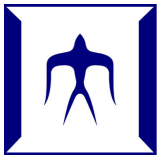
Lecture 4— Introduction to commutative algebra and Polynomial Calculus

Massimo Lauria — lauria.massimo@gmail.com
Office 1107, Ookayama West 8th Building

Friday — November 6th, 2015 (This document was updated on June 21, 2017)



We introduce polynomial calculus, a proof system that uses polynomial equations as proof lines. In order to discuss this proof system we introduce few basic concepts of commutative algebra. We also give a self contained introduction of the theory of ideals and on the computation of Gröbner basis

In the second and third lecture of the course we discussed the resolution proof system. A proof in resolution is a sequence of clauses, each inferred as a logical consequence of the previous one. In this lecture we discuss the proof system *Polynomial Calculus (*Pc*)*, which is very similar in spirit, but that uses a more expressive language than clauses: polynomial equations.[1]

To discuss this proof system we should refresh some preliminaries about commutative algebra. A good reference for this topic are the first two chapters of the book *Ideals, Varieties, and Algorithms*, by David Cox John Little Donal O'Shea.[2] Unfortunately we will not have time to cover such interesting topics, therefore I will keep at minimum the reference to these concepts so to make the lecture as self contained and possible, and so not to go overtime.

Let $\mathbb{F}$ a field[3] we will consider polynomials in the set $\mathbb{F}[x_1, x_2, \ldots, x_n]$,i.e. polynomials in n variables.

Each line in a proof is expressed as a polynomial equation as

$$x_1x_3 - 4x_1x_2x_3 = 0\,. \tag{1}$$

[1] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Gröebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, 1996

[2] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd edition.* Springer, 2007

[3] A field is an object from algebra. I It is essentially a set where the 0, 1 and the four operation $+, -, \times, \div$ behave as expected. We will mostly concern ourselves with fields like $\mathbb{Q}$, $\mathbb{R}$ and finite field like $\mathbb{F}_2$ or $\mathbb{F}_q$ with q a prime number.

Polynomial Calculus proof system

We are going to encode a CNF $\phi = \bigwedge_{i=1}^{m} C_i$ a sequence of polynomial equations

$$p_1(\vec{x}) = 0, p_2(\vec{x}) = 0, \ldots, p_m(\vec{x}) = 0$$

over the field $\mathbb{F}[x_1, x_2, \ldots, x_n]$. So that the *the satisfying assignments* of ϕ correspond exactly to the common roots of $p_1, \ldots, p_m$ inside the set of $\{0,1\}^n$.

Encoding of truth and falsehood In a polynomial calculus proof we interpret $\{0,1\}$ variable assignments as propositional variables

$x_i = 0$ when x_i is true;

$x_i = 1$ when x_i is false.

Notice that this is different from what it is usually done, but the reason for this interpretation is pretty clear. We interpret a polynomial as true when

the assignment of its variable is a root (a zero) of the polynomial. Therefore encoding the true assignment as the assignment to zero allows to say that the variables x_i is true when the polynomial x_i is true (i.e. when the equation $x_i = 0$ is true).

Encoding of clauses We encode a CNF as a set of polynomials, with the intended meaning that the satisfying assignment of a clause are in one-to-one correspondence with the roots of the corresponding polynomial that have all variables in $\{0,1\}$. We define the encoding by the means of the following example

$$x \vee \bar{y} \vee \bar{z} \vee u \qquad x(1-y)(1-z)u \tag{2}$$

where essentially we are using the fact that zero represents truth in order to use multiplication as a disjunction.

Exercise 1 (Efficiency of the encoding)**.** Show that there is a clause of k literals for which the polynomial encoding has one monomial, and another clause of k literals for which the polynomial encoding has 2^k monomials.

A polynomial calculus derivation over the field $\mathbb{F}$ from polynomials $p_1, \dots, p_m$ is a sequence of steps using the following rules.[4]

[4] The choice of the field is important and two polynomial calculus over two different fields are to be considered distinct proof systems.

Boolean axiom: $\dfrac{}{x_i^2 - x_i}$ for some $i \in [n]$.

Initial axiom: $\dfrac{}{p_j}$ for some $j \in [m]$;

Linear combination: $\dfrac{p \qquad q}{\alpha p + \beta q}$ for some $\alpha, \beta \in \mathbb{F}$;

Multiplication: $\dfrac{p}{x_i p}$ for some variable x_i with $i \in [n]$.

