \geq 0 \\ f_2(x_1,\dots,x_n) \geq 0 \\ f_3(x_1,\dots,x_n) \geq 0 \\ \vdots \\ f_k(x_1,\dots,x_n) \geq 0 \end{gathered}$$

And this defines a semi-algebraic set. Now:

$$\bigcup_{t\geq 0} \text{acceptance at time } t$$

is a countable union of algebraic sets. Moreover, it is known that the boundary of a semi-algebraic set is a variety in R^2 and therefore it has Haussdorf Dimension 1. Since countable unions of semi-algebraic sets preserve Haussdorf dimension the overall Haussdorf dimension is 1. Now, let's shift to what is known about Mandelbrodt sets. It turns out that the Mandelbrodt set is connected, and has Haussdorf dimension 2 – hence by the above argument, it cannot be decidable. This proof is remarkable because we translated known topological constructs into complexity theoretic constructs. ∎

6.966 Algebra and Computation November 23, 1998

Lecture 21

Lecturer: Madhu Sudan *Scribe: Venkatesan Guruswami*

Our goal in this lecture is to prove the following theorem:

Theorem 21.1 *For any two algebraically closed fields K, L of characteristic zero, we have*

$$\mathrm{NP}_K = \mathrm{P}_K \Longleftrightarrow \mathrm{NP}_L = \mathrm{P}_L.$$

Note that the above theorem is important as it lends a certain robustness to the model of algebraic computation we have been considering. Indeed, the model would not be very attractive at all if the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question had one answer for one field, but the opposite one for another (similar) field.

We will also observe, as an aside, in course of the proof that

Theorem 21.2 $0/1 - \mathrm{P}_{\mathbb{C}} \subseteq \mathrm{BPP}$.

Theorem 21.1 will be proved through the following two lemmas

Lemma 21.3 *If K, L are algebraically closed fields with $K \subset L$, then $\mathrm{NP}_K = \mathrm{P}_K$ implies $\mathrm{NP}_L = \mathrm{NP}_L$.*

Lemma 21.4 *If K is any algebraically closed field of characteristic zero, then $\mathrm{NP}_K = \mathrm{P}_K$ implies $\mathrm{NP}_{\bar{\mathrm{Q}}} = \mathrm{P}_{\bar{\mathrm{Q}}}$ where $\bar{\mathrm{Q}}$ is the algebraic closure of the field of rational numbers.*

First observe that given the above two lemmas, the proof of Theorem 21.1 is in fact straightforward.

Proof of Theorem 21.1: Let K, L be any two algebraically closed fields of characteristic zero. Suppose $\mathrm{NP}_K = \mathrm{P}_K$. Then, by Lemma 21.4, $\mathrm{NP}_{\bar{\mathrm{Q}}} = \mathrm{P}_{\bar{\mathrm{Q}}}$. Now, since L is of characteristic zero, $\mathbb{Q} \subseteq L$, and this means $\bar{\mathrm{Q}} \subseteq L$ as L is algebraically closed. Using Lemma 21.3 this implies $\mathrm{NP}_L = \mathrm{P}_L$. Similarly, we can prove that $\mathrm{NP}_L = \mathrm{P}_L$ implies $\mathrm{NP}_K = \mathrm{P}_K$. Hence $\mathrm{NP}_K = \mathrm{P}_K \Leftrightarrow \mathrm{NP}_L = \mathrm{P}_L$. $\square$

It therefore suffices to prove Lemmas 21.3 and 21.4.

Proof of Lemma 21.3

Assume $\mathrm{NP}_K = \mathrm{P}_K$. As seen in the previous lecture, HN/K is NP_K-complete, thus this condition is *equivalent* to $\mathrm{HN}/K \in \mathrm{P}_K$. In particular, this implies that there is a polynomial-time K-Turing Machine M that decides HN/K. Our goal is to prove that $\mathrm{HN}/L \in \mathrm{P}_L$, which will imply that $\mathrm{NP}_L = \mathrm{P}_L$. We do this by claiming that the Turing Machine M without any change by itself decides HN/L when fed inputs over the field L. (Note that since $K \subset L$, M is a legal L-Turing Machine as the *constants* it uses will all lie in L.)

Suppose not, so that there is an instance of HN/L on which on which M errs. Note that an instance of HN/L is given by polynomials $f_1, f_2, \ldots, f_m \in L[X_1, X_2, \ldots, X_N]$; we denote by N the number of coefficients in all the f_i's and denote by $\hat{C} \in L^N$ the *explicit* representation of the input in terms of the vector of coefficients. Thus there is an input $\hat{C} \in L^N$ on which the machine errs with respect to its decision on whether or not $\hat{C} \in \mathrm{HN}/L$. Assume, for definiteness, that $\hat{C}$ is an YES instance of HN/L but that the machine M rejects $\hat{C}$ (the other possibility can be handled similarly and will therefore not be discussed).

Consider the operation of M on input $\hat{C}$. By assumption it reaches a rejecting configuration σ, and looking at the computation path leading from the start configuration to σ, we obtain polynomials $r_1, r_2, \ldots, r_l \in K[X_1, \ldots, X_N]$ and polynomials $s_1, s_2, \ldots, s_k \in K[X_1, X_2, \ldots, X_N]$ such that any input $\hat{C}$ to M *leads* to the reject configuration σ if

$$r_1(\hat{C}) = r_2(\hat{C}) = \cdots = r_l(\hat{C}) = 0 \text{ and } s_1(\hat{C}) \neq 0, s_2(\hat{C}) \neq 0, \cdots, s_k(\hat{C}) \neq 0. \tag{21.1}$$

We assumed that $\hat{C}$ was a YES instance of HN/L, and we now proceed to characterize YES instances of HN/L. Let $f_1, f_2, \ldots, f_m \in L[X_1, X_2, \ldots, X_n]$ be such that $f_1, f_2, \ldots, f_m$ have a common zero; which, by the

Weak Nullstellensatz, is equivalent to $\sum_i f_i q_i \neq 1 \; \forall q_1, q_2, \ldots, q_n \in L[X_1, \ldots, X_N]$. We can express this as a linear system (with unknowns z being the coefficients of the q_i's and the entries of A being polynomials in the coefficients of $f_1, f_2, \ldots, f_m$, and therefore polynomials in $\hat{C}$), and our condition is equivalent to this linear system not having a solution. Expressing this as a condition in terms of the ranks of certain sub-matrices of A (as was done in Lecture 16 on Quantifier Elimination) we get polynomials $\phi_1^j, \ldots, \phi_m^j, \psi_1^j, \ldots, \psi_m^j \in \mathbb{Z}[X_1, \ldots, X_N]$ (for a certain number of j's) such that $\hat{C} \in \mathrm{HN}/L$ if and only if

$$\exists j \text{ s.t } \phi_1^j(\hat{C}) = \cdots \phi_m^j(\hat{C}) = 0 \text{ and } \psi_1^j(\hat{C}) \neq 0, \ldots, \psi_m^j(\hat{C}) \neq 0. \tag{21.2}$$

Combining Conditions 21.1 and 21.2, and renaming the polynomials for convenience, we obtain that there exist polynomials $g_1, g_2, \ldots, g_a, h \in K[X_1, X_2, \ldots, X_N]$ such that

$$g_1(\hat{C}) = g_2(\hat{C}) = \cdots = g_a(\hat{C}) = 0 \text{ and } h(\hat{C}) \neq 0. \tag{21.3}$$

(You might wonder whether there should be multiple polynomials which should all be non-zero at $\hat{C}$ in the above condition, but clearly this condition is equivalent to the single condition that their product is non-zero at $\hat{C}$.)

We are now going to claim the existence of a $\hat{C}' \in K^N$ which also satisfies the above Condition 21.3. This claim clearly follows from Lemma 21.5, and implies that $\hat{C}'$ is an Yes instance of HN/K but that M ends up in a reject configuration (namely σ) on input $\hat{C}'$, contradicting the hypothesis that M correctly decides HN/K. This contradiction proves that M in fact decides HN/L as well.

Lemma 21.5 *Let $S \subset K[X_1, X_2, \ldots, X_N]$, $h \in K[X_1, X_2, \ldots, X_N]$ and $V_K = \{\hat{x} \in K^N : f(\hat{x}) = 0 \; \forall f \in S\}$ and $V_L = \{\hat{x} \in L^N : f(\hat{x}) = 0 \; \forall f \in S\}$. Suppose that $\exists \hat{y} \in V_L$ such that $h(\hat{y}) \neq 0$. Then $\exists \hat{z} \in V_K$ such that $h(\hat{z}) \neq 0$.*

Proof: Since $h(\hat{y}) \neq 0$, we have using the strong form of the Nullstellensatz that $h \notin \mathrm{Rad}(\mathrm{Ideal}_L(S))$ (the subscript L indicates that this ideal is considered in the ring $L[X_1, \ldots, X_N]$). Hence $h \notin \mathrm{Rad}(\mathrm{Ideal}_K(S))$ as well, which implies, once again by the Nullstellensatz, that $V_K \cap \{\hat{x} : h(\hat{x}) \neq 0\} \neq \emptyset$, as desired. □

The proof of Lemma 21.3 is thus complete; in fact we have also proved the following:

Corollary 21.6 *If K, L are algebraically closed fields with $K \subset L$, then* $\mathrm{HN}/K = K^* \cap \mathrm{HN}/L$.

Proof of Lemma 21.4

By virtue of Corollary 21.6, we know that if a K-Turing Machine M decides HN/K then it also decides $\mathrm{HN}/\bar{\mathrm{Q}}$ correctly. The problem, however, is that the machine M uses constants from K while it is only allowed constants from $\bar{\mathrm{Q}}$.

Recall (from the previous lecture) that M is defined by constants $\alpha_1, \alpha_2, \ldots, \alpha_k \in K$ and polynomials $g_1, g_2, \ldots, g_l \in \mathbb{Z}[Y_1, Y_2, \ldots, Y_k, X_1, X_2, \ldots, X_N]$ (N as before stands for the size of the input $\hat{C} \in \bar{\mathrm{Q}}^N$) that act on $\alpha_1, \alpha_2, \ldots, \alpha_k$ and the cell contents. Note we have assumed that the g_i's are polynomials over the integers, since we can alway achieve the functionality of a polynomial over K by a polynomial over $\mathbb{Z}$ together with some (small number of) additional constants from K. Further we may assume that each of these polynomials g_i is used by M only in a branch condition by collapsing computations between successive branch computations into a single one, so that reaching a particular accept/reject state at the end of M's computation is determined by some subset of the g_i's evaluating to 0 and some subset of the g_i's being non-zero at $\alpha_1, \alpha_2, \ldots, \alpha_k, \hat{C}$ (as in Equation 21.1).

We now want to simulate the behavior of the machine M by a $\bar{\mathrm{Q}}$-Turing Machine M' which uses constants $\beta_1, \beta_2, \ldots, \beta_k \in \bar{\mathrm{Q}}$ where the β_j's are a function only of $M, \alpha_1, \alpha_2, \ldots, \alpha_k$ and the input $\hat{C}$.

The following lemma says that constants which are algebraic are easy to handle.

Lemma 21.7 *If M uses constants from $K(\gamma)$ where $f(\gamma) = 0$ for some $f \in K[X]$ and decides $L \subseteq K^*$, then there exists a K-Turing Machine M' that decides L with the same asymptotic run-time.*

Proof: Constants in $K(\gamma)$ can be represented as $a_0 + a_1\gamma + \cdots + a_l\gamma_l$ where $l = \deg(f) - 1$. Now it is easy to see that operations on constants in $K(\gamma)$ can be simulated by operations on K with a poly(l) factor blow-up in run time. □

In light of the above Lemma, we can assume that every polynomial $P \in \bar{\mathbb{Q}}[Y_1, Y_2, \ldots, Y_k] \setminus \{0\}$ satisfies $P(\alpha_1, \alpha_2, \ldots, \alpha_k) \neq 0$. For $1 \leq i \leq l$, let $g_i^{(\hat{C})} \in \bar{\mathbb{Q}}[Y_1, Y_2, \ldots, Y_k]$ be the polynomial obtained from g_i by fixing the variables $X_1, X_2, \ldots, X_N$ to equal the input $\hat{C} \in \bar{\mathbb{Q}}^N$. Hence by our argument, if $g_i^{(\hat{C})} \not\equiv 0$, then $g_i^{(\hat{C})}(\alpha_1, \alpha_2, \ldots, \alpha_k) \neq 0$. Thus, the effect of using $\alpha_1, \alpha_2, \ldots, \alpha_k \in K$ can be achieved by using any constants $\beta_1, \beta_2, \ldots, \beta_k \in \bar{\mathbb{Q}}$ which satisfy $g_i^{(\hat{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for *every* i such that $g_i^{(\hat{C})} \not\equiv 0$. The goal of the remaining part of the proof, therefore, will be to indicate how such a choice of the β_i's can be made, and this will clearly complete the proof of Lemma 21.4.

The key observation now is that the degrees of the g_i's (and hence those of the $g_i^{(\hat{C})}$'s) are not very large. Indeed, if M runs for T units of time, then all polynomials it computes and uses are of degree at most 2^T. Now consider picking $\beta_1, \beta_2, \ldots, \beta_k \in_{\mathrm{R}} \{1, 2, \ldots, l \cdot 2^{T^2}\}$ – clearly with high probability this random choice will satisfy $g_i^{(\hat{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for all i such that $g_i^{(\hat{C})} \not\equiv 0$. Since T is polynomial in N, picking the β_i's only takes poly(N) random bits and poly(N) time.

The above argument already gives a BPP algorithm simulating M since the β_i's used above are only *small* integers. In fact, our proof so far yields the containment $0/1 - \mathrm{P}_{\mathbb{C}} \subseteq \mathrm{BPP}$ which was claimed in Theorem 21.2.

It still remains to *derandomize* this simulation of M. The basic idea in doing this is to exploit the power we have of operating on very large integers in a single unit of time (since M' is a $\bar{\mathbb{Q}}$-Turing Machine) by choosing β_i's which are very large.

We first note that not only are the degrees of the g_i's all "small", but also the coefficients of each g_i are at most 2^{2^T}, which implies that the coefficients of $g_i^{(\hat{C})}$'s are at most $(\max\{2, \hat{C}_1, \ldots, \hat{C}_N\})^{4^T}$. We can now finish the argument using the following Lemma, whose proof we defer to the end.

Lemma 21.8 *If* $h \in \mathbb{Z}[Y_1, Y_2, \ldots, Y_k]$ *is a* non-zero *polynomial over the integers with total degree at most* D *and coefficients of absolute value at most* A*, then for* $\beta_1 = D \cdot A^D$, $\beta_2 = D \cdot A^D \cdot \beta_1^{D^2}$, $\ldots$, $\beta_k = D \cdot A^D \cdot \beta_{k-1}^{D^2}$, *we have* $h(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$.

Using the above lemma with $D = 2^T$ and $A = \max\{2, \hat{C}_1, \ldots, \hat{C}_N\}^{4^T}$, we find $\beta_1, \beta_2, \ldots, \beta_k$ such that $g_i^{(\hat{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for every i such that $g_i^{(\hat{C})} \not\equiv 0$, as was desired.

It has to be still checked that these β_i's can be computed quickly. But this is clear: indeed β_1 can computed in $O(T)$ time, and each β_i can once again be computed in $O(T)$ time from β_{i-1}. Thus the overall time to compute $\beta_1, \beta_2, \ldots, \beta_k$ is $O(kT)$ which is polynomial in N, the size of the original input $\hat{C} \in \bar{\mathbb{Q}}^N$, as both T and k are definitely polynomial in N.

