DD3501 Current Research in Proof Complexity Nov 28, 2011

Lecture 6

Lecturer: Jakob Nordström *Scribe: Björn Terelius*

In the first lecture of this course we gave three examples of concrete proof systems, namely resolution, polynomial calculus, and cutting planes. After that we have been focusing exclusively on resolution. Today, we want to switch gears and start talking about polynomial calculus (or PC for brevity) . We will also introduce a stronger variant of PC called polynomial calculus resolution (or PCR) which sometimes will be more convenient to work with. We will then follow roughly the same path as for resolution in lecture 2, i.e., define some proof complexity measures and state some basic results relating these measures.

But first, let us try to get a birds-view perspective of what we have been doing up to this point.

1 Summary of the Lectures So Far

We introduced the general definition of a propositional proof system for tautologies in the first lecture, but then immediately decided to focus on the slightly more restricted concept of sequential proof systems for unsatisfiable CNF formulas. In such a proof system, a refutation of a formula F (sometimes also referred to as a proof of F) is a sequence of lines written on a board, where at each step we can apply one of the following derivation rules:

1. Download of an axiom (a clause of F) to the board.

2. Application of an inference rule of the proof system to lines already on the board.

3. Erasure of a line from the board.

A concrete proof system is determined by the format of lines and the allowed inference rules. So far, we focused on the resolution proof system, where each line consists of a disjunction of literals and the only rule for deriving new clauses is the resolution rule

$$\frac{C_1 \lor x \qquad C_2 \lor \overline{x}}{C_1 \lor C_2} \tag{1.1}$$

where C_1 and C_2 denote any clauses, and x is the variable resolved over. A resolution refutation of a formula F, sometimes also referred to as a proof of F, is a derivation of the empty clause $\bot$ without any literals from F. We have studied the following measures on resolution proofs.

Length: the number of lines in a proof.

Space: the maximum number of clauses simultaneously on the board at any point during a proof (to be precise, this is the *clause space* measure).

Width: the maximum number of literals appearing in any clause in a proof.

The measures we were mainly interested in were length and space, but it turned out that width was a useful concept when trying to understand and prove properties about length and space.

Let us summarize what we known and do not know at this point about the relationships between these three measures.

1.1 Length Versus Width

In lecture 2, we learned that a strong upper bound on length implies a fairly strong upper bound on width [BW01]. Furthermore, as shown in lecture 5 this result is essentially tight [BG01]. By a simple counting argument, we also know that a strong upper bound on width implies a strong upper bound on length (since there are only so many distinct clauses in limited width, an every clause need appear only once in a resolution refutation if we are focused on minimizing length and do not care at all about space). One question we raised but did not answer was the following.

Question 1. *Are there trade-offs between length and width?*

1.2 Space Versus Width

In lecture 4 we proved that a strong upper bound on space implies a strong upper bound on width [AD08]. What about the converse?

Question 2. *Does a strong upper bound on width imply a strong upper bound on space?*

As to the question whether there are trade-offs between space and width, we have learnt that the answer is yes. We started to prove this in lecture 4 but did not have time to finish. However, in the scribe notes for "lecture 4$^1/_2$" (posted on the course webpage), there is a full proof of the fact that there are formulas which are maximally easy with respect to both width and clause space, but where optimizing one of the measures leads to essentially worst-case behaviour for the other measure.

1.3 Space Versus Length

In lecture 4 we also observed that a a strong upper bound on clause space implies a strong upper bound on length. This follows from the upper bound on width in terms of clause space in [AD08] plus a simple counting argument. It is natural to ask whether there is a similar implication in the other direction.

Question 3. *Does a strong upper bound on length imply a strong upper bound on space?*

Finally, one of the questions that we will be particularly interested in, but did not get to so far during the course, it whether length and space can be optimized simultaneously or whether minimizing one of the measures must cause a significant increase in the other measure.

Question 4. *Are there trade-offs between length and space?*

So what is the status of these questions? Question 1 is still open. Questions 2, 3, and 4 were open for a while, but were solved a couple of years back. To keep the suspense, however, we will have to wait until after Christmas to learn the answers.