Notice the boolean axioms: they are only valid for $\{0,1\}$ assignments.

Definition 2 (Complexity measures of polynomial calculus)**.** *The* size *of a polynomial calculus proof is the number of monomials (with repetition) that occur in a proof, intented as the sum of the number of monomials in each polynomial among all polynomials in the proof.*[5] *The* degree *of a polynomial calculus proof is a the maximum degree among all polynomials in the proof.*[6]

[5] In the proof all polynomials are expressed as linear combinations of distinct monomials. The number of monomial in a polynomial $p = \sum_m \alpha_m m$ is the number of non zero coefficients α_m.

[6] The degree of a multivariate monomial $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is $\sum_i d_i$. The degree of a multivariate polynomial is the maximum degree among its monomials.

While we are mostly concerned about size, the degree is a very important measure of complexity, in a way similar to what resolution width is with respect to resolution length.

Definition 3 (Derivation notation)**.** *Consider $P = \{p_1, \dots, p_m\}$ and a polynomial q.*

- *We say that q is "logically implied" by P when $q(\vec{x}) = 0$ for every $x \in \{0,1\}^n$ which is a common root of $p_1, \dots, p_m$.*
- *We denote as $P \vdash q$ the fact that q has a* Pc *derivation from P.*

- *We denote as* $P \vdash_d q$ *the fact that* q *has a* Pc *derivation from* P *of degree at most* d.

We extend this notations to the form $\phi \vdash q$, *and to the form* $\phi \vdash_d q$, *where* ϕ *denotes the set of polynomials that encodes it.*

In the next exercise we show that the notation $P \vdash q$ can also be used to indicate logical implication, at least for the case of boolean reasoning.[7]

[7] Indeed this is not the case in other polynomial frameworks.

Exercise 4. Consider $P = \{p_1, \ldots, p_m\}$ and a polynomial q, all over n variables and of degree $\leq d$. Prove that P logically implies q if and only if $P \vdash q$, and furthermore in this case there is always a derivation of q from P with size at most $2^{O(n)}$ and degree $max\{n+1, d\}$.

Exercise 5. Show that any polynomial calculus derivation can be transformed, with a linear blow-up in size and constant blow up in degree, into a derivation where at most one variable is raised to a power greater than 1 (assuming this holds for the initial and the target polynomials).

We also introduce the notation of equivalence modulo P. This notation is justified since this is indeed a generalization of the concept of arithmetic modulo some integer number, in a very precise sense. See Cox et al.[8] for more information.

[8] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd edition.* Springer, 2007

Definition 6 (Equivalence modulo P)**.** *Consider* $P = \{p_1, \ldots, p_m\}$ *and a polynomial* q.

- *We say that* $q \equiv 0 \pmod{P}$, *or* $q \equiv_P 0$, *if* $P \vdash q$;
- *We say that* $q \equiv q' \pmod{P}$, *or* $q \equiv_P q'$, *if* $P \vdash q - q'$.

The constant polynomial 1 has no zeros, so it represents the contradiction. This also fits with the encoding of falsehood as 1 and with the encoding of the empty clause, which is indeed the polynomial 1 itself.

Definition 7. *A polynomial calculus refutation of a set of polynomials* $P = \{p_1, \ldots, p_m\}$ *is a derivation of the polynomial* 1. *A refutation of a CNF* ϕ *is a refutation of the set of polynomials that encodes* ϕ.

Multilinearization It is convenient, when dealing with polynomials over variables in $\{0,1\}$, to observe that monomials can have either value 0 or 1, and therefore that $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ is essentially the same as $\prod_{i:d_i>0} x_i$. This means that we can ignore exponents larger than 1.

Definition 8. *A polynomial* p *is called* multilinear *if in every monomial, every variable occurs with degree at most 1. Let* m *be the monomial* $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$, *then its multilinearization* $\tilde{m}$ *is the monomial* $\prod_{i:d_i>0} x_i$. *The multilinearization of a polynomial* $p = \sum_{\alpha_m} m$ *is* $\tilde{p} = \sum_{\alpha_m} \tilde{m}$.

Since we have the axioms $x_i^2 - x_i$ in our proof system, we can show explicitly the equivalence over $\{0,1\}$ of a polynomial and its multilinearization.

Exercise 9. Show that from p there is a polynomial calculus derivation of $\hat{p}$ which degree is at most the degree of p. Show that from $\tilde{p}$ there is a polynomial calculus derivation of p which degree is at most the degree of p.