Thus the K-Turing Machine M can be simulated with just a polynomial slow down by the $\bar{\mathbb{Q}}$-Turing Machine M', and the proof of Lemma 21.4 is complete. □

Proof of Lemma 21.8: We claim, by induction, that the polynomial $h_i(Y_{i+1}, \ldots, Y_k)$ defined as $h(\beta_1, \beta_2, \ldots, \beta_i, Y_{i+1}, \ldots, Y_k)$ is non-zero and has coefficients at most $A \cdot \beta_i^D$. Clearly this will imply the Lemma, since the constant polynomial $h_k = h(\beta_1, \beta_2, \ldots, \beta_k)$ will be non-zero.

The base case of the induction ($i = 0$) is trivial. For $i \geq 1$, the bound on the coefficient size follows by simple inspection. The fact that h_i is non-zero given h_{i-1} is non-zero reduces to a univariate problem as follows: Express h_{i-1} as a polynomial in $Y_{i+1}, \ldots, Y_k$ with coefficients that are polynomials in Y_i. Since h_{i-1} is non-zero, one of these coefficients is a non-zero polynomial in Y_i, call this polynomial $p(Y_i)$. By induction hypothesis, the coefficients of p are at most $A \cdot \beta_{i-1}^D$. Thus our lemma reduces to proving the claim for the univariate case, namely that $\beta_i = D \cdot A^D \cdot \beta_{i-1}^{D^2}$ is *not* a root of $p(Y_i)$, and this follows from the bound on the size of roots of univariate polynomials over integers proved in Claim 1 of Lecture 10. □

6.966 Algebra and Computation November 25, 1998

Lecture 22

Lecturer: Madhu Sudan *Scribe: Yevgeniy Dodis*

In this lecture we will show that the Hilbert Nullstellensatz decision problem with integer coefficients (but complex witnesses!, denoted $\mathrm{HN}_{\mathbb{Z}}$) is in the polynomial time hierarchy (**PH**) under the Extended Riemann Hypothesis (ERH). This statement immediately implies that $0/1\text{–}\mathbf{NP}_{\mathbb{C}} \subseteq \mathbf{PH}$ under the ERH, strengthening the unconditional guarantee that $0/1\text{–}\mathbf{NP}_{\mathbb{C}} \subseteq \mathbf{PSPACE}$. In fact, we will show even more, namely that $\mathrm{HN}_{\mathbb{Z}} \in \mathrm{AM} \subseteq \mathbf{PH}$ under the ERH, where AM is the class of languages having a constant round Arthur-Merlin game.

22.1 Problem Statement and Overview of the Proof

Here is the general instance of $\mathrm{HN}_{\mathbb{Z}}$:

INPUT: $\mathbb{F} = \{f_1, \ldots, f_m\}$, $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$ of total degree at most d.

QUESTION: $\exists\, a_1, \ldots, a_n \in \mathbb{C}$ such that $f_i(a_1, \ldots, a_n) = 0$, for all $i \in [m]$?

So we are asking if our integer polynomials have at least one common zero over the *complexes*. We note that $\mathrm{HN}_{\mathbb{Z}}$ is indeed complete for $0/1\text{–}\mathbf{NP}_{\mathbb{C}}$. This follows from two facts that we indicated last lecture. Firstly, any language in $\mathbf{P}_{\mathbb{C}}$ can be decided by a polynomial-time $\mathbb{C}$-machine that uses only 0 and 1 as its constants. Using this and tracing carefully the reduction of any language $L \in \mathbf{NP}_{\mathbb{C}}$ to $\mathrm{HN}_{\mathbb{C}}$, we see that when $x \in \{0,1\}^*$, the resulting consistency checking polynomials indeed have all coefficients equal to 0 and 1, i.e. we indeed reduce $0/1\text{–}L$ to $0/1\text{–}\mathrm{HN}_{\mathbb{C}} \subseteq \mathrm{HN}_{\mathbb{Z}}$.

The input size is the number of bits to represent all the coefficients of $f_1, \ldots, f_m$, where each polynomial is given as a list of roughly $\binom{n+d}{d}$ coefficients, each of magnitude at most C. With a polynomial blowup in the input size, we could have assumed that $d = 2$ and all coefficients are 0 or 1. Thus, the issue of redundant/non-redundant representation of a given polynomial is not important. Let us denote by $|\mathbb{F}|$ the size of our input which can be assumed w.l.o.g. to be $|\mathbb{F}| = \mathrm{poly}(n, m, d, \log C)$.

Main Idea: Look at f_i's modulo a prime p, where $p \in [N]$ and N will be specified later. We hope that our system is unsatisfiable over $\mathbb{C}^n$ if and only if its version modulo p is unsatisfiable over $\mathbb{Z}_p^n$ for "most" primes $p \in [N]$.

We note that proving that $\mathbb{F}$ has a solution over $\mathbb{Z}_p^n$ takes only $(n \log p)$ bits (unlike the proof over $\mathbb{C}^n$ that may take unbounded number of bits). We also note that the above equivalence will have "two-sided error" in that unsatisfiable $\mathbb{F}$ over $\mathbb{C}^n$ might become satisfiable over $\mathbb{Z}_p^n$, and satisfiable $\mathbb{F}$ over $\mathbb{C}^n$ might become unsatisfiable over $\mathbb{Z}_p^n$. However, N will be chosen large enough so that there is a significant gap between the number of bad primes for a yes-instance and the number of bad primes for a no-instance. This will us allow to solve this "gap" problem (the gap is in the number of primes $p \in [N]$ for which $\mathbb{F}$ is satisfiable over $\mathbb{Z}_p^n$) in AM. More specifically, we show

Theorem 22.1 *There exists a constant N_1 such that if $\mathbb{F}$ is unsatisfiable over $\mathbb{C}^n$, there are at most N_1 primes $p \in \mathbb{Z}$ such that $\mathbb{F}$ is satisfiable over $\mathbb{Z}_p^n$. Moreover, $N_1 = \exp(|\mathbb{F}|)$* [15].

Theorem 22.2 *Under the ERH, there are constants N_2 and N_3 such that if $\mathbb{F}$ is satisfiable over $\mathbb{C}^n$, there are at least $(\frac{\pi(N)}{N_2} - N_3 - O(\sqrt{N}\log N))$ primes $p \le N$ such that $\mathbb{F}$ is also satisfiable over $\mathbb{Z}_p^n$, where $\pi(N)$ is the number of primes less than or equal to N (asymptotically, $\pi(N) \approx N/\log N \gg \sqrt{N}\log N$). Moreover, $N_2, N_3 = \exp(|\mathbb{F}|)$.*

Let N be such that $\frac{\pi(N)}{N_2} - N_3 - O(\sqrt{N}\log N) \gg N_1$, so in particular $N = \exp(|\mathbb{F}|)$ suffices. Then, Theorems 22.1 and 22.2 imply that it is sufficient to solve in AM the following promise problem that we call Gap-HN. Let $N = \exp(|\mathbb{F}|)$ be as specified, $N_1 = P_1 \ll P_2 = \frac{\pi(N)}{N_2} - N_3 - O(\sqrt{N}\log N)$.

[15] Notation $a = \exp(b)$ denotes the fact that $a = 2^{\mathrm{poly}(b)}$ for an appropriately chosen polynomial of b.

INPUT: $\mathbb{F} = \{f_1, \ldots, f_m\}$, $f_i \in \mathbb{Z}[x_1, \ldots, x_n]$ of total degree at most d.

Define $Primes(\mathbb{F}) = \{p \in [N] \mid p \text{ - prime}, \mathbb{F} \text{ is satisfiable over } \mathbb{Z}_p^n\}$.

NO INSTANCES: $|Primes(\mathbb{F})| \leq P_1$.

YES INSTANCES: $|Primes(\mathbb{F})| \geq P_2$.

Theorem 22.3 Gap-HN $\in AM$

In the remaining sections, we prove Theorems 22.1, 22.2, 22.3. For Theorems 22.1 and 22.2 we will separate conceptual ideas with the messy details by leaving the (sketch of the) details in a separate section.

22.2 Overview of AM and Proof of Theorem 22.3

First, let us recall what the class AM is. It is a class of (promise, but we ignore it for now) languages L for which there is a constant round Arthur-Merlin game. An Arthur-Merlin game is simply a public coin interactive protocol between an all-powerful prover Merlin and a probabilistic polynomial time verifier Arthur. Except for the verification stage, at each round Arthur only flips new truly random coins and gives them to Merlin. Merlin has to provide some reply (which can be thought as commitment) for each sequence of random coins (which are kind of challenges) from Arthur. At the verification stage, Arthur simply performs some polynomial time computation on his random coins and Merlin's responses. For an yes-instance x, honest Merlin can make Arthur accept with probability at least $\frac{2}{3}$. For a no-instance x, Arthur should reject with probability at least $\frac{2}{3}$ no matter how hard Merlin tries to cheat.

A crucial characteristic of an Arthur-Merlin game is the number of rounds. With polynomially many rounds, we can recognize any language in **IP** = **PSPACE**. With any constant number of rounds, it is possible to show that we can collapse them to only 1 round: a random challenge from Arthur, followed by a reply from Merlin. Using this and the standard boosting probability of error trick, we can place AM in the hierarchy: AM $\subseteq \Pi_2^p \subseteq$ **PH**.

One of the most useful properties of AM is its ability to recognize the following languages. Assume that we have some universe U and every input x defines "good" set $Good(x) \subseteq U$. Assume further that for each x, the membership in $Good(x)$ is in **NP**, i.e. given $y \in U$, there is a polynomial-size witness proving that $y \in Good(x)$. Define a promise language L where x is a yes-instance if $|Good(x)| \geq P_2$, and a no-instance if $|Good(x)| \leq P_1$, where $P_2 > 4P_1$.

Lemma 22.4 $L \in AM$

Proof: A first attempt that fails is for Arthur to pick a random element $y \in U$ and ask Merlin to prove that $y \in Good(x)$. However, P_2 is often exponentially smaller than $|U|$. So even a yes-instance x will convince Arthur with a negligible probability $P_2/|U|$. A solution is to use a *universal family of hash functions*. Assume $|S| = P_2$ and $\mathcal{H}$ is a family of universal hash functions from U to S. Take any $T \subseteq U$ with $|T| = \alpha|S|$, $\alpha \leq 1$. Then universality of $\mathcal{H}$ easily implies that if we choose a random $h \in \mathcal{H}$ and a random point $s \in S$ we have

$$\alpha - \frac{\alpha^2}{2} \leq \Pr[s \in h(T)] \leq \alpha \tag{22.1}$$

Let us now apply this to $T = Good(x)$. For a no-instance x, $\alpha = \frac{|Good(x)|}{|S|} \leq \frac{P_1}{P_2} \leq \frac{1}{4}$ and we get $\Pr[s \in h(Good(x))] \leq \frac{1}{4}$. For a yes-instance x, we can make $\alpha = 1$ by considering any P_2 elements of $Good(x)$, and get $\Pr[s \in h(Good(x))] \geq 1 - \frac{1}{2} = \frac{1}{2}$. This suggests the following AM protocol. Arthur chooses a random $h \in \mathcal{H}$ and a random $s \in S$. Merlin has to reply with any preimage $y \in Good(x)$ such that $h(y) = s$ and a proof that indeed $y \in Good(x)$ (which is possible as the language is in **NP**). Above analysis shows that yes-instances are accepted with probability $\frac{1}{2}$ and no-instances are accepted with probability at most $\frac{1}{4}$, so $L \in$ AM. ■

This Lemma proves Theorem 22.3. Take U to be all the primes below N, $Good(\mathbb{F}) = Primes(\mathbb{F})$ and the membership in $Primes(\mathbb{F})$ can be proven in **NP** by giving a common zero $\in \mathbb{Z}_p^n$ of $f_1, \ldots, f_m$. Here we used the fact that $N = \exp(|\mathbb{F}|)$, so that for all $p < N$, $\log p = \text{poly}(|F|)$, i.e. the common zero in $\mathbb{Z}_p^n$ takes only polynomial number of bits to specify.

22.3 Proof of Theorem 22.1

Assume $\mathbb{F}$ is not satisfiable over $\mathbb{C}^n$. By Hilbert Nullstellensatz, $\exists q_1, \ldots, q_m \in \mathbb{C}[x_1, \ldots, x_n]$ such that $\sum q_i f_i = 1$. How low can we place coefficients of q_i's? Clearly we can place them in $\bar{\mathbb{Q}}$ (the algebraic closure of $\mathbb{Q}$) since $\bar{\mathbb{Q}}$ is an algebraically closed field (so HN holds) and $f_i \in \mathbb{Z}[x_1, \ldots, x_n] \subseteq \bar{\mathbb{Q}}[x_1, \ldots, x_n]$. Can we place them in $\mathbb{Q}$? Yes. Effective HN gives us the bounds on degrees of $q_i \in \bar{\mathbb{Q}}[x_1, \ldots, x_n]$, so we can set up a linear system to obtain the coefficients of q_i's. Since all the coefficients of the system will be in $\mathbb{Q}$ and we know that the solution in $\bar{\mathbb{Q}}$ exists, the solution in $\mathbb{Q}$ exists as well (since $\mathbb{Q}$ is a field).

Let a be the least common denominator of coefficients of these q_i's. We will demonstrate later that we can ensure the following bound on a.

Claim 22.5 *There are valid q_i's such that a is at most doubly exponential in $|\mathbb{F}|$.*

Now, by multiplying all the q_i's by a we obtain:

Lemma 22.6 *If $\mathbb{F}$ is not satisfiable over $\mathbb{C}^n$, then*

$$\exists r_1, \ldots, r_m \in \mathbb{Z}[x_1, \ldots, x_n],\ a \in \mathbb{Z},\ 0 \neq a = \exp(\exp(|\mathbb{F}|)) \text{ such that } \sum r_i f_i = a$$

Therefore, if $f_1, \ldots, f_m$ have a common zero over $\mathbb{Z}_p^n$, it must be the case that p divides a. The number of such p's is at most $N_1 := \log a = \exp(|\mathbb{F}|)$. ■

22.3.1 Bound on a (Proof of Claim 22.5)

Let us look closer at the linear system for the coefficients of q_i's. Effective HN tells us that degrees of q_i are at most $\exp(|F|)$, so the number of unknowns in the system is still at most $\exp(|\mathbb{F}|)$. Notice also that each coefficient of this system is a coefficient of some f_i and thus takes $|\mathbb{F}|$ bits to represent. Hence, we have an exponential size system over $\mathbb{Q}$ where each coefficient is $|\mathbb{F}|$ bits long. Moreover, we know the system has some solution. Let us throw away all redundant equations and take some principal minor whose determinant is not zero. Set all the variables outside this principal minor to 0. The remaining variables are then set in a unique way. Moreover, all of them have a denominator divisible by the determinant of this principal minor. Let the size of the minor be $D = \exp(|\mathbb{F}|)$. Then its denominator is at most $\log(D! \cdot 2^{|\mathbb{F}|D}) = \exp(|\mathbb{F}|)$ bits long. So all the variables (including the ones that we set to 0) in this particular solution have a common denominator a that is at most $\exp(|\mathbb{F}|)$ bits long, proving the claim. ■

22.4 Proof of Theorem 22.2

This is the most technical part where we will end up using ERH. Assume that $f_1, \ldots, f_m$ have a common zero $(a_1, \ldots, a_n) \in \mathbb{C}^n$. First of all, since all the coefficients of f are in $\mathbb{Z} \subseteq \bar{\mathbb{Q}}$, we can assume that $a_i \in \bar{\mathbb{Q}}$, i.e. there is a common zero in $\bar{\mathbb{Q}}^n$. This follows from Hilbert Nullstellensatz. Indeed, a zero in $\mathbb{C}^n$ implies that the linear system $\sum q_i f_i = 1$ is unsatisfiable over $\mathbb{C}$. Since all the coefficients are also in $\mathbb{Z} \subseteq \bar{\mathbb{Q}}$, this system is unsatisfiable over $\bar{\mathbb{Q}}$ as well. And since $\bar{\mathbb{Q}}$ is algebraically closed, the converse of HN tells us that there is a common zero in $\bar{\mathbb{Q}}^n$. Let us develop some intuition by looking at some easy sub-cases.