2 Polynomial Calculus

We now turn our attention to a proof system called polynomial calculus (PC) that was introduced by Clegg, Edmonds and Impagliazzo [CEI96]. In this proof system, the lines are polynomial equations and the inference rules are addition and multiplication, with the calculations done over some fixed field $\mathbb{F}$. For applications, the coefficient field is usually $\mathbb{F}_2$ (also known as GF(2)), the field consisting only of 0 and 1. In the theoretical definition, any finite or even infinite field can be used, but which field we choose can make a difference for which results hold. In particular, the *characteristic* of the field is important.

Definition 2.1. The *characteristic* of a field is the smallest positive number p such that $p \cdot 1 = 0$, or 0 if no such number exists.

Some theorems in the literature hold only for fields of certain characteristics, or for all fields except those of certain characteristics. For instance, when proving lower bounds the hardest case seems to be $\mathbb{F}_2$. All this will not make too much difference for us, however, since in this course we will focus on results that hold regardless of characteristic.

To work in the polynomial calculus proof system, we need to translate CNF formulas F to the admissible format in this proof systems, i.e., to polynomial equations. As we shall soon see, in the context of polynomial calculus it is more natural to think of 0 as true and 1 as false, which is the opposite of the convention we have for resolution and for CNF formulas. So the reader should be warned that some care will be needed when switching between these two contexts. Thus, in all of our lectures on polynomial calculus, true will be translated to 0 and false will be translated to 1. The translation of CNF clauses is best illustrated by an example.

Example 2.2. The clause $x \vee \overline{y} \vee z$ is translated to the equation $x(1-y)z = 0$. To get a unique representation of the polynomials, we require them to be written in expanded form, i.e., as a sum of monomials. In this case, we get $xz - xyz = 0$. Clearly, any truth value assignment satisfies the clause iff the corresponding 0/1-assignment is a root to the polynomial.

In general, each clause C in a formula F is translated into a product where each factor is either a variable or one minus a variable depending on whether the corresponding literal was positive or negative, respectively. This gives us a system of polynomial equations, and we want to know if they have any common root where all variables are assigned either 0 or 1. To force 0/1-solutions, we also add equations $x^2 - x = 0$ for every variable x.

If we have derived two equations $p = 0$ and $q = 0$, then clearly $\alpha p + \beta q = 0$ must also hold for any $\alpha, \beta \in \mathbb{F}$. Similarly, it is clear that $xp = 0$ for any variable x. This motivates the following collection of derivation rules for PC, where $\alpha, \beta \in \mathbb{F}$, $p, q \in \mathbb{F}[x, y, z, \ldots]$, and x is any variable:

Linear combination $\dfrac{p = 0 \qquad q = 0}{\alpha p + \beta q = 0}$

Multiplication $\dfrac{p = 0}{xp = 0}$ for any variable x

Boolean axioms $\dfrac{}{x^2 - x = 0}$ for all variables x (forcing 0/1-solutions)

Note that all polynomials occurring in the proof can be made multilinear by use of the Boolean axioms, so without loss of generality we can assume that no variable occurs with higher exponent than 1. (In fact, if we wanted to, we could define the multiplication rule to always result in a multilinear polynomial equivalent to xp. Although we will not do so here, we will be fairly liberal in assuming that all polynomials occurring in a PC-derivation are multilinear without going into too much detail about how this happens.)

To refute a formula F in the PC proof system, we derive the equation $1 = 0$ from the polynomials corresponding to the clauses of F.

Theorem 2.3. *Polynomial calculus is sound and implicationally complete.*

Proof sketch. Soundness is easy. Common zeroes to the polynomial equations are preserved by our inference rules. If we manage to derive the polynomial 1, then clearly there cannot exist any common zero to the polynomial equations.