Polynomial calculus with resolution

Exercise 10. Consider an unsatisfiable CNF formula ϕ. Show that if ϕ has a resolution refutation of width W, then ϕ has a Pc refutation of degree $W+1$ regardless the field $\mathbb{F}$.

If you try to solve the previous exercise, you would notice that if the original resolution proof uses clauses with many negative literals, then the simulation in polynomial calculus has large size. Since we want polynomial calculus to be an improvement over resolution, we want to avoid the size explosion that is required to simulate a clause with many negative literals.

The *Polynomial calculus with Resolution (*Pcr*)* is an extension of polynomial calculus defined in[9] which use additional variables to represent negations efficiently. Proof lines now are polynomials over both variables $x_1, \ldots, x_n$ and twin variables $\bar{x}_1, \ldots, \bar{x}_n$. The proof system now can use additional logical axioms to express the fact that the new variables are the negations of the old ones.

[9] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002

Negation axiom: $\overline{1 - x_i - \bar{x}_i}$ for some $i \in [n]$.

The reason to define Pcr is that now we can use a more efficient encoding of clauses.

$$x \vee \bar{y} \vee \bar{z} \vee u \qquad x\bar{y}\bar{z}u \tag{3}$$

from which we get an efficient simulation in term of size.

Exercise 11. Consider an unsatisfiable CNF formula ϕ. Show that if ϕ has a resolution refutation of width W and size S, then ϕ has a Pcr refutation of degree $W+1$ and size $O(S)$ regardless of the field $\mathbb{F}$.

Question (Question for the class)**.** *A* Pc *derivation is also a legal* Pcr *derivation. Nevertheless could it be that* Pcr *has proofs of smaller degree than* Pc *for some formulas?*

Definition 12. *We extend the notations from Definition 3 and Definition 6 to* Pcr *. This extension is non ambiguous since if P and q do not have twin variables, then the a derivation is possible in both proof systems, within the same degree (of course we are assuming the same underlying field).*

We see that Pcr simulates resolution. Now we argue that indeed there are cases in which polynomial calculus can be stronger than resolution. In the previous lecture 3 left as exercise to show that there are particular unsatisfiable systems of linear equation modulo 2 so that any resolution refutation of their CNF encoding requires exponential size refutation.

Proposition 13. *Let ϕ be CNF encoding of an unsatisfiable system of linear equations module 2. Show that ϕ has a* Pc *refutation of size polynomial in the size of ϕ, assuming the underlying field is $\mathbb{F}_2$.*

In the next class we will see, if time permits, that in other fields such formulas may require exponential size refutations (e.g., in $\mathbb{Q}$, $\mathbb{C}$, $\mathbb{R}$ and in $\mathbb{F}_q$ with $q \neq 2$).

Algorithm 1: The greedy division algorithm

Data: $(p_1, p_2, \ldots, p_m)$ and q
Result: $(h_1, h_2 \ldots, h_m)$ and a residue r such that

- $q = r + \sum_j h_j p_j$;
- $LT(p_j)$ does not divide any monomial in r;
- the degrees of all $h_j g_j$ and of r are less than the degree of q.

$r := 0$;
$h_j := 0$ for all $j \in [m]$;
while $q \neq 0$ **do**
  pick the first j such that $LT(p_j)$ divides $LT(q)$;
  $h_j := h_j + \frac{LT(q)}{LT(p_j)}$;
  $q := q - \frac{LT(q)}{LT(p_j)} \cdot p_j$;
  if *there is no such* j **then**
    $r := r + LT(q)$;
    $q := q - LT(q)$;
  end
end
return $(h_1, h_2 \ldots, h_m; r)$

Proof search

We want to understand Pc derivations and in particular derivation within a bounded degree. Since we only care about degree in this particular part of the lecture, we will only discuss Pc proofs and ingore PcR ones. Compare the next theorem with the one for resolution refutation of width d.

Theorem 14 (Main theorem)**.** *If $P \vdash_d q$ then some derivation of degree d can be produced in time $n^{O(d)}$.*

To operate on polynomial we need give a total order over the monomials, so to identify the leading term.