Assume all $a_i \in \mathbb{Z}$. In this case $(a_i \bmod p)$ form a solution in $\mathbb{Z}_p^n$ for any prime p, so the number of such p's is exactly $\pi(N)$ which is great. The next sub-case is when all $a_i \in \mathbb{Q}$, i.e. $a_i = \frac{a_i'}{b}$, where $a_i', b \in \mathbb{Z}$. Define $g_i(y_1, \ldots, y_n) = b^d f_i(\frac{y_1}{b}, \ldots, \frac{y_n}{b})$, so that $g_i \in \mathbb{Z}[x_1, \ldots, x_n]$. Then $f_i(\frac{a_1'}{b}, \ldots, \frac{a_n'}{b}) = 0 \Longrightarrow g_i(a_1', \ldots, a_n') = 0$. Hence, for any prime p, $(a_i' \bmod p)$ is a common zero of $g_1, \ldots, g_m$ over $\mathbb{Z}_p^n$. Then, provided p does not divide b, $(a_i' b^{-1} \bmod p)$ form a common zero of $f_1, \ldots, f_m$ over $\mathbb{Z}_p^n$. This follows from $f_i(x_1, \ldots, x_n) = b^{-d} g_i(bx_1, \ldots, bx_n)$. Hence, the number of good primes is at least $\pi(N) - \log b$. We will not do it, but it is possible to show that $b = \exp(\exp(|\mathbb{F}|))$, so we would get $\pi(N) - \exp(|\mathbb{F}|)$ good primes.

With these ideas, let us move now to the general case when all $a_i \in \bar{\mathbb{Q}}$. First, we want a simple compact representation of a_i in terms of $\mathbb{Z}$. Consider $\mathbb{Q}(a_1, \ldots, a_n)$, the field extension of $\mathbb{Q}$ obtained by adjoining $a_1, \ldots, a_n$ to it. By the famous primitive element theorem, we know that there exists $\beta \in \bar{\mathbb{Q}}$ such that in fact $\mathbb{Q}(a_1, \ldots, a_n) = \mathbb{Q}(\beta)$, i.e. a single algebraic extension suffices. Let $R(x) \in \mathbb{Q}[x]$ be the minimal polynomial for β (i.e. $R(\beta) = 0$ and $R'(\beta) = 0 \Longrightarrow \deg R' \geq \deg R$). By multiplying by the least common denominator, we can assume that $R \in \mathbb{Z}[x]$. Also, since $a_1, \ldots, a_n \in \mathbb{Q}(\beta)$, there are polynomials $p_1, \ldots, p_n \in \mathbb{Z}[x]$ and a least common denominator b such that $a_i = p_i(\beta)/b$. This is the concise representation that we wanted. We will sketch later the proof of the following claim that we assume for now:

Claim 22.7 *By an appropriate choice of the common zero $(a_1, \ldots, a_n)$, b and the coefficients of R are at most doubly exponential in $|\mathbb{F}|$, while the degree D of R is at most singly exponential in $|\mathbb{F}|$.*

Define $g_i(x) = b^d f_i(\frac{p_1(x)}{b}, \ldots, \frac{p_n(x)}{b})$, so that for $i \in [m]$

1. $g_i(x) \in \mathbb{Z}[x]$.
2. $g_i(\beta) = b^d f_i(\frac{p_1(\beta)}{b}, \ldots, \frac{p_n(\beta)}{b}) = b^d f_i(a_1, \ldots, a_n) = 0$.
3. $R(x)$ divides $g_i(x)$ (since R is a minimal polynomial of β and $g_i(\beta) = 0$).

Suppose now that a prime p does not divide b and that $R(x)$ has a root γ in $\mathbb{Z}_p$, i.e. $R(\gamma) \bmod p = 0$. Then $g_i(\gamma) \bmod p = 0$ as well since R divides g_i. Then, $\mathbb{F}$ is satisfiable over $\mathbb{Z}_p^n$ by $(p_1(\gamma)b^{-1} \bmod p, \ldots, p_n(\gamma)b^{-1} \bmod p)$ since $f_i(\frac{p_1(\gamma)}{b}, \ldots, \frac{p_n(\gamma)}{b}) = b^{-d} g_i(\gamma) = 0 \bmod p$. Define

$$S_{N,R} = \{p \in [N] \mid p \text{ - prime}, p \text{ does not divide } b, \exists \gamma \in \mathbb{Z}_p \text{ s.t. } R(\gamma) = 0 \bmod p\}$$

From the above discussion, the number of primes satisfying Theorem 22.2 is at least $|S_{N,R}|$. Since b is doubly exponential, there are at most $\log b = \exp(|\mathbb{F}|)$ primes dividing b. We would be done except we need R to have a root in $\mathbb{Z}_p$. For that, we will need to use ERH by using (as a black-box) the following powerful theorem telling us that we will have roots for a lot of p's.

Theorem 22.8 (Effective Chebotarev Density Theorem) *Suppose $R \in \mathbb{Z}[x]$ is irreducible (square free suffices as well) of degree D. Let $\Delta = \mathrm{disc}(R)$ be the discriminant of R (recall, the resultant of R and the derivative of R, which is not zero since R is square free). Define*

$$\Pi'_{N,R} = \{p \in [N] \mid p \text{ - } \textit{prime}, p \textit{ does not divide } \Delta\}$$

$$W_{N,R} = \{(\gamma, p) \mid p \in \Pi'_{N,R}, R(\gamma) = 0 \bmod p\}$$

Then, assuming ERH,

$$|W_{N,R}| = |\Pi'_{N,R}| - O(\sqrt{N}\log(N\Delta)) \tag{22.2}$$

We are almost done. First, using the fact that $D = \exp(|\mathbb{F}|)$ and the coefficients of R are $2^{\exp(|\mathbb{F}|)}$, we bound the logarithm of the discriminant of R: $\log \Delta \leq \log((2D)! \cdot 2^{\exp(|\mathbb{F}|)2D}) = \exp(|F|)$. Also, $|\Pi'_{N,R}| \geq \pi(N) - \log \Delta$ since at most $\log \Delta$ primes divide Δ. Using this, Chebotarev's Density Theorem and the fact that R can have at most D roots in $\mathbb{Z}_p$ (so that each p is counted at most D times in $|W_{N,R}|$), we get:

$$\begin{aligned} |S_{N,R}| &\geq \frac{|W_{N,R}|}{D} - \log b \overset{(22.2)}{\geq} \frac{|\Pi'_{N,R}| - O(\sqrt{N}\log(N\Delta))}{D} - \log b \\ &\geq \frac{\pi(N) - \log\Delta - O(\sqrt{N}\log(N\Delta))}{D} - \log b = \frac{\pi(N)}{N_2} - N_3 - O(\sqrt{N}\log N) \end{aligned}$$

Here $N_2 := D = \exp(|\mathbb{F}|)$, $N_3 := \frac{\log\Delta}{D} + \log b = \exp(|\mathbb{F}|)$, $\log(N\Delta) = O(\log N)$ provided $N = \exp(|\mathbb{F}|)$. This completes the proof. ∎

22.4.1 Bounds on b, D and the coefficients of R (Proof of Claim 22.7)

This is the most annoying part that we sketch since it does not shed too much light on what is going on. Here for the first time we have to be slightly careful when dealing with the input size $|\mathbb{F}| = \mathrm{poly}(n, m, d, \log C)$. Let $R = \max(n, m, d)$, so that $|\mathbb{F}| = \mathrm{poly}(R, \log C)$ i.e. we separate out the sizes of the coefficients of f_i's. Let us start with bounds on minimal polynomials over $\mathbb{Q}$ defining $a_i \in \bar{\mathbb{Q}}$, where $(a_1, \ldots, a_n)$ is our common zero. Recall that $\mathrm{HN}_{\mathbb{Z}}$ is the following problem: determine if $\exists x_1, \ldots, x_n$ such that $f_1(x_1, \ldots, x_n) = 0, \ldots, f_1(x_1, \ldots, x_n) = 0$. Let us use quantifier elimination and eliminate some x_k. Then the predicate that such $x_1, \ldots, x_n$ exist becomes a formula of x_k that can be written in the form:

$$\Gamma^k(x_k) := \bigvee_i \bigwedge_j (\Phi^k_{ij}(x_k) = 0 \wedge \Psi^k_{ij}(x_k) \neq 0)$$

where (we mentioned these bounds when we studied quantifier elimination).

- Φ^k_{ij} and Ψ^k_{ij} are polynomials over $\mathbb{Z}$.
- Their degree is at most $\exp(R)$.
- Their coefficients are at most $\exp(R) \log C$ bits long.
- The range of i and j is at most $\exp(R)$.

Since we know that $\Gamma^k(a_k)$ is true, there is a particular i_0 such that $\bigwedge_j (\Phi^k_{i_0 j}(a_k) = 0 \wedge \Psi^k_{i_0 j}(a_k) \neq 0)$ is true. We split the analysis into 2 cases.

Assume that for some $1 \leq k \leq n$ all $\Phi^k_{i_0 j}(x) \equiv 0$, say $k = n$ w.l.o.g. Then $\Gamma^n(x_n)$ has a clause of the form $\bigwedge_j (\Psi^n_{i_0 j}(x_n) \neq 0)$. Since each $\Psi^n_{i_0 j}$ has a singly exponential degree (everything is in R for now), it has only that many zeros, and as there are exponentially many indices j, we get that only exponentially many x_n make this clause false. Hence, the whole $\Gamma^n(x_n)$ is false only for at most *exponentially* many x_n. Thus, in this case we can redefine a_n (if necessary) so that it even becomes a singly exponential *integer*, $a_n \leq \exp(R)$. This a_n is trivially a root of a degree 1 integer polynomial $x - a_n = 0$. Of course, this implicitly redefines $a_1, \ldots, a_{n-1}$, but we know that our new a_n still leaves $\mathbb{F}|_{x_n = a_n}$ satisfiable. Now we can completely eliminate x_n by setting it equal to (integer) a_n everywhere in $\mathbb{F}$. We get a new (satisfiable) system with only $n-1$ variables, same maximum degree d and number of equation m (so that we can use the same R for this system). The only thing that grows is the magnitude of the largest coefficient that can be up to $(\exp(R) \cdot C)$. In this case we apply our analysis from scratch to an $(n-1)$-variable system $\mathbb{F}|_{x_n = a_n}$ with $R' = R$, $C' = \exp(R) \cdot C$.

The complementary case to the above is when for *all* $1 \leq k \leq n$ there exists some $j_0(k)$ such that $t_k(x) := \Phi^k_{i_0 j_0(k)}(x) \not\equiv 0$. Then $t_k(a_k) = 0$, so some factor of $t_k(x)$ is a minimal polynomial for a_k. From the bounds above on quantifier elimination, we get that the degree of this minimal polynomial is at most $\exp(R)$ and its coefficients are at most $(\exp(R) \log C)$ bits long. In general, to obtain all n minimal polynomials we will be in the first case for $0 \leq s \leq n$ times, and then be once in the second case. Since R never grows, all the polynomials have degree at most $\exp(R) = \exp(|\mathbb{F}|)$. And the largest coefficient will be at most $\exp(R) \log(\exp(R)^s \cdot C) = \exp(R) \log C = \exp(|\mathbb{F}|)$ bits long. To summarize,

Lemma 22.9 *If $\mathbb{F}$ is satisfiable, there exists a common zero $(a_1, \ldots, a_n) \in \bar{\mathbb{Q}}^n$ such that for any k, the minimal polynomial $q_k(x) \in \mathbb{Z}[x]$ for a_k has at most singly exponential degree d_i and at most doubly exponential coefficients (in $|\mathbb{F}|$).*

It is a classical theorem that if a_1 and a_2 have minimal polynomials of degree d_1 and d_2 and coefficients bounded by C, then the primitive element τ of the joint extension $\mathbb{Q}(a_1, a_2) = \mathbb{Q}(\tau)$ has minimal polynomial of degree at most $d_1 d_2$ and coefficients at most $C^{d_1 d_2}$. Moreover, a_i can be written as $r_i(\tau)/m$, where $m < C^{d_1 d_2}$ and r_i is an integer polynomial, $i = 1, 2$. By adjoining a_1,...,a_n one at a time and using the above lemma about the minimal polynomials of a_i's, our claim on the degree D of R, size of its coefficients and the size of b follows. ■

22.5 Concluding Remarks

The proof presented here is due to Koiran. He also gives an example demonstrating that when $\mathbb{F}$ is satisfiable over $\mathbb{C}^n$, there could be indeed exponentially many primes for which $\mathbb{F}$ is unsatisfiable over $\mathbb{Z}_p^n$. Thus, Theorem 22.2 can not be improved too much. However, no example demonstrating the tightness of Theorem 22.1 is known. We note that the use of ERH was hidden by using the powerful Chebotarev's Density Theorem. While conceivably possible, no one knows how to prove an unconditional bound on $|S_{N,R}|$.

6.966 Algebra and Computation November 30, 1998

Lecture 23

Lecturer: Madhu Sudan *Scribe: Rocco Servedio*

23.1 Introduction

In today's lecture we discussed some lower bounds on the complexity of computing various algebraic functions under several different nonuniform models of computation. Some useful references for today's material are Borodin and Munro's 1975 book "The Computational Complexity of Algebraic and Numeric Problems," Strassen's entry in the Handbook of Theoretical Computer Science on algebraic complexity theory, and von zur Gathen's survey paper.

23.2 Nonuniform Models

The models we will discuss are nonuniform in the sense that a different algorithm may be used for each different input size. Each of the models described below – straight-line programs, algebraic decision trees, and algebraic computation trees – fit into this framework.

23.2.1 Straight-line programs

Let R be an arbitrary ring and let $f : R^n \to R$ be some function. A straight-line program of length t for computing the value of $f(x_1, \ldots, x_n)$ is a sequence of t instructions, where the ith instruction is of the form

$$v_i \leftarrow y_1 \circ y_2.$$

Here each y_i is either some $v_j, j < i$, or is some input $x_j, 1 \le j \le n$, or is some scalar value (an element of R). The operator $\circ$ belongs to $\{+, -, \times\}$ (or may belong to $\{+, -, \times, \div\}$ if R is a field). The output of a straight-line program of length t on input $(x_1, \ldots, x_n)$ is the value of the final variable v_t, and the running time of the program is t. The complexity of such a program can be further broken down into the number of $+$ or $-$ operations performed and the number of $\times$ or $\div$ operations, and can be even further broken down into the number of scalar multiplications (of the form $r \times z$ with $r \in R$) versus nonscalar multiplications.