Completeness is harder, but can be demonstrated either algebraically by Hilbert's Nullstellensatz or via a simulation argument (by having PC mimic resolution proofs without worrying about the efficiency of the simulation). We did not at all go into any details here but just decided to accept that this is a well-known fact. (However, see the write-up in Section 2.2 added during preparation of the scribe notes). □

There are solvers based on Gröbner bases that search for PC-proofs, for instance PolyBoRi [BD09], but although this seemed to be a very promising direction at the time of the paper [CEI96], it is probably fair to say that SAT solvers based on algebra have so far failed to deliver on the hopes raised then. Current state-of-the-art solvers based on resolution—so-called DPLL solvers with clause learning, also known (perhaps more correctly) as conflict-driven clause learning solvers or CDCL solvers for short—seem to be an order of magnitude faster on most instances occuring in practice (although the paper [BDG$^+$09] presents data claiming that PolyBoRi can be faster on certain industrial instances).

Let us next see how our generic proof complexity measures for sequential proof systems get instantiated for polynomial calculus.

Definition 2.4 (Proof size). The *size* $S(\pi)$ of a polynomial calculus proof π is the total number of monomials in all the lines in π (counted with repetitions). The *size* of refuting an unsatisfiable CNF formula F in polynomial calculus, denoted $S_{\mathcal{PC}}(F \vdash \bot)$, is the minimal size of any PC-refutation of F.

Note that if we wanted to be precise, we should also count the number of variables in each monomial, but the size measure as defined in Definition 2.4 clearly is within a linear factor of this and is much cleaner to work with (assuming that the field $\mathbb{F}$ is constant so that we do not need to worry about issues regarding representation of the coefficients).

Definition 2.5 (Proof length). The *length* $L(\pi)$ of a PC-proof π is the number of derivation steps in π. The *length* of refuting F in PC, denoted $L_{\mathcal{PC}}(F \vdash \bot)$, is the minimal length of any PC-refutation of F.

This definition is the same as for resolution. Note, however, that while for resolution size and length are essentially the same measure[1] (to within a linear factor), this is not necessarily the case for PC since a single line can have exponential size.

Definition 2.6 (Proof space). The *(monomial) space* $Sp(\pi)$ of a polynomial calculus proof $\pi = \{\mathbb{P}_0 = \emptyset, \mathbb{P}_1, \ldots, \mathbb{P}_\tau\}$ is the maximal number of monomials (counted with repetitions) in any configuration $\mathbb{P}_t \in \pi$. The *total space* $TotSp(\pi)$ is the maximal total number of symbols (counted with repetitions) in any configuration.[2] The *(monomial) space* and *total space* of refuting F, denoted $Sp_{\mathcal{PC}}(F \vdash \bot)$ and $TotSp_{\mathcal{PC}}(F \vdash \bot)$, respectively, is the minimal monomial space or total space of any any PC-refutation of F.

The measure of monomial space is the polynomial calculus analogue of clause space in resolution. In this course, we will mainly be interested in size and (monomial) space.

2.1 An Alternative Algebraic Interpretation

Derivations in the PC proof system can also be interpreted naturally in terms of ideals in polynomial rings.

Definition 2.7 (Ideal). Let $\mathcal{R}$ be a (commutative) ring. An *ideal* I of $\mathcal{R}$ is a subset such that

1. if $p, q \in I$ then $p + q \in I$
2. if $p \in I$ then $rp \in I$ for any $r \in \mathcal{R}$

[1]Indeed, in much of the proof complexity literature, what we refer to in this course as the "length" of a resolution proof is called the "size" of the proof, which is not unreasonable given the convention we just adopted ourselves in Definition 2.4. However, since we will sometimes be interested in measuring both size and length, and in contrasting the two measures, we want to have a slightly more precise terminology that does not mix them up.

[2]If the field $\mathbb{F}$ is finite, which is the case we are mostly interested in, we can just charge one unit of memory for every coefficient, but in infinite fields one should probably also take into consideration the size of the coefficients.

That is, an ideal is a subring that is not only closed under addition and multiplication within the subring, but is also closed under multiplication by any element of the ambient ring.