Definition 15 (Graded Lexicographic order)**.** *We order variables as $x_1 \prec x_2 \prec \cdots \prec x_n$. We order monomials so that $m_1 \prec m_2$*

- *the degree of m_1 is less than the degree of m_2;*
- *if the degree of m_1 and m_2 is the same, $m_1 \prec m_2$ if m_1 precedes m_2 in the lexicographic order.*

In a polynomial $p = \sum_m \alpha_m m$ the leading term (denoted as $LT(p)$) *is $\alpha_m m$ for m is the largest monomial in p according to $\prec$, such that $\alpha_m \neq 0$. If $\alpha_m m$ is the leading term, then m is called the leading monomial and is denoted as $LM(p)$.*

The first (incomplete) attempt is to find proofs in Pc is to to use the standard polynomial division Algorithm 1.

Algorithm 1 runs in polynomial time in the length of the initial polynomials. Its running produces a proof of $q - r$ in Pc (here we need to explicitly include the boolean axioms among the input polynomials P).

Exercise 16. Show that if the boolean axioms $x_i^2 - x_i$ for $i \in [n]$ are included in the set of dividends, then the residue is always multilinear (i.e. no variable occurs with degree larger than 1 in any monomial).

Unfortunately Algorithm 1 is not a complete test to check whether $P \vdash q$. If q divided by P gives zero, that is a proof that $P \vdash q$, but there are examples where q divided by P is not zero and still $P \vdash q$. Since testing whether $P \vdash 1$ is coNP-complete, it is expected that such an efficient algorithm would not work. Consider

$$x^2y + xy^2 + y^2 \text{ divided by } (xy - 1, y^2 - 1)$$
$$\text{that produces } (x + y)(xy - 1) + 1(y^2 - 1) + \underbrace{x + y + 1}_{r}$$

and now the same input where the divisors a ordered differently

$$x^2y + xy^2 + y^2 \text{ divided by } (y^2 - 1, xy - 1)$$
$$\text{that produces } (x + 1)(y^2 - 1) + x(xy - 1) + \underbrace{2x + 1}_{r}$$

The division algorithm fails to find that the smaller polynomial identity.

$$x - y = (2x + 1) - (x + y + 1)$$

is still equivalent to $x^2y + xy^2 + y^2$ modulo $(xy - 1, y^2 - 1)$. One way to generate further equivalences modulo P is to use S-polynomials.

Definition 17. *The S-polynomial of q_1 and q_2, denoted as $S(q_1, q_2)$ is the polynomial*

$$\frac{LCM(LM(q_1), LM(q_2))}{LT(q_1)} \cdot q_1 - \frac{LCM(LM(q_1), LM(q_2))}{LT(q_2)} \cdot q_2 \qquad (4)$$

Example 18. See that $S(xy - 1, y^2 - 1) = y(xy - 1) + x(y^2 - 1) = x - y$.

Exercise 19. Prove that:

- for every $\alpha \neq 0$ and $\beta \neq 0$, $S(q_1, q_2) = S(\alpha q_1, \beta q_2)$.
- for every monomials m_1, m_2 and polynomials q_1, q_2 with $LM(m_1 q_1) = LM(m_2 q_2)$, then

  $$S(m_1 q_1, m_2 q_2) = m S(q_1, q_2)$$

  for some monomial m.

The S-polynomials are useful to produce identities that are not reachable from the division algorithm. Furthermore the S-polynomials introduces some **higher degree reasoning** in our proof search arsenal. Indeed to prove $q_1, q_2 \vdash S(q_1, q_2)$ we may need to use intermediate polynomials

$$\frac{LCM(LM(q_1), LM(q_2))}{LT(q_1)} \cdot q_1 \text{ and } \frac{LCM(LM(q_1), LM(q_2))}{LT(q_2)} \cdot q_2 \qquad (5)$$

of degree larger than the degree of q_1, q_2 and $S(q_1, q_2)$.

Lemma 20 (Lemma 5 in Section 2.6 of Cox et al. (2007))**.** *Let $q = \sum_{i=1}^{\ell} c_i f_i$ where $LM(f_i) = m$ for every i and where $LM(q) \prec m$. Then we can write*

$$q = \sum_{i=1}^{\ell-1} c'_i S(f_i, f_{i+1}) \tag{6}$$

for some coefficients c'_i.