We will see that it is possible to prove some sharp bounds in this model. However, this is clearly a weak model since there is no way to do any kind of branching.

23.2.2 Algebraic decision trees

Unlike a straight-line program, which computes a function $f : R^n \to R$, an algebraic decision tree computes a boolean function $f : R^n \to \{0, 1\}$. An algebraic decision tree T is a rooted, complete (i.e. each node has either zero or two descendents) binary tree. Each leaf node is labelled with an element of $\{0, 1\}$ and each non-leaf node v is labelled with a predicate "$p_v(x_1, \ldots, x_n) = 0$," where p_v is a polynomial. (If p_v is restricted to be of total degree 1, then we that T is a *linear* decision tree, and if p_v is restricted to be of total degree at most d, we say that T is a *degree-d* algebraic decision tree.) If R is an ordered ring, the label of a non-leaf node can be of the form "$p_v(x_1, \ldots, x_n) \ge 0$." Evaluation proceeds down the tree from the root by going left if the predicate at a node is satisfied and going right if it is not. The complexity of a tree T is the depth of T.

This model clearly allows branching, and in that sense is stronger than the straight-line program model. However, the algebraic decision tree model is weaker in that straight-line programs can compute and use intermediate values, whereas algebraic computation trees can only maintain state information but cannot compute intermediate values.

23.2.3 Algebraic computation trees

Algebraic computation trees compute functions from R^n to R; they combine the branching ability of algebraic decision trees with the storage ability of straight-line programs. Unlike algebraic decision trees, algebraic computation trees can have non-leaf nodes of degree either 1 or 2. A node v of degree 1 is labelled with an instruction of the form "$v \leftarrow y_1 \circ y_2$," where each a_i is either some v' which was computed earlier on the path from the root to v or is some input x_j or is a scalar value, and $\circ$ is a basic arithmetic operation ($+$, $\times$,etc.) A node v of degree 2 is a branching node, labelled with a predicate of the form "$v' = 0$" where v' is an ancestor of v. (As with algebraic decision trees, predicates of the form "$v' \geq 0$" are allowed over ordered rings.) The complexity of an algebraic computation tree is the depth of the tree. Algebraic computation trees clearly generalize straight-line programs, and since polynomials can be built up using intermediate values and can be tested at branch nodes, they generalize algebraic decision trees as well.

23.3 Horner's Rule is Optimal

We begin by considering the straight-line program complexity of evaluating a polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$. Horner's rule for evaluting such a polynomial is to write it in the following form:

$$p(x) = a_0 + x(a_1 + \dots x(a_{n-1} + xa_n) \dots).$$

This algorithm uses n additions and n multiplications. In 1954, A. Ostrowski posed the question of whether any other method of computing $p(x)$ could use fewer operations. He proved that n additions are necessary, but it was not until 1966 that V. Pan showed that n multiplications are necessary as well.

Before proving these results, let us state the problem clearly. The input is a sequence $a_0, \dots, a_n$ of coefficients and an argument x, all taken from some infinite field K. The desired output is the value of $p(x) = \sum_{i=0}^{n} a_i x^i$. The questions we are interested in are the following: suppose that A is a straight-line program with operations $\{+, -, \times\}$ which computes the desired output for all inputs.

1. What is the minimum number of additions or subtractions which A can have?

2. What is the minimum number of multiplications?

23.3.1 Horner's Rule is optimal for additions

To see that at least n additions/subtractions are required, we consider the restriction of algorithm A to inputs which have $x = 1$. This yields a new algorithm A' which, on input $a_0, \dots, a_n$, computes $a_0 + \dots + a_n$; we now use induction on n to prove that A' must perform at least n additions or subtractions.

For the base case $n = 1$, note that if a straight-line program cannot perform any additions or subtractions then it can only compute values of the form $c \cdot a_0^{b_0} \cdot a_1^{b_1}$, where $c \in K$, a_0 and a_1 are the two inputs to the algorithm, and b_0 and b_1 are non-negative integers. Since the equation $c \cdot a_0^{b_0} \cdot a_1^{b_1} = a_0 + a_1$ cannot hold for all values of a_0, a_1 over the infinite field K, it follows that at least one addition/subtraction must be required.

For the inductive step, suppose that A' is a straight-line program which computes $a_0 + \dots + a_n$. Let

$$v_i \leftarrow y_1 \pm y_2$$

be the first addition/subtraction which occurs in A'; as in the last paragraph, each y_i must be of the form $c \cdot a_0^{b_0} \cdots a_n^{b_n}$. Without loss of generality we can assume that a_n appears with nonzero exponent in the expression for y_2. Now if we consider the restriction of algorithm A' to inputs which have $a_n = 0$, we can eliminate the above instruction and obtain a new algorithm A'' which, on input $a_0, \dots, a_{n-1}$, computes $a_0 + \dots + a_{n-1}$. By the induction hypothesis, algorithm A'' must use at least $n - 1$ additions/subtractions, and hence A' must use at least n additions/subtractions.

23.3.2 Horner's Rule is optimal for multiplications

We use the so-called *method of linear independence* to show that at least n multiplications are required to evaluate $p(x)$. To motivate this approach, note that if we consider the space $S = \{\sum c_i a_i : c_i \in K\}$ as a vector space over K, then the n elements $\{a_1, \ldots, a_n\}$ are linearly independent. The method of linear independence works by analyzing the amount of linear independence that can exist among the coefficients of polynomials (in x) that are generated as intermediate computations.

Throughout this section $\ell(a_1, \ldots, a_n)$ denotes a linear combination $\sum_{i=1}^n c_i a_i$ where $c_i \in K$. We begin with the following claim (the proof is left as an exercise for the reader):

Claim 23.1 *Let $\ell_1(a_1, \ldots, a_n), \ldots, \ell_u(a_1, \ldots, a_n)$ each be a linear combination of $a_1, \ldots, a_n$ as described above, and let $\ell(a_2, \ldots, a_n)$ be some linear combination of $a_2, \ldots, a_n$. For $1 \le i \le u$, let $\ell_i'(a_2, \ldots, a_n)$ be $\ell_i(\ell(a_2, \ldots, a_n), a_2, \ldots, a_n)$. Then the following holds: if the collection*

$$\{\ell_1(a_1, \ldots, a_n), \ldots, \ell_u(a_1, \ldots, a_n)\}$$

has rank k over K, then the collection

$$\{\ell_1'(a_2, \ldots, a_n), \ldots, \ell_u'(a_2, \ldots, a_n)\}$$

has rank at least $k - 1$ over K.

We also require the following definition:

Definition 23.2 *We say that a multiplication of two elements of $K[a_1, \ldots, a_n, x]$ is* insignificant *if either both multiplicands belong to $K[x]$, or else one belongs to $K[a_1, \ldots, a_n]$ and the other belongs to K.*

Now we can prove the following lemma, which immediately yields the desired lower bound of n multiplications:

Lemma 23.3 *Let A be a straight-line program over $\{+, -, \times\}$ which, on inputs $a_0, \ldots, a_n, x$, computes*

$$\sum_{i=1}^{u} \ell_i(a_1, \ldots, a_n)x^i + r(x) + \ell_0(a_1, \ldots, a_n)$$

where each ℓ_i is a nontrivial linear combination of the a_i's and $r(x)$ is a polynomial. Let k be the rank of $\{\ell_1, \ldots, \ell_u\}$. Then A must have at least k significant multiplications.

Proof The proof is by induction on k. If $k = 1$, then $u \ge 1$ and then since $\ell_1 \ne 0$ at least one significant multiplication must be performed (without a significant multiplication we can only compute elements of the form $\ell_0(a_1, \ldots, a_n) + r(x)$).

For the inductive step, let A be as described in the statement of the lemma and let k be the rank of $\{\ell_1, \ldots, \ell_u\}$. The first significant multiplication performed by A must be of the form

$$\left[c_0(x) + \sum_{i=1}^{n} c_i a_i\right] \times \left[d_0(x) + \sum_{i=1}^{n} d_i a_i\right]$$

where $c_0(x), d_0(x) \in K[x]$ and $c_i, d_i \in K$ with some $c_i \ne 0$ and some $d_i \ne 0$. Without loss of generality we may assume that $c_1 \ne 0$. Now fix c to be some element of K, and consider the restriction of algorithm A to inputs $(a_0, a_1, \ldots, a_n, x)$ which satisfy

$$a_1 = \frac{c - c_0(x) - \sum_{i=2}^n c_i a_i}{c_1}.$$

Let us write $\frac{c - c_0(x) - \sum_{i=2}^n c_i a_i}{c_1}$ as $\ell(a_2, \ldots, a_n) + p(x)$. Then under this restriction, the program A computes

$$\sum_{i=1}^{u} \ell_i(\ell(a_2,\ldots,a_n)+p(x),a_2,\ldots,a_n)x^i + r(x) + \ell_0(\ell(a_2,\ldots,a_n)+p(x),a_2,\ldots,a_n) = \sum_{i=1}^{u} \ell'_i(a_2,\ldots,a_n)x^i + r'(x) + \ell'_0(a_2,\ldots,a_n)$$

for $\ell'_1,\ldots,\ell'_u$ as defined in Claim 1 and some r', ℓ'_0. Since $c_0(x)+\sum_{i=1}^{n} c_i a_i = c$, under this restriction we obtain a new program A' on inputs $a_0, a_2, \ldots, a_n, x$ which has one fewer significant multiplication than A (since this first significant multiplication is no longer significant). By Claim 1 the rank of $\{\ell'_1,\ldots,\ell'_u\}$ is at least $k-1$, and thus by the induction hypothesis A' must have at least $k-1$ significant multiplications, which implies that A must have at least k significant multiplications. ■

23.4 Complexity of evaluating specific polynomials

In the last section, the inputs to our polynomial evaluation algorithm were the coefficients a_i of the polynomial as well as the argument x. A natural question is whether fewer operations are required for evaluating *specific* polynomials; so now the input to the algorithm is only the value x, and the output is the desired value of $p(x) = \sum_{i=0}^{n} a_i x^i$. (We can view this setup as one in which the algorithm is allowed to do "pre-computation" using the coefficients a_i.) Clearly, there are specific polynomials which can be evaluated using fewer than n additions or multiplications, such as the polynomial $p(x) = x^n$ which can be computed using $\log n$ multiplications and no additions. In this section we will see that every polynomial can be evaluated in this framework using approximately $n/2$ multiplications and that there are polynomials for which any algorithm requires $n/2$ multiplications.

23.4.1 $n/2$ multiplications suffice

For a complete proof of this result, see the book by Borodin and Munro or Volume II ("Seminumerical Algorithms") of Knuth's series; here we provide only an outline. The basic idea is to use an alternative representation (other than a list $a_0, a_1, \ldots$ of coefficients) of the polynomial $p(x)$. More concretely, given any $b_1 \in K$, we can write $p(x)$ as

$$p(x) = (x^2 - b_1)p_1(x) + R_1(x),$$

where $\deg p_1 = \deg p - 2$ and $\deg R_1 \le 1$. Continuing in this fashion, we can write

$$p_1(x) = (x^2 - b_2)p_2(x) + R_2(x)$$

and so on. The crux of the method is to find $b_1, b_2, \ldots$ so that $R_1, R_2, ...$ are each of degree 0 (i.e. $R_1 = c_1, R_2 = c_2, \ldots$). Then we can write

$$\begin{aligned} p(x) &= (x^2 - b_1)p_1 + c_1 \\ &= (x^2 - b_1)((x^2 - b_2)p_2 + c_2) + c_1 \\ &\vdots \\ &= (x^2 - b_1)(\ldots(x^2 - b_{n/2}) + c_{n/2}) + \cdots + c_2) + c_1 \end{aligned}$$

Since we only need to compute x^2 once, this representation requires only $n/2+1$ multiplications.

23.4.2 $n/2$ multiplications can be necessary

To prove that $n/2$ multiplications can be required to evaluate certain polynomials of degree n, we use the *method of algebraic independence,* which is similar to the method of linear independence. Recall the following definitions:

Definition 23.4 *Let K, G be fields with $K \subseteq G$. Elements $g_1, \ldots, g_m \in G$ are said to be* algebraically dependent over K *if there is a nonzero polynomial $p \in K[y_1, \ldots, y_m]$ for which $p(g_1, \ldots, g_m) = 0$. A maximal set of algebraically independent elements of G is called a* transcendence basis *and the size of such a set is the* transcendence degree *of G over K.*

The following lemma is well known:

Lemma 23.5 *Let $H = K(\alpha_1, \ldots, \alpha_t)$ and let $h_1, \ldots, h_m \in H$. If $m > t$ then the elements $h_1, \ldots, h_m$ are algebraically dependent over K.*

We use this lemma to prove the following:

Theorem 23.6 *Let $p(x) = \sum_{i=0}^{n} a_i x^i \in K[x]$ where K is a field of characteristic 0. If there is a straight-line program over $\{+, -, \times\}$ which evaluates $p(x)$ using fewer than $(n+1)/2$ multiplications, then $a_0, \ldots, a_n$ are algebraically dependent over Q (the field of rational numbers).*

We note that this theorem immediately implies that certain polynomials require $n/2$ multiplications, e.g. by taking $a_0, \ldots, a_n$ to be algebraically independent real numbers.

Proof of Theorem 6: Suppose that A is a straight-line program which evaluates $p(x)$ using k multiplications. For $1 \leq i \leq k$ let S_i denote the result of the ith multiplication in the program. Since the program can only do additions and subtractions between computing S_i and S_{i+1}, we have that (taking $S_0 = x$)

$$S_1 = [c_1 + n_1 \cdot S_0] \times [d_1 + m_1 \cdot S_0]$$

where $c_1, d_1 \in K$ and n_1, m_1 are integers. More generally, we have that

$$S_i = [c_i + \sum_{j=0}^{i-1} n_{j,i} \cdot S_j] \times [d_i + \sum_{j=0}^{i-1} m_{j,i} \cdot S_j]$$

where each $c_i, d_i \in K$ and each $n_{j,i}, m_{j,i}$ are integers. It follows that

$$p(x) = \sum_{i=0}^{n} a_i x^i \in Q[x, c_1, \ldots, c_k, d_1, \ldots, d_k],$$

and hence that each $a_i \in Q(c_1, \ldots, c_k, d_1, \ldots, d_k)$. By Lemma 5, if $n+1 > 2k$ then $a_0, \ldots, a_n$ are algebraically dependent over Q, which proves the theorem. ■

Specific polynomials with integer coefficients are known which require $n/2$ multiplications. In 1980, Strassen showed that any straight-line program which computes

$$p(x) = \sum_{i=0}^{n} 2^{2^i} x^i$$

must use at least $n/2$ multiplications. This is an impressive lower bound, since it is exponential in the number of bits required to specify n.

23.5 Lower Bounds for Algebraic Computation Trees

Thus far we have been interested in the complexity of evaluating a polynomial at a particular argument value x. In the problem of computing the Fourier transform of a fixed polynomial $a_0, \ldots, a_n$, the inputs are $n+1$ argument values $x_0, \ldots, x_n$, and the goal is to evaluate the value of the polynomial at all $n+1$ argument values, i.e. to compute

$$\{\sum_{i=0}^{n} a_i x_0^i, \sum_{i=0}^{n} a_i x_1^i, \ldots, \sum_{i=0}^{n} a_i x_n^i\}.$$

Fast algorithms are known which can solve this problem in $O(n \cdot \log n \cdot poly(\log\log n))$ steps for arbitrary inputs x_i, and in the special case when the x_i's are the complex $(n+1)$th roots of unity, an $O(n \log n)$ algorithm is known.