Instead of translating CNF clauses to polynomial *equations* $p(x, y, \ldots) = 0$, we can consider them as just the *polynomials* $p(x, y, \ldots)$ in the polynomial ring $\mathbb{F}[x, y, z, \ldots]$. This has the notational advantage that we can drop all the "$= 0$" from the inference rules and proof lines. Looking at the ideal I generated by those polynomials, it is easy to show that the polynomials derivable by polynomial calculus are precisely the polynomials in the ideal I (that is simply the way PC is defined). Refuting a CNF formula F in polynomial calculus amounts to showing that the ideal I contains 1, and hence that the ideal is the entire ring.

During the rest of this course, we will usually think of polynomial calculus as making derivations in the ideal generated by the clauses of a CNF formula F, rather than as deriving polynomial equations.

2.2 Implicational Completeness of PC (Detour Courtesy of Scribe)

Let us now do something that was not at all covered in class, namely sketch how the Nullstellensatz can be used to prove that PC is complete. Before stating the Nullstellensatz, we need the concept of algebraic closure.

Definition 2.8 (Algebraic closure). The *algebraic closure* $\overline{\mathbb{F}}$ of a field $\mathbb{F}$ is the smallest field containing $\mathbb{F}$ such that every non-constant polynomial in $\overline{\mathbb{F}}[x, y, z, \ldots]$ has a root in $\overline{\mathbb{F}}$.

For example, the complex numbers $\mathbb{C}$ is the algebraic closure of the real numbers $\mathbb{R}$. Somewhat similar constructions can be performed to obtain an algebraic closure of any field $\mathbb{F}$.

Theorem 2.9 ((Weak) Nullstellensatz). *Let $\overline{\mathbb{F}}$ be an algebraically closed field. An ideal I in a polynomial ring $\overline{\mathbb{F}}[x, y, z, \ldots]$ contains 1 if and only if the polynomials in I have no common zero in $\overline{\mathbb{F}}$.*

Let us use the Nullstellensatz to argue that polynomial calculus is complete.

Proof sketch for Theorem 2.3. To prove that polynomial calculus is complete, we need to show that if the axioms have no common zero in $\mathbb{F}$, then there are polynomials $g_1, \ldots, g_k \in \mathbb{F}[x, y, z, \ldots]$ such that $1 = \sum_{i=1}^{k} g_i f_i$. The Nullstellensatz tells us that if the axioms have no common zero in $\overline{\mathbb{F}}$, then there are polynomials $g_1, \ldots, g_k \in \overline{\mathbb{F}}[x, y, z, \ldots]$ such that $1 = \sum_{i=1}^{k} g_i f_i$. It remains for us to show that the algebraic closure does not matter in this case. First, because of the Boolean axioms, any common zero will assign 0 or 1 to every variable. Hence, there are zeroes in $\overline{\mathbb{F}}$ if and only if there are zeroes in $\mathbb{F}$.

It is not as easy to show that polynomials g_i with coefficients in $\mathbb{F}$ can be found. From the Nullstellensatz we know that there are polynomials g_i with coefficients in $\overline{\mathbb{F}}$. Observe that $\overline{\mathbb{F}}$ is a vector space over $\mathbb{F}$, so there is some (possibly infinite) basis for $\overline{\mathbb{F}}$ over $\mathbb{F}$. Let $\{1, \theta_1, \ldots, \theta_m\}$ be a (finite) subset that spans all the coefficients of the polynomials g_i and note that $\sum_{i=1}^{k} g_i f_i$ is a linear expression in $\{1, \theta_1, \ldots, \theta_m\}$ with coefficients in $\mathbb{F}[x, y, z, \ldots]$. Since 1 is linearly independent of $\theta_1, \ldots, \theta_m$ and $\sum_{i=1}^{k} g_i f_i = 1$, the θ_j parts cannot contribute to the sum. Thus we can remove any θ_j parts from the coefficients of $g_1, \ldots g_k$ without changing the sum. This gives polynomials in $\mathbb{F}[x, y, z, \ldots]$, as desired. Thus, the Nullstellensatz implies that polynomial calculus is complete, i.e., that we can derive 1 if the formula F in unsatisfiable. $\square$