Proof. Write $f_i = d_i(m + f'_i)$ where $LM(f'_i) \prec m$. We get that

$$\begin{aligned} q = \sum_{i=1}^{\ell} c_i f_i \\ = \sum_{i=1}^{\ell} c_i d_i (m + f'_i) = \\ = c_1 d_1 (f'_1 - f'_2) + \\ + (c_1 d_1 + c_2 d_2)(f'_2 - f'_3) + \\ + (c_1 d_1 + c_2 d_2 + c_3 d_3)(f'_3 - f'_4) + \\ \ldots + (c_1 d_1 + \cdots + c_{\ell-1} d_{\ell-1})(f'_{\ell-1} - f'_\ell) + \\ + (\sum_{i=1}^{\ell} c_i d_i) f_\ell. \end{aligned} \tag{7}$$

From the fact that $LM(q) \prec m$ we get that $\sum_{i=1}^{\ell} c_i d_i = 0$ and since $f'_i - f'_{i+1} = S(f_i, f_{i+1})$, we are done. □

We now put together the division algorithm and the S-polynomials, and we define a strategy to search for Pc proofs. This essentially amount to the computation of Gröbner basis.

Definition 21 (Gröbner basis)**.** *A Gröbner basis for $P = \{p_1, p_2, \ldots, p_m\}$ is a set of polynomials $G = \{g_1, g_2, \ldots, g_\ell\}$ such that*

- *$P \vdash q$ if and only if $G \vdash q$;*
- *for every g_i, g_j in G, $S(g_i, g_j)$ divided by G is zero.*

Definition 22 (Gröbner pseudo-basis of degree d)**.** *A Gröbner pseudo-basis of degree d for $P = \{p_1, p_2, \ldots, p_m\}$ is a set of polynomials $G_d = \{g_1, g_2, \ldots, g_\ell\}$ such that*

- *$P \vdash_d q$ if and only if $G_d \vdash q$;*
- *for every g_i, g_j in G_d, either $S(g_i, g_j)$ divided by G_d is zero or $LCM(LM(g_i), LM(g_j))$ has degree larger than d.*

Before explaining how to compute such basis and pseudo-basis, we claim the most important property of these objects.

Theorem 23. *Let q be derivable with a degree d proof in polynomial calculus from $p_1, \ldots, p_m$. Let G_d output of Algorithm 2 from $p_1, \ldots, p_m$. Then q divided by G_d is equal to 0.*

Proof sketch. We say about a polynomial q with respect to P

- is reducible, if q divided by P gives zero;

- is semi-reducible, if $q = \sum_j h_j p_j$ where the $LT(h_j p_j) \preceq LT(q)$ for every j.
- has a degree d representation if $q = \sum_j h_j p_j$ where the degree of $h_j p_j$ is at most d for every j.

To prove the result we show that

1. if a polynomial is added in S at any point in the computation, then is semi-reducible w.r.t. the final set G_d;
2. when q has a degree d representation w.r.t. to G_d, then it is semi-reducible w.r.t. G_d;
3. when q of a degree at most d is semi-reducible w.r.t. G_d, then it is reducible with respect to G_d;
4. every polynomial in a degree d derivation in Pc is semi-reducible.

□

Corollary 24. *A Gröbner pseudo-basis of degree $n+1$ is an actual Gröbner basis.*

Corollary 25. *There is a* Pc *refutation of degree d from $p_1, \ldots, p_m$ if and only if $1 \in G_d$, where G_d is computed from by Algorithm 2 from $p_1, \ldots, p_m$.*

Algorithm 2: The Buchberger algorithm

```
Data: P = {p_1, p_2, ..., p_m}
Result: A pseudo Gröbner basis of degree d, G_d = (g_1, g_2 ..., g_ℓ)
Fix G_d := ∅;
Fix S := {x_i^2 − x_i | i = 1...n} ∪ {p_1, ..., p_m};
while S ≠ ∅ do
    pick some p in S (give precedence to the boolean axioms);
    let p' be the residue after dividing p by G_d;
    if p' ≠ 0 then
        G_d := G_d ∪ p' (append in the end);
        for g ∈ G_d with g ≠ p' do
            Add S(g, p') to S when the degree of LCM(LM(g), LM(p'))
            is at most d
        end
    end
end
return (h_1, h_2 ..., h_m; r)
```

Exercise 26. Show that every $g \in G_d$ has a Pc proof of degree d from P, and that it is possible to extract such proof from the the running of Algorithm 2.

Exercise 27. Show that Algorithm 2 computes G_d it time $n^{O(d)}$ assuming $P = \{p_1, \ldots, p_m\}$ has size $n^{O(1)}$, where n is the number of variables in P.

References

[ABSRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002.

[CEI96] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Gröebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, 1996.

[CLO07] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms : An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3rd edition.* Springer, 2007.