Motivated by these fast algorithms, Strassen studied the question of whether even faster algorithms could exist for this problem. He showed an $\Omega(n \log n)$ lower bound in the straight-line model, which can be extended to an $\Omega(n \log n)$ lower bound in the algebraic computation tree model. In this section we outline the proof of the $\Omega(n \log n)$ lower bound in the algebraic computation tree model. The proof requires some machinery from algebraic geometry, which we review briefly below.

23.5.1 Preliminaries

For the rest of today's lecture we will assume that the ground field K is algebraically closed. In future lectures we will consider fields such as $\mathbf{R}$ which are not algebraically closed.

Recall that if $f_1, \ldots, f_s \in K[x_1, \ldots, x_n]$, then the *variety* generated by these polynomials is

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in K^n \mid f_1(a_1, \ldots, a_n) = \cdots = f_s(a_1, \ldots, a_n) = 0\}.$$

The *dimension* of a variety V, written $\dim(V)$, is the maximum dimension of a linear subspace $L \subseteq K^n$ such that $|L \cap V|$ is finite. The *degree* of a variety V, written $\deg(V)$, is the maximum number of points than can occur in any such intersection. It is easy to verify that if V, W are varieties then $V \cap W$ is a variety as well; Bezout's theorem tells us that

$$\deg(V \cap W) \leq \deg(V) \cdot \deg(W).$$

In our informal exposition here, we will use the above definitions and will use varieties over K^n. However, for a rigorous proof of Strassen's result, one should work over the projective space P^n rather than K^n (we will see why this is necessary in the next section). The space P^n has as its points equivalence classes of points in $K^{n+1}/\{(0, \ldots, 0)\}$ under the equivalence relation $\equiv$, where $(x_0, \ldots, x_n) \equiv (y_0, \ldots, y_n)$ iff for some $0 \neq \lambda \in K$ it is the case that $(x_0, \ldots, x_n) = (\lambda y_0, \ldots, \lambda y_n)$. We write $\langle x_0, \ldots, x_n \rangle$ to represent the equivalence class to which $(x_0, \ldots, x_n)$ belongs. It would take us too far afield to develop all of the necessary machinery for discussing varieties over P^n; a more thorough exposition can be found in the book by Borodin and Munro.

23.5.2 Algebraic computation trees and varieties

In this section we will establish the following bound, which relates the depth of an algebraic computation tree to the degree of the variety of the function which the tree computes.

Lemma 23.7 *Let $f_1, \ldots, f_m \in K[x_1, \ldots, x_n]$ and let $f : K^n \to K^m$ be defined by $f(\overline{x}) = (f_1(\overline{x}), \ldots, f_m(\overline{x}))$. Let $V \subseteq K^{n+m}$ be the variety*

$$V = \{(\overline{x}, f(\overline{x})) \mid \overline{x} \in K^n\}.$$

If $\deg(V) \geq D$, then any algebraic computation tree for f must have depth at least $\log_4 D$.

Proof sketch: Let T be an algebraic computation tree of depth h which computes f, and let P_1 be some path through T from the root to a leaf. A given node on such a path is:

- either a predicate of the form "$v_i = 0$" or "$v_i \neq 0$",
- or an assignment statement of the form "$v_i = v_j \circ v_k$" where $\circ$ is some operation from $\{+, -, \times, \div\}$ and $i, j, k \leq h$.

The key point is that each node on this path corresponds to some polynomial equation of degree at most 2 over the variables $x_1, \ldots, x_n, v_1, \ldots, v_h$. This is clearly the case for a statement of the form "$v_i = 0$;" for a statement of the form "$v_i \neq 0$" we can employ Rabinovich's trick, i.e. introduce a new variable w_i and use the equation $w_i \cdot v_i = 1$ (so in fact our new set of variables will be $\{x_1, \ldots, x_n, v_1, \ldots, v_h, w_1, \ldots, w_h\}$). For an assignment statement "$v_i = v_j \circ v_k$" we use the equation $v_i - v_j \circ v_k = 0$ if $\circ \in \{+, -, \times\}$ and we use the equation $v_i \cdot v_k - v_j = 0$ if $\circ$ is $\div$.

Thus, the set of those points $(x_1,\ldots,x_n,v_1,\ldots,v_h,w_1,\ldots,w_h)$ which correspond to a particular path through the tree will be the variety generated by a set of at most h polynomials:

$$\begin{aligned} p_1(x_1,\ldots,x_n,v_1,\ldots,v_h,w_1,\ldots,w_h) &= 0 \\ &\vdots \\ p_h(x_1,\ldots,x_n,v_1,\ldots,v_h,w_1,\ldots,w_h) &= 0. \end{aligned}$$

By Bezout's theorem, the degree of this variety can be at most 2^h (since the degree of each polynomial is at most 2). However, this is a variety over the $(n+2h)$-dimensional space of points $(x_1,\ldots,x_n,v_1,\ldots,v_h,w_1,\ldots,w_h)$, which is a larger space than we are really interested in. We would like to consider the set of points

$$V_{P_1} = \{(\overline{x}, f(\overline{x})) \mid (x_1,\ldots,x_n) \text{ goes down path } P_1\}.$$

This is the projection of the original variety onto the coordinates $x_1,\ldots,x_n,v_{i_1},\ldots,v_{i_m}$, where $i_1,\ldots,i_m$ are such that $(v_{i_1},\ldots,v_{i_m}) = f(\overline{x})$. To analyze such a variety we require the following fact:

Fact 23.8 *Let $\Pi : P^n \to P^{n-1}$ be the mapping which takes $\langle x_0,\ldots,x_n\rangle$ to $\langle x_0,\ldots,x_{n-1}\rangle$. Let V, W be varieties over P^n, P^{n-1} respectively. Then $\Pi(V), \Pi^{-1}(W)$ are varieties over P^{n-1}, P^n respectively and*

$$\begin{aligned} \deg(\Pi(V)) &\le \deg(V) \\ \dim(V) - 1 &\le \dim(\Pi(V)) \le \dim(V) \\ \deg(\Pi^{-1}(W)) &= \deg(W) \\ \dim(\Pi^{-1}(W)) &= \dim(W) + 1. \end{aligned}$$

(We note that projections of varieties over K^n are not necessarily themselves varieties; this is why we must work over P^n.) Using this fact, we have that the degree of V_{P_1} is also at most 2^h. But recall that V_{P_1} is the variety generated by a single path P through the computation tree. Since the tree T has depth h, there are at most 2^h different paths through the computation tree, and hence we can write

$$V = V_{P_1} \cup \cdots \cup V_{P_{2^h}} = \{(\overline{x}, f(\overline{x}))\}.$$

Since $\deg(A \cup B) \le \deg(A) + \deg(B)$ for varieties A, B, we have that $\deg V \le 2^h \cdot 2^h = 4^h$. ∎

23.5.3 A lower bound for the Fourier transform

Using Lemma 7 we can show the following, which yields the desired $\Omega(n \log n)$ complexity lower bound for the Fourier transform.

Claim 23.9 *Let $p \in K[x]$ be a univariate polynomial of degree m. Then any algebraic computation tree which evaluates $p(x_1),\ldots,p(x_n)$ must have depth $\Omega(n \log m)$.*

Proof sketch: Let

$$V = \{(x_1,\ldots,x_n,p(x_1),\ldots,p(x_n)) \mid (x_1,\ldots,x_n) \in K^n\}.$$

By Lemma 7, it suffices to show that $\deg(V) \ge m^n$. Let $c \in K$ be such that the equation "$p(x) - c = 0$" has m distinct roots $\alpha_1,\ldots,\alpha_m$ (we leave it as an exercise for the reader to show that such a c must exist). For $1 \le i \le n$, define

$$H_i = \{(y_1,\ldots,y_n,z_1,\ldots,z_{i-1},c,z_{i+1},\ldots,z_n) \mid y_j \in K, z_j \in K\}.$$

Each H_i is a variety of degree 1. Let $V' = V \cap (\cap_{i=1}^n H_i)$; by Bezout's theorem,

$$\deg(V') = \deg(V \cap (\cap_{i=1}^n H_i)) \le \deg(V) \cdot \left(\prod_{i=1}^n \deg(H_i)\right) = \deg(V).$$

However,

$$V' = \{(r_1,\ldots,r_n,c,\ldots,c) \mid r_i \in \{\alpha_1,\ldots,\alpha_m\}\}.$$

Hence $|V'| = m^n$. Since V' is contained a linear subspace of K^{2n}, we have that $\deg(V') = m^n$, which proves the claim. ∎

6.966 Algebra and Computation December 2, 1998

Lecture 24

Lecturer: Madhu Sudan *Scribe: Sofya Raskhodnikova*

In this lecture we investigate topological methods for obtaining lower bounds on the depth of real computation trees which decide the membership problem for a subset Π of $\mathbb{R}$. We study the theorem due to Ben-Or which gives a lower bound in terms of the number of connected components of Π and apply it to several problems. We also discuss lower bounds in terms of volume and mention a generalization of Ben-Or's technique to Euler characteristic and Betti numbers.

24.1 Connected Components

24.1.1 Model

All computation trees we consider in this lecture are defined over the reals and allow both arithmetic operations and tests for equality and inequality to zero. The lower bound technique we are going to study is so powerful that we will count only the operations from $\{\times, \div, \sqrt{}, = 0, \geq 0\}$. Each of these operations can be done for a unit cost. Addition, subtraction, and multiplication by scalars from $\mathbb{R}$ are performed for free.

24.1.2 Motivating Example and Statement of the Ben-Or's Theorem

Example 6 [Element Distinctness Problem] *Given n real numbers, determine if two of them are equal. Equivalently and more formally,*

$$ED_n = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid \prod_{i \neq j}(x_i - x_j) = 0\}.$$

Element Distinctness can be solved in $O(n \log n)$ time given ≥ 0 and $-$ operations. In comparison-based model it is not hard to show that Element Distinctness requires $\Omega(n \log n)$ comparisons. However, comparison-based model does not allow serious computation. We will investigate whether a less costly solution to this problem is feasible in our model.

Steel and Yao proved a lower bound of $\Omega(n \log n)$ for the case when only "linear" operations are allowed. Today we will study a more general result by Ben-Or which uses topological invariants to obtain a lower bound on the complexity of real computation trees.

Theorem 24.1 [Ben-Or] *If* Yes *instances of some problem* Π *have* N *distinct connected components in* $\mathbb{R}^n$*, then the depth of the real computation tree for this problem is* $\Omega(\log N - n)$.

Applying Theorem 24.1 to the complement of Π, further denoted as $\overline{\Pi}$, we get the same statement for No instances.

Let us give an application of this result before we attempt to prove it.

Theorem 24.2 *Any algebraic computation tree solving Element Distinctness problem for n numbers must have depth of* $\Omega(n \log n)$.

Proof Yes instances of this problem (the points in ED_n) form one connected component in $\mathbb{R}^n$ because ED_n contains a continuous path from any point with two equal coordinates to the origin. Therefore, in this case, applying Theorem 24.1 to the Yes instances of the problem is not particularly useful for obtaining a lower bound.

However, the number of connected components, N, for No instances of Element Distinctness (the points in $\overline{ED_n}$) is $n!$ because we have one component for every permutation $\pi : [n] \to [n]$. To see this, observe that any No instance $(x_1, \ldots, x_n)$ satisfying condition $x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}$ is connected only to No instances for which this condition still holds. Applying Theorem 24.1, we get a lower bound of $\Omega(n \log n)$. ■

24.1.3 Bounding the Number of Connected Components of a Semi-Algebraic Set

Definition 24.3 *A subset S of $\mathbb{R}^N$ is called* **semi-algebraic** *if it is the set of all real solutions to a system of finitely many polynomial equalities and inequalities or a finite union of such sets.*

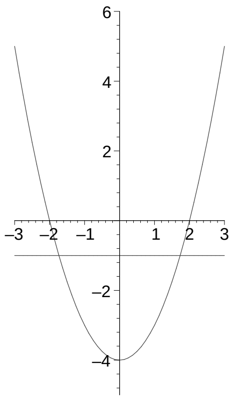
We are interested in bounding the number of connected components of a semi-algebraic set of the form

$$S = \left\{ \overline{x} \in \mathbb{R}^M \;\middle|\; \begin{array}{l} p_1(\overline{x}) = 0, \ldots, p_l(\overline{x}) = 0, \\ q_1(\overline{x}) \geq 0, \ldots, q_{K_1}(\overline{x}) \geq 0, \\ r_{K_1+1}(\overline{x}) < 0, \ldots, r_K(\overline{x}) < 0 \end{array} \right\} \tag{24.1}$$

To see that the number of connected components in the set S of this form can be more than one, consider the following example.

Example 7 *Look at the set of points in $\mathbb{R}^2$ defined by*

$$\begin{aligned} y - x^2 + 4 &\leq 0 \\ y + 1 &\geq 0 \end{aligned}$$



There are two distinct connected components. Both lie above the line $y = -1$. One is positioned to the left of the parabola $y = x^2 - 4$, the other is to the right of the parabola.

To derive an upper bound on the number of connected components of a semi-algebraic set given by a system of the form (24.1), we use without proof the following very powerful theorem.

Theorem 24.4 [Milnor-Thom] *(Real varieties).*
Let V be a variety in $\mathbb{R}^M$ defined by polynomial equalities $h_1 = 0, h_2 = 0, \ldots, h_L = 0$ of degree at most D. Then the number of connected components of V is at most $D(2D-1)^{M-1}$.

Notice that this statement resembles Bezout's Theorem. However, here we are not working over algebraically closed fields and instead of the number of irreducible components we are counting the number of connected components. This theorem is more powerful than Bezout's. Also observe that L, the number of polynomial equations, does not enter the bound.

As a corollary of Milnor-Thom theorem, we get the following result.

Theorem 24.5 *The number of connected components in S is at most $D(2D-1)^{M+K-1}$, where D is the maximum degree of polynomials defining S.*

Proof The main idea of the proof is to convert the inequalities defining the variety to equalities without decreasing the number of connected components, and then to apply the Minor-Thom Theorem.

Let r be the number of connected components. Pick points $v_1, \ldots, v_r$ from distinct connected components.

We start by converting strict inequalities to non-strict. Let $\epsilon = \max_{i,j}\{-r_i(v_j)\}$ where i ranges from $K_1 + 1$ to K and j is from 1 to r. Since $v_j \in S$, we know that $r_i(v_j) < 0$, and hence ϵ is negative. Replace $r_i < 0$ with $r_i \leq \epsilon$ or, equivalently, with $\epsilon - r_i \geq 0$. Set $q_i = \epsilon - r_i$.

Let S' be the variety defined by the modified set of polynomial equalities and inequalities. Then $v_1, \ldots, v_r \in S'$ and S' is contained in S. Since S' is a subset of S, all v_js are in distinct connected components of S'. Therefore,

$$\mathcal{CC}(S) \leq \mathcal{CC}(S'). \tag{24.2}$$

We are left with K non-strict inequalities of the form $q_i \geq 0$. We replace them with equalities of the form $q_i = z_i^2$ where z_i's are new variables. Let S'' be the variety in $\mathbb{R}^{M+K}$ defined by the modified set of polynomial equalities. Notice that S' is a projection of S'' onto the first n coordinates. By a simple topological fact[16],

$$\mathcal{CC}(S') \leq \mathcal{CC}(S''). \tag{24.3}$$

Finally, applying Milnor-Thor Theorem to $S'' \subseteq \mathbb{R}^{M+k}$ and combining the result with inequalities 24.2 and 24.3, we get

$$\mathcal{CC}(S) \leq \mathcal{CC}(S') \leq \mathcal{CC}(S'') \leq D(2D-1)^{M+K-1}.$$

■

24.1.4 Proof of the Ben-Or's Theorem

Now we are ready to derive the main theorem of the lecture.
Proof of Theorem 24.1 The idea of the proof is to look at the decision tree.