Sometimes it is useful to have the stronger notion of *implicational completeness*. Implicational completeness in general means that if a set of Boolean functions S semantically implies another function h (in whatever representation the proof system uses), then we can derive h from S in the proof system. For the case of polynomial calculus, it means that if a polynomial h vanishes on all common roots of the polynomials $f_1, \ldots, f_k$, then we can find polynomials $g_1, \ldots, g_k$ such that $h = \sum_{i=1}^{k} g_i f_i$. The fact that PC is implicationally complete appears to be more or less part of folklore, but a formal statement and proof can be found, for instance, in Buss et al. [BIK+97, Theorem 5.2]. The paper by Buss et al. also contains a constructive proof for a version of the Nullstellensatz that uses the Boolean axioms to avoid the algebraic closures. We will now prove the implicational completeness of PC using essentially the same argument as in [BIK+97], except that we use Theorem 2.3 instead of their version of the Nullstellensatz.

Theorem 2.10 (Implicational completeness). *Given a set of polynomials $\{f_1, \ldots, f_k\}$ in $\mathbb{F}[x, y, z, \ldots]$ plus the Boolean axioms $x_i^2 - x_i$ for all variables x_i involved in the polynomials, and given a polynomial $h \in \mathbb{F}[x, y, z, \ldots]$ which is zero for all common roots to $\{f_1, \ldots, f_k\}$, there exists polynomials $g_1, \ldots, g_k \in \mathbb{F}[x, y, z, \ldots]$ such that*

$$h = \sum_{i=1}^{k} g_i f_i \ .$$

Proof. Let $\mathbb{F}$ be a finite field with $q = p^n$ elements. By the assumption of the theorem, h is zero whenever the polynomials $f_1, \ldots, f_k$ are zero, so $f_1, \ldots, f_k, 1 - h^{q-1}$ have no common roots. From the proof of Theorem 2.3, we get polynomials $g_1, \ldots, g_{k+1} \in \mathbb{F}[x, y, z, \ldots]$ such that

$$1 = \sum_{i=1}^{k} g_i f_i + g_{k+1}(1 - h^{q-1}) \ . \tag{2.1}$$

Multiplying both sides by h gives

$$h = \sum_{i=1}^{k} (g_i h) f_i + g_{k+1}(h - h^q) \ . \tag{2.2}$$

Recall that in a field (or ring) of characteristic p, it holds that $(a + b)^p = a^p + b^p$ and similarly for sums with more terms. Thus, raising the polynomial h to the power q is the same as raising each term in h to the power q. Look at such a term $c \prod x_i$. We see that $(c \prod x_i)^q = c^q \prod x_i^q = c \prod x_i$ because of the Boolean axioms $x_i^2 = x_i$ and the fact that $c^q = c$ for all $c \in \mathbb{F}$. Hence $h - h^q = 0$ (modulo the Boolean axioms) and we get polynomials $g_1, \ldots, g_k \in \mathbb{F}[x, y, z, \ldots]$ such that $h = \sum_{i=1}^{k} g_i f_i$. □

3 Polynomial Calculus Resolution

There is one annoying problem with how we encode formulas as polynomials. Consider the clause $\overline{x}_1 \vee \overline{x}_2 \vee \ldots \vee \overline{x}_w$. This is translated to the polynomial

$$\prod_{i=1}^{w} (1 - x_i) = \sum_{S \subseteq [w]} (-1)^{|S|} \prod_{i \in S} x_i \tag{3.1}$$

which has size exponential in the clause width w. Thus, if a formula F has a clause with a linear number of negative literals, then just downloading that clause gives an exponential lower bound on size and space. Usually, exponential lower bounds would get us very excited, but somehow it is clear that this particular type of exponential lower bound is *not* what we are looking for...

Let us discuss two ways to “fix” this problem.

1. We can choose to consider only k-CNF formulas for some constant k. We know that any CNF formula F can be converted to an equivalent 3-CNF formula $\widetilde{F}$ using extension variables, and for formulas of bounded width the blow-up in (3.1) is not an issue.

2. We can change the encoding of clauses as polynomials by introducing new variables corresponding to negated literals and additional axioms making sure that variables corresponding to positive and negative literals get opposite values in $\{0, 1\}$ as will be described shortly. This second "fix" was introduced by Alekhnovich, Ben-Sasson, Razborov, and Wigderson in the paper [ABRW00] (journal version in [ABRW02]) and results in a stronger proof system which we will refer to as *polynomial calculus resolution* or usually just PCR for brevity.

Which fix is "the right one"? Well, both are interesting and well-motivated, but perhaps in slightly different ways.

In general, we will always prefer to get space bounds that hold also for k-CNF formulas. Since any formula can be converted to a k-CNF, it seems desirable to have results that do not break completely just because we make such a transformation. On the other hand, sometimes it is just too hard to prove bounds for k-CNF formulas, and in this case results for formulas of unbounded width can also be very interesting. (For instance, as we will learn in a later lecture, for a long time we only had nontrivial space lower bounds in polynomial calculus for formulas of unbounded width.)

As to the issue of PC versus PCR, some Gröbner basis SAT solvers (for instance, [BD09]) use a representation where the issue of exponential blow-up of negative literals does not arise when encoding the CNF formulas, whereas it seems to be a more serious problem for other solvers. So the case could be made for studying both PC and PCR. When we prove lower bounds, however, of course it is preferable to prove such bounds against as strong an adversary as possible, and therefore PCR would seem to be the proof system of choice in such cases.

The translation of clauses in PCR is done by introducing new variables $\overline{x}, \overline{y}, \overline{z}, \ldots$, and using the new variable $\overline{x}$ instead of the factor $1 - x$ to encode negation. Let us again explain this by an example.

Example 3.1. In PCR, the disjunctive clause $x \vee \overline{y} \vee z$ is encoded as the equation $x\overline{y}z = 0$, or the polynomial $x\overline{y}z$ if we use the algebraic formulation (as we will tend to do below).

Now we need to enforce that $\overline{x}$ is the negation of x, i.e., that $\overline{x} = 0$ if $x = 1$ and vice versa. We do this in the obvious way by requiring that $\overline{x} = 1 - x$, i.e., by adding the following axiom.

Complementarity axioms $\dfrac{}{x + \overline{x} - 1}$ for all variables x

For simplicity and symmetry, we also add the Boolean axioms for negated variables.

Boolean axioms $\dfrac{}{\overline{x}^2 - \overline{x}}$ for all variables $\overline{x}$

All other derivation rules and proof complexity measures are as for polynomial calculus. Note that for PCR, it is even clearer that monomial space is the analogue of clause space in resolution, since every clause gets translated to one monomial.

4 Some Basic Facts About PC and PCR

It is not hard to see that PCR can simulate PC refutations in essentially the same size and space. PCR can do everything PC can do, except downloading axiom clauses in the PC format. A minor technical issue that can arise here is that if there is an axiom clause with a linear number of negative literals, PC can download the exponential-size representation of this axiom in one step, whereas in a naive simulation by PCR we will have to derive the inefficient representation in an exponential number of steps from the efficient PCR-representation with negated literals as variables, using the complementarity axioms $x + \overline{x} - 1$ repeatedly. Each application of the complementarity axioms increases the number of monomials by 1, so we need to use the rule at most 2^s times if there are s negative literals in the clause. Since each of those lines have size at most 2^s, the size of the total proof is not much worse than 2^{2s}. This is only polynomially worse than downloading the PC axiom directly, since the axiom has size 2^s. (And also, if we really wanted to, it seems that there would be smarter ways of having PCR simulate PC than immediately expanding out a single monomial to an exponential number of monomials, but since we do not really need it we will not waste any time on thinking about it.)

From the fact that PCR can simulate PC proofs, we conclude that upper bounds on size and space for PC hold for PCR too. Similarly, lower bounds on size and space for PCR hold for PC too.