- Fix a decision tree T that decides Π. Let h be the depth of T.
- Fix an accepting leaf L in T.
- Consider set S of inputs in $\mathbb{R}^n$ whose path in T leads to L.
- **Bound the number of connected components in S.**
- Bound the number of connected components in Π based on the previous step.

The core of the proof is bounding the number of connected components in S.

Set S can be described by a system of equalities and inequalities. We assign indeterminates $x_1, \ldots, x_n$ to the inputs. Each node from the root to leaf L gets assigned a value v_i depending on the operation it performs. If it computes one of the free operations, its value is one of the following

$$v_i = c$$
$$v_i = v_j \pm v_k$$
$$v_i = c \times v_j$$

where c is a constant and v_i and v_j are values of the node's ancestors. If it computes $\times, \div$ or $\sqrt{}$, it is assigned an indeterminate y_i. Each node with a multiplication instruction $v_i = v_j \times v_k$ is associated with equation $y_i - v_j * v_k = 0$, each node with a division instruction $v_i = v_j \div v_k$ is associated with equation $y_i v_k - v_j = 0$, and each node with a square root instruction $v_i = \sqrt{v_j}$ is associated with equation $y_i^2 - v_j = 0$.[17]

[16] If $\mu : S' \to S''$ is a continuous surjective map of topological spaces, then $\mathcal{CC}(S') \leq \mathcal{CC}(S'')$.

[17] If we are interested in enhancing our model, we can easily add new operations. For example, the operation of taking the k-th square root can be handled as follows. The node with a k-th square root instruction $v_i = \sqrt[k]{v_j}$ would be associated with $O(\log k)$ equations of degree ≤ 2, each introducing a new variable s_i: $v_i = s_0, s_1 = s_0^2, s_2 = s_1^2, \ldots, v_j = s_{i'} \cdot s_{j'}$. The cost of this operation should be equal to the number of variables introduced, $O(\log k)$. We can add any operation of taking roots of a polynomial in a similar fashion.

A branching node with a test instruction of the form $v_i \geq 0$ is assigned the inequality $v_i \geq 0$ ($v_i < 0$) if the path proceeds along the YES (NO) branch, respectively. Finally, to a branching node with a test instruction of the form $v_i = 0$? we assign the equality $v_i = 0$ if the path proceeds along the YES branch. If the path continues with the NO branch, we employ (guess what) Rabinovich's trick, i.e. introduce a new indeterminate y_i and assign the equality $y_i v_i - 1 = 0$ to the node.

Observe that all introduced equalities and inequalities have degree at most two.

$$D = 2$$

The number of inequalities, call it k, is equal to the number of ≥ 0 instructions along the path to L. The number of indeterminates y_i, call is m, is bounded above by the number of operations from $\{\times, \div, \sqrt{}, = 0\}$. Since each tree node introduces either at most one new indeterminate or at most one inequality, we know that

$$k + m \leq h,$$

where h is the depth of T. Now S can be defined using above equalities and inequalities as follows.

$S = \{x_1, \ldots, x_n \mid \exists y_1, \ldots y_m$ satisfying

$$\begin{aligned} p_1(\overline{x}, \overline{y}) = 0, \ldots, p_l(\overline{x}, \overline{y}) = 0, \\ q_1(\overline{x}, \overline{y}) \geq 0, \ldots, q_{k_1}(\overline{x}, \overline{y}) \geq 0, \\ r_{k_1+1}(\overline{x}, \overline{y}) < 0, \ldots, r_k(\overline{x}, \overline{y}) < 0\}. \end{aligned}$$

Let E be the set of solutions in $\mathbb{R}^{n+m}$ of the above equations and inequalities.

$E = \{x_1, \ldots, x_n, y_1, \ldots y_m$ satisfying

$$\begin{aligned} p_1(\overline{x}, \overline{y}) = 0, \ldots, p_l(\overline{x}, \overline{y}) = 0, \\ q_1(\overline{x}, \overline{y}) \geq 0, \ldots, q_{k_1}(\overline{x}, \overline{y}) \geq 0, \\ r_{k_1+1}(\overline{x}, \overline{y}) < 0, \ldots, r_{k_2}(\overline{x}, \overline{y}) < 0\}. \end{aligned}$$

Notice that S is a projection of E onto the first n coordinates. Hence, $\mathcal{CC}(V) \leq \mathcal{CC}(E)$. Applying Theorem 24.5 to E and substituting $D = 2$, we obtain

$$\mathcal{CC}(S) \leq \mathcal{CC}(E) \leq D(2D-1)^{M+K-1} = 2 \cdot 3^{n+m+k-1} \leq 2 \cdot 3^{n+h-1} \leq 3^{n+h}.$$

Since the number of accepting leaves in T is at most 2^h and each leaf has at most 3^{n+h} connected components, the total number of connected components is

$$N \leq 2^h 3^{n+h}.$$

Consequently,

$$h = \Omega(\log N - n).$$

■

24.1.5 Applications

Discriminant

Example 8 [Computing Discriminant] *Given real numbers $x_1, \ldots, x_n$, evaluate the discriminant polynomial $\prod_{i \neq j}(x_i - x_j)$.*

Theorem 24.6 *An algorithm for this problem requires $\Omega(n \log n)$ operations.*

Proof Observe that $(x_1, \ldots, x_n)$ is a YES instance of the Element Distinctness problem if and only if $\prod_{i \neq j}(x_i - x_j) = 0$. Therefore, our lower bound of $\Omega(n \log n)$ for the Element Distinctness problem implies the same lower bound for this problem. ■

The Convex Hull Problem

Let C be a convex set. A point $x \in C$ is an *extreme* point of C if there are no points y and z in C such that x lies on the open line segment between y and z.

Example 9 [The Convex Hull Problem in $\mathbb{R}^n$] *Given n points on the plane, determine if their convex hull has n extreme points, i.e. no point belongs to the convex hull of other $n-1$ points. Formally,*

$$CH_n = \{(x_1, \ldots, x_n) \in (\mathbb{R}^2)^n \mid \textit{the convex hull of } x_1, \ldots, x_n \textit{ has } n \textit{ extreme points}\}.$$

There is an algorithm that decides this problem with $O(n \log n)$ operations in the worst case. We show that this algorithm is asymptoticly optimal.

Theorem 24.7 *Any computation tree solving the Convex Hull Problem has worse case complexity $\Omega(n \log n)$.*

Proof Consider $X \in CH_n$. Let O denote the center of mass of $x_1, \ldots, x_n$. Since $x_1, \ldots, x_n$ form a convex set, O lies inside the region defined by $x_1, \ldots, x_n$. For i from 2 to n, let θ_i be the angle $x_1 O x_i$. For each of $(n-1)!$ permutations $\pi : [n-1] \to [n-1]$ of θ_is, there is a corresponding connected component in CH_n. To see this, observe that any X satisfying condition $\theta_{\pi(2)} < \theta_{\pi(3)} < \cdots < \theta_{\pi(n)}$ is connected only to YES instances for which this condition still holds. Applying Theorem 24.1, we get a lower bound of $\Omega(n \log n)$. ■

The Knapsack Problem

For some problems the Ben-Or's technique enables us to procure stronger lower bounds than $\Omega(n \log n)$.

Example 10 [The Knapsack Problem] *Given a set of n real numbers, determine if any its subset sums up to 1. Formally,*

$$KS_n = \{(x_1, \ldots, x_n) \in \mathbb{R}^n \mid \exists v \in \{0,1\}^n \textit{ such that } \sum_{i=1}^{n} x_i v_i = 1\}.$$

We know that Knapsack is **NP**-complete in Boolean world. Surprisingly, this problem can be solved with a computation tree of depth $O(n^5 \log^2 n)$ over the reals. We will prove a lower bound of $\Omega(n^2)$ for it.

Theorem 24.8 *The depth of an algebraic computation tree for the Knapsack problem has to be $\Omega(n^2)$.*

Proof The YES instances are connected. But luckily, the NO instances have a huge number of connected components. It is easy to see that two points $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are in different connected components of $\overline{KS_n}$ if there exists $v \in \{0,1\}^n$ such that

$$\sum_{i=1}^{n} x_i v_i > 1 \quad \text{and} \quad \sum_{i=1}^{n} y_i v_i < 1.$$

The remainder of the proof is devoted to obtaining a good lower bound on the $\mathcal{CC}(\overline{KS_n})$.

Let B_n be the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$. Each NO instance naturally corresponds to a Boolean function f: if $\sum_{i=1}^{n} x_i v_i < 1$, then $f(v) = 0$; if $\sum_{i=1}^{n} x_i v_i > 1$, then $f(v) = 1$. For each f define

$$\begin{aligned} C_f = \{x \mid x \in \mathbb{R}^n \text{ and } \forall v \in \{0,1\}^n \ & \sum_{i=1}^{n} x_i v_i < 1 \text{ if } f(v) = 0 \text{ and} \\ & \sum_{i=1}^{n} x_i v_i > 1 \text{ if } f(v) = 1\}. \end{aligned} \tag{24.4}$$

Intuitively, C_f is the collection of NO instances for which f is the corresponding Boolean function.

Notice that $\overline{KS_n} = \bigcup_{f \in B_n} C_f$ and the C_fs are open and pairwise disjoint. Therefore, the number of nonempty C_fs is a lower bound on $\mathcal{CC}(\overline{KS_n})$. In other words, $\mathcal{CC}(\overline{KS_n}) \geq |G_n|$, where $G_n = \{f \mid f \in B_n$ and $C_f \neq \emptyset\}$.

Definition 24.9 *We say $f \in B_n$ is a* **threshold function** *if $\exists x_1, \ldots, x_n, \theta \in \mathbb{R}$ such that*

$$\forall v \in \{0,1\}^n \quad \sum_{i=1}^{n} x_i v_i < \theta \text{ if } f(v) = 0 \text{ and}$$
$$\sum_{i=1}^{n} x_i v_i > \theta \text{ if } f(v) = 1,$$

where θ is called a **threshold** *and $(x_1, \ldots, x_n)$ is a* **weight vector** *for f.*

Claim 24.10 *C_f is non-empty iff f is a threshold function with $f(\overline{0}) = 0$.*

Proof If $C_f \neq \emptyset$, then there exists a vector x satisfying (24.4). Therefore f is a threshold function with threshold 1 and weight vector x. Since $\sum_{i=1}^{n} x_i \cdot 0 = 0 < 1$, $f(\overline{0}) = 0$.

If f is a threshold function with $f(\overline{0}) = 0$, then the threshold must be greater than $\sum_{i=1}^{n} x_i \cdot 0 = 0$. Divide the weight vector by the threshold to obtain x that satisfies (24.4). ∎

Let T_n be the set of the threshold functions in B_n. The number of threshold functions f with $f(\overline{0}) = 1$ is equal to the number of threshold functions f with $f(\overline{0}) = 0$ because there is a one-to-one map between these two sets that takes $f(v)$ to $1 - f(v)$ (negate $x_1, \ldots, x_n, \theta$ to obtain a weight vector and a threshold for the new function). Consequently,

$$|T_n| = 2|G_n|. \tag{24.5}$$

Now our goal is to find a lower bound for $|T_n|$.

We will call a vector x *diverse* if all sums $\sum_{i=1}^{n} x_i v_i$ have different values for distinct v. Namely,

$$\forall v^{(1)} \neq v^{(2)} \in \{0,1\}^n \quad \sum_{i=1}^{n} x_i v_i^{(1)} \neq \sum_{i=1}^{n} x_i v_i^{(2)},$$

$$\text{or equivalently, for all non-zero } u \in \{-1,0,1\}^n \quad \sum_{i=1}^{n} x_i u_i \neq 0.$$

Given a weight vector for f we can wiggle its coordinates to obtain a diverse weight vector for f with the same threshold. By above argument $f \in T_n$ iff f has a diverse weight vector.

Claim 24.11 *If $x \in \mathbb{R}^n$ is a diverse vector, then it is a diverse weight vector for at least $2^n + 1$ threshold functions.*

Proof Since all sums $\sum_{i=1}^{n} x_i v_i$ have distinct values, they split $\mathbb{R}$ into $2^n + 1$ open intervals. Taking threshold θ from different intervals and setting $f(v) = 0$ iff $\sum_{i=1}^{n} x_i v_i < \theta$, we obtain $2^n + 1$ distinct threshold functions, as required. ∎

Claim 24.12 $|T_n| \geq 2^{\binom{n}{2}+2}$.

Proof Consider the map $B_n \times B_n \to B_{n+1}$ which takes $f, g \in B_n$ to

$$h(v_1, \ldots, v_{n+1}) = f(v_1, \ldots, v_n) v_{n+1} + g(v_1, \ldots, v_n)(1 - v_{n+1}).$$

Notice that $h \in T_{n+1}$ iff f and g have a common diverse weight vector. Hence, using Claim 24.11 we get

$$|T_{n+1}| \geq (2^n + 1)|T_n|.$$

Unfolding the recursion, we obtain

$$|T_n| \geq 2^{n-1} \cdot 2^{n-2} \cdot \cdots \cdot 2 \cdot |T_1| = 2^{\binom{n}{2}+2}.$$

∎

Claim 24.12 and equation 24.5 give a lower bound of $2^{\binom{n}{2}+1}$ on the number of connected components of $\overline{KS_n}$. By Ben-Or's Theorem, the worst-case complexity of the Knapsack is $\Omega(n^2)$. ∎

24.1.6 Limit of Connected Components Technique

We can prove a good lower bound for Knapsack using the Connected Components Technique, but it has a relatively low complexity in this model. Unfortunately, this technique is not strong enough for harder problems.

Suppose YES instances come from a semi-algebraic set S defined by degree D polynomials in $\mathbb{R}^N$. By Theorem 24.5 derived from the Milnor-Thom result, the number of connected components of S is at most $D(2D-1)^{N-1}$. Therefore, the best lower bound we can hope to prove using Ben-Or's theorem is $\Omega(N \log D)$. How large can D be for an $\mathbf{NP}_{\mathbb{R}}$-complete problem? D is at most exponential in the running time of the non-deterministic machine. Therefore the best lower bound we can prove is polynomial in the size of the input. This technique is too weak to prove $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$.

24.2 Volume

24.2.1 k-Distinctness Problem

INPUT: $(x_1, \ldots, x_n) \in \mathbb{R}^n$.
YES INSTANCES: $\exists$ distinct $i_1, \ldots, i_k$ such that $x_{i_1} = x_{i_2} = \cdots = x_{i_k}$.

For $k = 2$ we already proved a lower bound of $\Omega(n \log n)$. However, as k increases, the problem becomes easier. In general, we can solve it with $O(n \log \frac{n}{k})$ comparisons.

Now both YES and NO instances form one connected component for $k \geq 2$. So the Connected Components method does not apply.

Bjorner, Lovasz and Yao '91 introduce a new technique which applies only to linear decision trees (each node can question if a linear function is greater or equal to zero). They look at volume of YES and NO instances to establish a lower bound. We will give an outline of their method.