Furthermore, it is not hard to show that once we add separate variables for negated literals, PCR can simulate resolution in a line-by-line fashion (which is the reason that it is called polynomial calculus *resolution*).

Proposition 4.1. *PCR polynomially simulates resolution in essentially the same length, size and space.*

Proof. Left as an exercise. □

From Proposition 4.1 it follows immediately that the worst-case behaviour of PCR cannot be worse than that of resolution.

Corollary 4.2. *Let F be an unsatisfiable CNF formula and let $S(F)$ denote the size of F, i.e., the number of literals in F counted with repetitions. Then*

$$S_{\mathcal{PCR}}(F \vdash \bot) = \exp(\mathrm{O}(S(F)))$$
$$Sp_{\mathcal{PCR}}(F \vdash \bot) = \mathrm{O}(S(F))$$

and there are PCR-refutations attaining both of these bounds simultaneously.

Remark 4.3. The first part of the corollary holds also for polynomial calculus, i.e., it is true that $S_{\mathcal{PC}}(F \vdash \bot) = \exp(\mathrm{O}(S(F)))$. This is because the PCR-simulation of resolution in size $\exp(\mathrm{O}(S(F)))$ essentially has all polynomials being single monomials (corresponding to the clauses in the simulated resolution refutation), and such a PCR-refutation can in turn be mimicked by a PC-refutation where each monomial grows by at most a factor $\exp(\mathrm{O}(S(F))$. This gives a PC refutation of size $\bigl(\exp(\mathrm{O}(S(F)))\bigr)^2$ which is still at most $\exp(\mathrm{O}(S(F))$.

On the other hand, it does *not* hold that $Sp_{\mathcal{PC}}(F \vdash \bot) = \mathrm{O}(S(F))$. For a counterexample, consider the family of formulas

$$\begin{aligned} F_n = \quad & (\overline{x}_1 \lor \overline{x}_2 \lor \cdots \lor \overline{x}_n) \\ & \land x_1 \\ & \land x_2 \\ & \vdots \\ & \land x_n \end{aligned} \tag{4.1}$$

which has small resolution and PCR refutations, but for which any PC-refutation must at some point download the first clause, which requires exponential space.

The following result seems to be almost folklore and we probably will not cover it in class. It is just stated here for reference.

Theorem 4.4. *PCR is exponentially stronger than resolution with respect to proof size.*

A natural question is whether an analogue of Theorem 4.4 holds also with respect to space. One would expect the answer to be yes, but this is not known.

Open Problem 5. *Is PCR asymptotically stronger than resolution with respect to space, i.e., when comparing monomial space to clause space?*

Another question that arises quite naturally at this point is how much stronger PCR is than PC. Well, that is a good question, but we will not discuss it in any greater detail. Clearly, the formula in (4.1) provides an exponential separation between PCR and PC, but this example seems artificial since it only works because of the first clause of unbounded width. For k-CNF formulas we know of no such simple example. And it so happens that the lower bound techniques that we will use work for both PC and PCR simultaneously, so we will not get any separations between the two systems in that way. Also, although it is not a priori obvious, it turns out that for k-CNF formulas PC has the same worst-case behaviour as resolution.

Lemma 4.5 ([FLN+11]). *If F is an unsatisfiable k-CNF formula (for some universal constant k), then there is a PC-refutation $\pi : F \vdash \bot$ such that*

$$S(\pi) = \exp(\mathrm{O}(S(F)))$$
$$Sp(\pi) = \mathrm{O}(S(F))$$

where the constants hidden in the asymptotic notation depend on k.

Proof. Left as a (somewhat non-trivial) exercise. □

As already stated, we will mainly be interested in the size and (monomial) space of PC- and PCR-refutations. As we will soon see, however, when we want to prove size lower bounds it is very helpful to study the auxiliary measure of *degree*. This is similar to how width turned out to be a useful measure in resolution, and indeed, it is not hard to see an analogue here since clauses of width k in resolution correspond to monomials of degree k in PCR.