- Fix a linear decision tree T that decides Π..
- Fix an accepting leaf L in T.
- The set of inputs in $\mathbb{R}^n$ whose path in T leads to L is convex because it is defined by linear equations and inequalities.
- Look at the largest convex set V defined by accepting leaves. Let U be an upper bound on the volume of V.
- Let L be a lower bound on the total volume of YES instances.
- The depth of the linear decision tree is lower bounded by $\log \frac{U}{L}$.

Using the Volume technique, we can obtain an optimal lower bound of $\Omega(n \log \frac{n}{k})$ for $k < \sqrt[4]{n}$ for the k-Distinctness Problem.

24.3 Euler Characteristic and Betti Numbers

The number of connected components is also known as the 0th Betti number. Defining higher Betti numbers is beyond the scope of this lecture, but we will mention that the method of connected components can be generalized to higher Betti numbers and Euler characteristics (another useful notion from topology). As with the number of connected components, log of Euler characteristic is a lower bound on the computation time. Higher Betti numbers are used similarly for obtaining lower bounds. The generalized method gives super-linear lower bounds for a broader class of problems.

6.966 Algebra and Computation December 7, 1998

Lecture 25

Lecturer: Madhu Sudan *Scribe: Amit Sahai*

25.1 Introduction

In this lecture, we introduce Mulmuley's Algebraic Parallel computation model (Algebraic PRAM), and give a very high level overview of Mulmuley's Lower Bound techniques in this model without bit operations. For more details, see Mulmuley's paper "Lower Bounds in a Parallel Model without bit operations," available at `http://www.cs.uchicago.edu/~mulmuley/`.

25.2 Algebraic PRAM Model

Mulmuley introduces the Algebraic PRAM model, which we will consider in the non-uniform setting. We think of each of p processors as being defined by a separate algebraic computation tree, each with some specified topological ordering indicating the sequence in which nodes are to be evaluated. We allow the intermediate variables of any processor at time t to be used by all other processors at time $t+1$. We think of all operations as being done over the integers; thus rational computations are possible since they can be simulated by integer operations on a pair of integers corresponding to the numerator and denominator of a rational number. We allow the computation to use the standard algebraic operations, $+, -, *$ and branching operations based on ≥ 0 (and hence also $= 0$). Note that all these operations and / can be simulated for rationals with only a constant factor overhead. For simplicity in certain arguments, we will also consider the *Linear* PRAM model, in which we require that one of the operands in any $*$ operation be a constant.

The main novelty of the model is that we allow the running time and number of processors to depend not only on the number of integers n in the input, but also the *bit length* N of the input. On the other hand, we only charge a unit cost for every algebraic operation, regardless of the bit lengths of the operands.

The main restriction of this model is that bit operations are not allowed. Note, however, that the i'th bit can be extracted in $O(N)$ time by repeated $*, -$ and ≥ 0 operations. It is not clear how this can be significantly speeded up in the parallel setting, however. Indeed, we will show an effective lower bound for this problem later on.

25.3 Lower Bound Results of Mulmuley

Mulmuley shows that several natural problems in P, in particular MAX-FLOW, MINCOST FLOW, and LINEAR PROGRAMMING, cannot be solved in time $c\sqrt{n}$ with $2^{c\sqrt{n}}$ processors for some positive constant c, where n denotes either the number of nodes in the underlying graph or the total number of variables and constraints. In particular, this gives a separation of Algebraic-NC from P. Note NC is the class of languages that can be recognized by machines with a polynomial number of processors running in polylogarithmic time.

A key idea in Mulmuley's work is to look at a decision problem restricted to a very low dimensional space, and apply algebraic lower-bound techniques on the resulting problem.

25.4 Intuition: Lower Bounds for $\lfloor x \rfloor$

Is there hope in applying algebraic lower-bound techniques in very small dimensions? We begin by looking at one place we can apply the connected-components technique from the last lecture on a very low-dimensional problem. We consider the problem of determining whether some number $y \geq \lfloor x \rfloor$. This problem is clearly very closely related to the bit representation of x so we expect it to be hard. However, the YES set has only one connected component. We can easily get around this problem by adding the additional constraint that $y < x$.

More formally, let x and y be given rational numbers (each given by a pair of integers). We will consider YES instances to be those where $y < x$ but $y \geq \lfloor x \rfloor$, as shown in Figure 1 below. The first thing to note is that the YES set has infinitely many connected components, so in the usual computation model over the reals (not the one considered here, where time is measured in terms of the bit size of the inputs), by the theorem proved in the last lecture, this problem is undecidable. Of course, this should not surprise anyone, since this problem is essentially a problem about bits, so its undecidability in a model without bit operations is much like the impossibility of computing the NOT function with a monotone circuit.

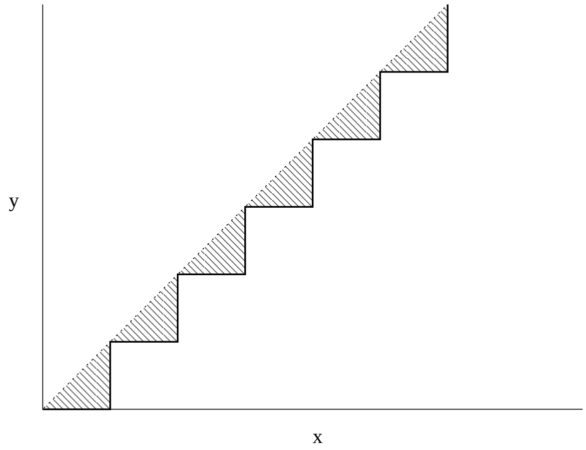


Figure 1.

Remembering that we have a bound N on the number of bits needed to represent x and y, we see that this implies that $|x| \leq 2^N$ and $|y| \leq 2^N$. Hence, the number of connected components of the YES set is $2^{\Omega(N)}$, so it seems that we arrive at an $\Omega(n)$ lower bound for the *sequential* time complexity of this problem. Even this, however, is not quite correct, as the lower bound theorem from the last lecture only applies to computations with real numbers, not rationals with bounded precision. Indeed, it could be that the restriction of a computational problem to bounded precision inputs becomes much simpler. The techniques used by Mulmuley to deal with this problem of finite precision inputs are technically very involved, and we will give only the simplest intuition as to how this problem is dealt with at the end of the lecture.

For now, let us assume these types of arguments can be made to work in the bounded precision model. The challenge is to now convert this sequential lower bound into one for *parallel* computation. We will show how to achieve the bound that this problem cannot be solved in time $c\sqrt{N}$ with $2^{c\sqrt{N}}$ processors for some positive constant c.

25.5 Parametric Complexity and MAX-FLOW

We would also like to generalize arguments like the one above to work for more natural problems. As one might suspect, however, we cannot hope for low-dimensional natural problems to break into many connected components so easily. Instead, we will focus on problems where the surface separating YES instances from NO instances is piecewise-linear, and consider the number of *slope changes* in this surface. Mulmuley is able to show that a lower bound based on the number of slope changes holds similar to the one known based on the number of connected components. We will only be able to give high-level intuition for why such an argument might work in this lecture. In the technical part of this lecture, we will instead concentrate on showing that a large number of connected components imply lower bounds on parallel algebraic complexity. Now, we will see how natural problems can yield low-dimensional problems with many slope changes. We consider, for example, the problem of MAX-FLOW:

MAX-FLOW: Given a graph $G = (V, E)$ on n vertices, with distinguished vertices s and t, rational capacities $C = \{c_e | e \in E\}$, and a rational number f, decide if there exists a flow from s to t in G of value at least f.

This is a problem of high dimension, so the first thing we do is restrict to a low dimensional version of the problem. For a fixed such graph $G = (V, E)$ and sets of rational numbers $A = \{a_e | e \in E\}$ and $B = \{b_e | e \in E\}$, we define the problem:

PARAMETRIZED-MAX-FLOW: Given a rational number λ and rational number f, decide if there exists a flow of value at least f, in the graph G with capacities $C(\lambda) = \{c_e = a_e + \lambda b_e\}$.

We will denote by $F(\lambda)$ the maximum flow value in the graph G with capacities $C(\lambda)$. These types of problems have been studied in the context of "Parametric Complexity," and it is interesting that a connection exists between that area and Algebraic Complexity. In particular, for this problem, it is known that there exist graphs G on n vertices and sets of rationals $A = \{a_e | e \in E\}$ and $B = \{b_e | e \in E\}$, such that each rational in the sets A and B can be specified with $O(n)$ bits, and such that the function $F(\lambda)$ is piecewise linear with $2^{\Omega n}$ slope changes, all occuring at values of λ specified by $O(n^2)$ bits.

25.6 The Lower Bound

To summarize, there are three mountains we must climb in order to establish lower bounds for parallel computation with restricted bit lengths. These are:

1. To translate the known argument for lower bounds for sequential computation to one for parallel computation. In particular, we must show that p processors and 1 unit of time do not give as much power as 1 processor with p units of time.

2. To adapt the lower bound techniques for computations with infinite-precision inputs to ones for inputs of bounded precision.

3. To adapt the lower bound techniques to work with the complexity measure of the number of slope changes in a piecewise-linear convex function, rather than the number of connected components.

We will concentrate on showing how to do Item 1. We will give some intuition for Item 2 at the end of the lecture. As for Item 3, we give only the following high-level intuition:

When we were considering the problem of determining whether $y \geq \lfloor x \rfloor$ – a piecewise constant constraint with many discontinuities – initially we had only one connected component. We got around this problem by adding the simple linear constraint $y < x$. Similarly for piecewise linear constraints with many slope changes, we could imagine adding constraints that would break it up into many connected components.

25.6.1 Linear PRAM

Here, we will consider establishing a lower bound in the Linear PRAM model, where the PRAM can only make linear computations of its inputs. Thus, each branching in the computation divides the space of possible inputs into two linear halfspaces.

With a single processor, after t time steps, it is easy to see that the space of possible inputs can be divided into 2^t regions, as shown pictorially below in Figure 2.

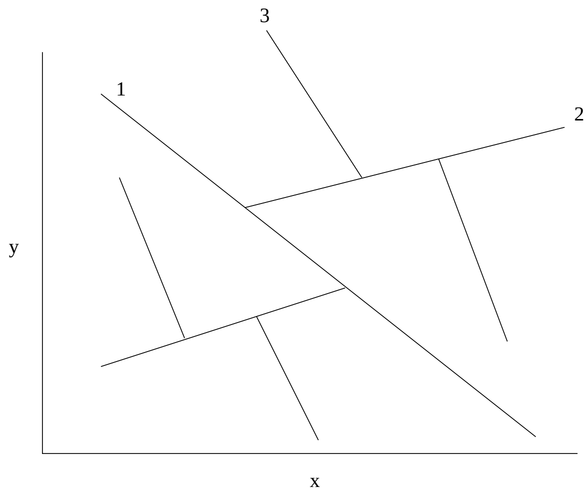


Figure 2. Decomposition after 3 time steps.

But with p processors and only 1 time step, the picture is very different; the half-spaces must be given all at once, as shown in Figure 3.

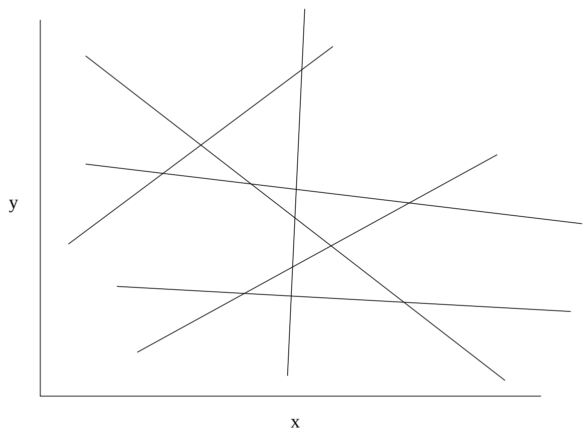


Figure 3. Decomposition by a parallel compuatation in one time step.

It is easy to see that p half-spaces in d dimensions can decompose $\mathbb{R}^d$ into at most $O(p^d)$ cells. Thus, if the dimension is small, this is much weaker than what can be done sequentially with p time steps. This allows us to prove in a straightforward manner the following Lemma:

Lemma 25.1 *A Linear PRAM computation with p processors and t time steps in d-dimensional space can partition the space into at most $O(p^{dt})$ cells.*

Proof We will proceed by induction on t. The base case was already argued. Consider a cell at time $t-1$. At time t, the p processors can specify up to p half-spaces. This can partition the cell into at most $O(p^d)$ sub-cells. Hence, using the induction hypothesis, after t time steps, we end up with at most $O(p^{d(t-1)}) \cdot O(p^d) = O(p^{dt})$ cells. ■

It simple to see that this Lemma implies the desired lower bound on p and t.

25.6.2 Algebraic PRAM

For the general algebraic case, a key ingredient will be a version of a theorem of Milnor and Thom:

Theorem 25.2 (Milnor-Thom) *Given p degree D polynomials $g_1, \ldots, g_p$ in d variables, the number of connected components of $\{v \in \mathbb{R}^d \mid g_1(v) \geq 0, \ldots, g_p(v) \geq 0\}$ is bounded by $(p \cdot D)^{O(d)}$.*

We proceed with the same inductive argument as before, but now at time t, Processor i can branch based on a degree 2^{t-1} inequality g_i. We need to bound the number of connected components in the partition induced by these inequalities. Note that this is *not* just the number of connected components of $\{v \in \mathbb{R}^d \mid g_1(v) \geq 0, \ldots, g_p(v) \geq 0\}$, which can be bounded directly by the Milnor-Thom Theorem. We have to find the number of connected components induced by *all* the possible branches at time t (for example when $g_i(v) < 0$).

Here, we must recall that our bounds are for the case that the inputs are specified by rational points of bounded precision. Hence, they must come from some sufficiently fine lattice. If none of these lattice points v are such that $g_i(v) = 0$ for some i, then the number of connected components *that contain a lattice point* induced by the inequalities can be bounded by applying the Milnor-Thom Theorem to $\{v \in \mathbb{R}^d \mid g_1^2(v) > 0, \ldots, g_p^2(v) > 0\}$. If however, some lattice point v is such that $g_i(v) = 0$, we can (for the sake of argument) replace g_i with $f_i = g_i - \epsilon$ and $h_i = g_i + \epsilon$, for some ϵ chosen small enough so that neither f_i nor h_i contains any lattice point, and so that all the lattice points v such that $g_i(v) > 0$ are now such that $f_i(v) > 0$, while all lattice points v such that $g_i(v) < 0$ are now such that $h_i(v) < 0$. (See Figures 4 and5 below.) Now we may apply the Milnor-Thom Theorem to the set $\{v \in \mathbb{R}^d \mid f_1^2(v) > 0, \ldots, f_p^2(v) > 0, h_1^2(v) > 0, \ldots, h_p^2(v) > 0, \}$, to arrive at a bound of at most $(4p \cdot 2^{t-1})^{O(d)}$ subcells created per cell at time t.