It was shown in [CEI96] that if there is a PC-refutation in degree d, then it is possible to find a PC-refutation in time $n^{\mathrm{O}(d)}$. We want to show that there is a kind of converse to this in that if the minimal degree of any refutation is large, then there cannot exist small-size refutations. But let us first give a formal definition of the degree measure.

Definition 4.6. The *degree* $Deg(\pi)$ of a PC- or PCR-derivation is the largest total degree of any monomial appearing in π, i.e., (by multilinearity) the largest number of variables that appears in any monomial. The *degree* of refuting F in PC or PCR, denoted $Deg_{\mathcal{PC}}(F \vdash \bot)$ and $Deg_{\mathcal{PCR}}(F \vdash \bot)$ respectively, is the minimal degree of any refutation of F in PC or PCR respectively.

In fact, without loss of generality we can drop the subscript from the degree measure whenever we like and just talk about the degree $Deg(F \vdash \bot)$ of refuting F.

Proposition 4.7. $Deg_{\mathcal{PC}}(F \vdash \bot) = Deg_{\mathcal{PCR}}(F \vdash \bot)$ *for any unsatisfiable CNF formula F.*

Proof. Left as a fairly straightforward exercise. □

Now we can state the relation between degree and size that we want to prove. We state the theorem for PCR below, but exactly the same result holds also for PC. (Indeed, it was originally proven for PC by Impagliazzo, Pudlák, and Sgall [IPS99], but the proof for PCR is the same line by line.)

Theorem 4.8 ([IPS99]). *Let F be an unsatisfiable formula over n variables. Then*

$$Deg_{\mathcal{PCR}}(F \vdash \bot) \leq W(F) + \mathrm{O}\left(\sqrt{n \ln S_{\mathcal{PCR}}(F \vdash \bot)}\right) .$$

Focusing on CNF formulas of constant width, we have the following corollary.

Corollary 4.9 ([IPS99]). *Let F be an unsatisfiable k-CNF formula over n variables for $k = \mathrm{O}(1)$. Then*

$$S_{\mathcal{PCR}}(F \vdash \bot) = \exp\left(\Omega\left(\frac{(Deg_{\mathcal{PCR}}(F \vdash \bot))^2}{n}\right)\right) \ .$$

Looking back to our study of resolution, it should be clear that Theorem 4.8 and Corollary 4.9 are very, very similar to the relations between length and width in [BW01]. And indeed, the results in [IPS99] were very much an inspiration for [BW01]. Even more can be said: the proofs in [BW01] are essentially a translation line by line of the corresponding proofs in [IPS99] for PC to the resolution setting. But of course, this is obvious only in hindsight. And it was a truly brilliant translation—one that has made [BW01] one of the most cited papers in all of proof complexity. The moral of this, perhaps, is that often the really great results are the simple but ingenious ones.

In the next lecture we will prove Theorem 4.8 (and hence Corollary 4.9), but this is all we had for today.

Acknowledgements

The lecturer wants to thank Massimo Lauria and Alexander Razborov for helping with pointers and explanations for the material for this lecture. (Of course, any remaining errors or ambiguities are solely the lecturer's responsibility.) Thanks also to the scribe Björn Terelius for writing up the material for the "detour" in Section 2.2.

References

[ABRW00] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC '00)*, pages 358–367, May 2000.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version appeared in *STOC '00*.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, May 2008. Preliminary version appeared in *CCC '03*.

[BD09] Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *Journal of Symbolic Computation*, 44(9):1326–1345, September 2009.

[BDG$^+$09] Michael Brickenstein, Alexander Dreyer, Gert-Martin Greuel, Markus Wedler, and Oliver Wienand. New developments in the theory of Gröbner bases and applications to formal verification. *Journal of Pure and Applied Algebra*, 213(8):1612–1635, August 2009.

[BG01] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, December 2001. Preliminary version appeared in *FOCS '99*.

[BIK$^+$97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiri Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version appeared in *STOC '99*.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, May 1996.

[FLN$^+$11] Yuval Filmus, Massimo Lauria, Jakob Nordström, Neil Thapen, and Noga Zewi. Space complexity in polynomial calculus. Submitted, December 2011.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiri Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.