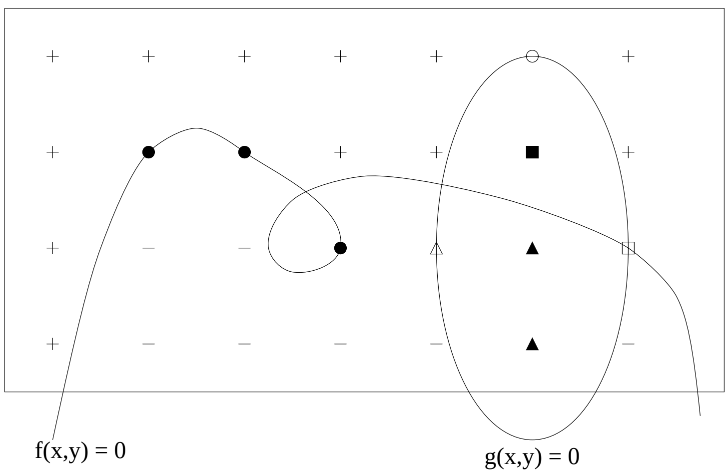


Figure 4. Two algebraic curves and lattice points.

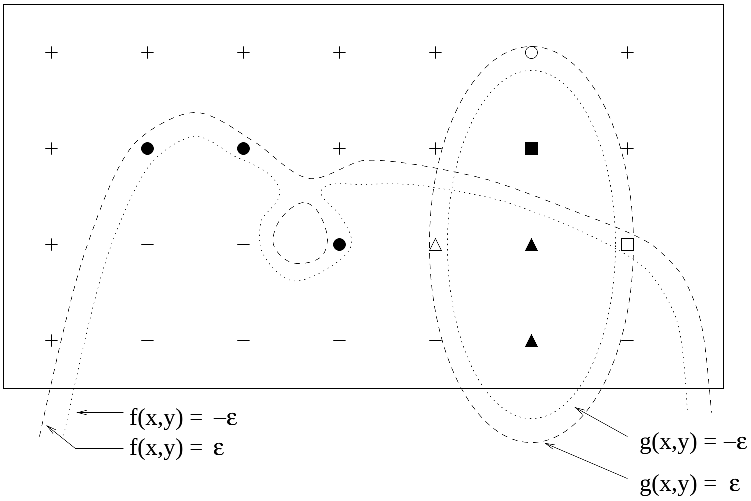


Figure 5. ϵ-shifting of two algebraic curves.

This shows that at most $O(p^d \cdot 2^{td})^t = O(p^{dt} \cdot 2^{t^2 d})$ cells can exist at time t in a general Algebraic PRAM computation.

Thus, for constant d, there exists some constant c such that if $p \leq 2^{c\sqrt{n}}$ and $t \leq c\sqrt{n}$, then at most 2^n cells can exist after t timesteps for an Algebraic PRAM with p processors. This gives the lower bound we were after.

25.7 Intuition for bounded precision

In this final section, we give some high-level intuition of how the proof proceeds to deal with the problem of bounded precision. We must essentially show that the set of rational points that can be generated in our bounded precision model is sufficiently dense to force many cells to contain a point from our set for any cell decomposition that could come from a feasible computation.

Recall that in our model, rational numbers were actually specified by a pair of bounded integers. We were also considering some problems (like the $\lfloor x \rfloor$ problem or the parametrized version of MAX-FLOW) which involved two rational numbers. We can consider these problems as problems that take three bounded integers w, y, z and use w/z as the first rational and y/z as the second. In general we are trying to lower bound the complexity of deciding a question like is $w/z \leq C(y/z)$ for some piecewise-linear function C. As shown in Figure 6 below, we can consider the possible rational points in our two variable problem to be the projections of some large cube of the 3-dimensional integer lattice onto the plane where $z = 1$. Mulmuley argues that if the bounds on w, y, and z are reasonably large (like $1 \leq w, y, z \leq 2^{n^2}$), then the density of the projection onto $z = 1$ is high enough to establish the result.

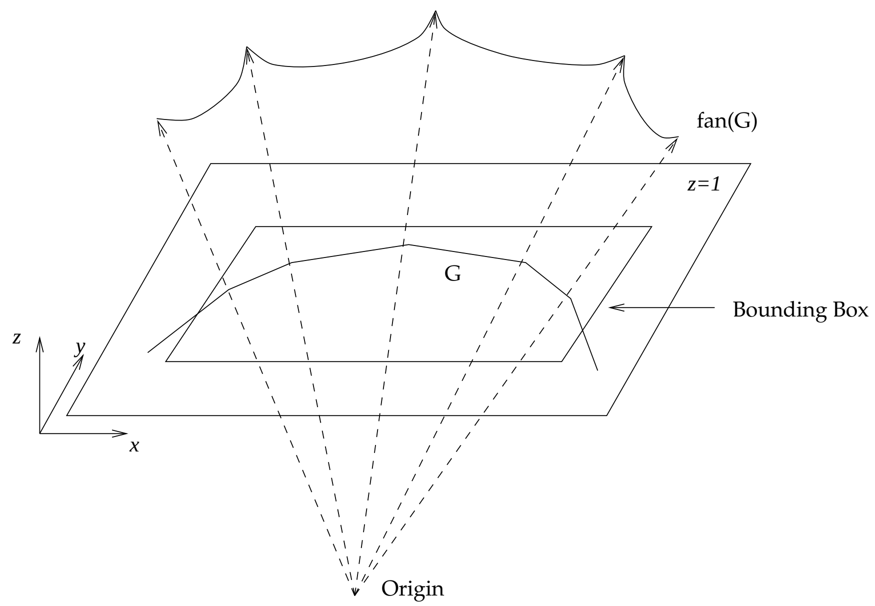


Figure 6. Projecting the integer lattice onto $z = 1$.

6.966 Algebra and Computation	December 7, 1998
Lecture 26	
Lecturer: Madhu Sudan	*Scribe: Prahladh Harsha*

In this lecture, we shall give a summary of the topics covered as part of this course, look at why we studied them and also mention some of the topics we did not cover in the course.

26.1 Factorisation

The first 11 lectures of the course were devoted to factorisation. We looked at factorisation of univariate polynomials over finite fields, integers; multivariate polynomial factorisation using Hensel's lifting, black-box factorisation etc.

The explicit motivation for studying factorisation was that it was a non-trivial algorithm achievement. Another motivation for studying factorisation is in the attempt to answer the question:

When is a function $f : F \to F$ (or more generally $f : F^n \to F$) , where F is a field, a degree d polynomial.

In the case of the univariate version of the above question, for the YES instances, the proof that f is a polynomial is the polynomial $p \in F[x]$ itself such that $p(x) = f(x) \forall x \in F$. For the NO instances of the problem, a simple observation tells us that there exist points $x_1, \ldots x_{d+2} \in F$ such that f when restricted to these points is also not a degree d polynomial. The obvious question that then arises in this case is how can one find such a set of $d+2$ points. There seems to be no "better" way than examining f on all of F. However, we can do better if the function f were to disagree with every degree d polynomial in $F[x]$ in atleast $\epsilon-$fraction of the points in F (and not atleast one point as before). In that case, a set of $d+2$ points can be found easily. In fact, a random set of $d+2$ points will suffice as a counterexample with probability ϵ.

In the multivariate version of the question (ie., $f : F^n \to F$) for the NO instances as before we have that if f does not agree with any polynomial of degree d and $|F| \geq 2(d+2)$, then there exist points $x_1, \ldots x_{d+2} \in F$ such that f when restricted to these points is also not a degree d polynomial. Here again, if we relax our NO instances to functions which disagree on a $\epsilon-$fraction of points with very degree d polynomial instead of disagreement on atleast one point, we can obtain a conterexample set easily by considering f restricted to a random line as follows: Pick $\hat{x}, \hat{y} \in F^n$ at random and look at $f|_{\hat{x}+t\hat{y}}$ and check whether this is a polynomial of degree d in the single variable t. As mentioned in the univariate case, if $f|_{\hat{x}+t\hat{y}}$ were not a polynomial, we can easily find a random set of points $t_1, \ldots t_{d+2}$ such that they serve as a counterexample. Thus the $d+2$ points $\hat{x}+t_1\hat{y} \ldots \hat{x}+t_{d+2}\hat{y}$ in F^n are proof that f is not a degree d polynomial.

Factorisation of bivariate polynomials can be used to find $\epsilon-$close polynomials to a given (univariate) function f in the following manner: Given a function $f : F \to F$ (where F is a finite field of size $|F|$), we can easily find a $Q \in F[x, y]$ of degree less than $\sqrt{|F|}$ such that $Q(x, f(x)) = 0 \forall x \in F$ and Q not identically 0. Suppose Q is of the form $Q(x, y) = \sum q_{ij} x^i y^j$, then we have $(\sqrt{|F|} + 1)^2$ unknown coefficients and $|F|$ constraints of the form $Q(x, f(x)) = 0$. This linear system can be solved to obtain the q_{ij}'s.

Claim 26.1 *If there exists a degree d polynomial $p \in F[x]$ such that $p(x) = f(x)$ for atleast $2d\sqrt{|F|}$ of the x's in F, then $y - p(x)$ is a factor of $Q(x, y)$. (where $Q(x, y)$ is the polynomial mentioned earlier)*

Proof Let $g(x) = Q(x, p(x))$. g has degree less then $2d\sqrt{|F|}$. But $g(x) = 0$ whenever $p(x) = f(x)$. Hence g has atleast $2d\sqrt{|F|}$ distinct roots. Therefore, $g \equiv 0$. Thus, $y - p(x)$ divides $Q(x, y)$ ■

Thus with the above claim and having obtained Q, we can factorise Q and search for factors of Q of the form $y - p(x)$. Comparing each such p against f, we can find whether f has any close polynomial and find it if there exists one,

A similar procedure can be adopted when we are given a bivariate function $f : F^2 \to F$. In such a case, we find a $Q \in F[x, y, z]$ of degree $|F|^{\frac{2}{3}}$ such that $Q(x, y, f(x, y)) = 0 \forall x, y \in F$, factorise Q and search for factors of the form $z - p(x, y)$.

Some of the topics we did not cover in this area are

- factorisation over $\mathbb{R}$
- factorisation over $\mathbb{C}$ [Neffs]
- factorisation over function fields and number fields [Cohen], [Shokrollahi, Wasserman]
- factorisation of sparse polynomials [Zippel] and interpolation[Borodin, Tiwari], [Ben-Or, Tiwari]

26.2 Quantifier Elimination

The second main topic covered in the course was Quantifier Elimination (lectures 12-18). Here we looked at the computational complexity of determining the truth value of statements like

$$Q_1 x_1 \dots Q_n x_n (f_1(x_1 \dots x_n) = \dots f_s(x_1 \dots x_n) = 0)$$

where $Q_i \in \{\exists, \forall\}$ and $f_i \in F[x_1, \dots x_n]$. We then looked at functions of the form $Q_1 x_1 \dots Q_n x_n \Phi(x_1, \dots x_n, y_1, \dots y_m)$ (Q_i's being quantifiers) versus functions of the form $\Phi(y_1, \dots y_m)$ (ie., definable functions vs computable functions) and showed how the former could be reduced to the latter under certain settings. A few of the topics we missed out in this part are

- Quantifier Elimination methods over $\mathbb{R}$
- Effective Hilbert's Nullstellensatz
- Bezout's Theorem.

26.3 Complexity Classes

In the third part of the course (lectures 19-25), we studied the complexity classes over different algebraic domains and looked at various models of algebraic computation .One of the interesting results not covered is:

$$\mathbf{P}^0_{\mathbb{R},+,-,\geq 0} = \mathbf{PAR}^{\mathbf{0}}_{\mathbb{R},+,-,\geq \mathbf{0}} \Longleftrightarrow \mathbf{P} = \mathbf{PAR}$$

where **PAR** stands for class of languages recognised by parallel polynomial time algorithms. This result is different in nature from the results presented in class like

$$\mathbf{P} = \mathbf{PSPACE} \Longrightarrow 0/1\mathbf{P}_{\mathbb{C}} = 0/1\mathbf{NP}_{\mathbb{C}}$$

in the sense that it does not restrict the inputs to be boolean for $\mathbf{P}^{\mathbf{0}}_{\mathbb{R},+,-,\geq \mathbf{0}}$ and $\mathbf{PAR}^{\mathbf{0}}_{\mathbb{R},+,-,\geq \mathbf{0}}$.

We also did not study the notion of p-computable and p-definable functions [Valiant] and find in what cases are p-definable functions also p-computable. A function f is a p-computable function if it is a polynomial (possibly multivariate) of degree $n^{O(1)}$ and computable in polynomial time. A function g is p-definable if $g(y_1, \dots y_n) = \sum_{x_1 \in \{0,1\}} \dots \sum_{x_m \in \{0,1\}} f(x_1, \dots x_m, y_1, \dots y_n)$ for some p-computable function f.

References

Lecture 1: Course overview. `http://theory.lcs.mit.edu/~madhu/algcomp.ps` (Lecture 1).

Lectures 2–4: Review of standard algebra.

- B. L. van der Waerden. **Algebra I.** Springer-Verlag, 1994.
- Rudolf Lidl and Harald Niederreiter. **Introduction to finite fields and their applications.** Cambridge University Press, 1994.

Lectures 5, 6: Factoring of univariate polynomials over finite fields.

- Donald Ervin Knuth. **The Art of Computer Programming: Seminumerical Algorithms (vol. 2, 3rd ed.).** Addison-Wesley Publishing Company, 1997.
- Richard Zippel. **Effective Polynomial Computation.** Kluwer Academic Press, 1993.
- Henri Cohen. **A Course in Computational Algebraic Number Theory.** Springer-Verlag, 1993.

Lectures 7–9: Factoring of multivariate polynomials.

- Erich Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469-489, May 1985.
- Erich Kaltofen. Effective Noether irreducibility forms and applications. *Journal of Computer and System Sciences*, 50(2):274-295, April 1995.

Lectures 10, 11: Factoring of rational polynomials.

- A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261: 513-534, 1982.
- Henri Cohen. **A Course in Computational Algebraic Number Theory.** Springer-Verlag, 1993.

Lectures 12–18: Quantifier elimination in algebraic sentences. A good overall reference is:

- David A. Cox, John B. Little, and Donal O'Shea. **Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra.** Springer-Verlag, 1996.

Lecture 14: Complexity of ideal membership.

- Ernst W. Mayr and Albert R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46: 305-329, 1982.

Lectures 16, 17: Complexity of quantifier elimination.

- Joos Heintz. Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 24(3):239-277, August 1983.

Lectures 20, 21: The Blum-Shub-Smale (BSS) model of algebraic computation.

- Lenore Blum, Felipe Cucker, Michael Shub and Steve Smale. **Complexity and Real Computation.** Springer-Verlag, 1997.

Lectures 22: Relating an NP-complete problem in the BSS model to traditional complexity.

- Pascal Koiran. Hilbert's nullstellensatz is in the polynomial hierarchy. *Journal of Complexity*, 12(4):273-286, December 1996.

Lectures 23–25: Lower bounding techniques in non-uniform algebraic computation.

- ALLAN BORODIN AND IAN MUNRO. **The computational complexity of algebraic and numeric problems.** American Elsevier Publishing Company, 1975.
- MICHAEL BEN-OR. Lower bounds for algebraic computation trees (preliminary report). *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 80-86, Boston, Massachusetts, 25-27 April 1983.
- ANDREW CHI-CHIH YAO. Decision tree complexity and Betti numbers. *Journal of Computer and System Sciences,* 55(1):36-43, August 1997.
- KETAN MULMULEY. Lower bounds in a parallel model without bit operations. To appear *SIAM Journal on Computing.* Available from `http://www.cs.uchicago.edu/~mulmuley`.

Lecture 26: Conclusion. `http://theory.lcs.mit.edu/~madhu/algcomp.ps` (Lecture 26).