CS 880: Advanced Complexity Theory 4/7/06

Lecture 25: Proof Complexity

Instructor: Dieter van Melkebeek Scribe: Scott Diehl, Jeff Kinne

Last time we discussed our first application of expanders to lower bounds, namely, lower bounds on the distortion of metric embeddings. Next, we turn our attention to the application of expanders to lower bounds for proof complexity. Today's lecture is an introduction to this field.

The study of proof complexity is intimately related to the question of NP versus coNP. It studies the possibility of the existence of short proofs for tautologies, a coNP-complete problem. The existence of a short proof depends on the choice of proof system, which we formally define as:

Definition 1 (Proof system)**.** *A proof system is a polynomial-time computable mapping f from Σ^* onto the set of Boolean tautologies.*

The interpretation we apply to such a system is that if $f(x) = y$, then x is a proof that y is a tautology. If x is an invalid proof, then we set $f(x) = z$ for z some fixed tautology, say, $a \vee \overline{a}$. Notice that this definition is indeed sound, since f does not map to any non-tautologies, and it is complete since it is onto (and therefore can prove all tautologies). The restriction that f be polynomial-time computable enforces that the proofs can be efficiently verifiable.

The best possible proof system that we could hope for would have a proof whose length is polynomial in the size of the tautology it is proving.

Definition 2 (Polynomially-bounded proof system)**.** *A proof system is said to be polynomially bounded if there is a constant $c > 0$ such that for all tautologies y, there is a proof x with $|x| \leq |y|^c$ such that $f(x) = y$.*

Clearly, such fantastic proof systems exist if and only if $\mathsf{NP} = \mathsf{coNP}$. Therefore, we can make progress towards proving $\mathsf{NP} \neq \mathsf{coNP}$ by considering different proof systems and proving they are not polynomially bounded– namely, by showing that certain families of tautologies require superpolynomial length proofs within a given system.

1 Examples of Proof Systems

We now discuss several classes of proof systems, some more powerful than others.

(0) **Truth table**– The simplest proof system one can think of is to simply list the value of a formula under all assignments to its variables. It is especially easy to verify such a proof, but clearly such a system is not polynomially bounded since there are exponentially many assignments in the truth table.

(1) **Logic based systems**– Logic based systems use logical formulae as the basic objects to try to prove tautologies.

Frege systems– Frege systems are a particular type of logic based system, which are based on a finite set of axioms (i.e. $A \vee \overline{A}$) and inference rules, such as modus ponens,

$$\frac{A \qquad A \rightarrow B}{A}$$

which says that if A is a tautology and $A \to B$ is a tautology, then so is B. A proof within a Frege system is a sequence of formulae (called *lines*), each of which is either an instantiation of an axiom (e.g. for the axiom $A \vee \overline{A}$, we can replace A by any formula) or the result of an inference rule applied to previous lines. The final line of the proof must be the formula which we wish to prove is a tautology.

One way to classify the various Frege systems is by the complexity of the formulae allowed in lines of a proof. For example, we might only allow DNF formulae or bounded-depth formulae, which characterize *bounded-depth* Frege systems (DNF is an instantiation of the latter for depth one). More powerful systems allow arbitrary formulae, and are referred to simply as Frege systems. The most powerful and general system allows functions computed by circuits as lines in the proofs– of course, to write these out as formulae of reasonable size, we need to allow for binding of variables to expressions (i.e. variables can be used to name expressions and allow their reuse without copying them down many times). The latter systems are known as *extended* Frege systems.

Resolution– A particular depth-one Frege system which has received much attention is known as *resolution*. This system is a refutation proof system, which proves that a formula φ is a tautology by proving that $\neg\varphi$ is not satisfiable. Each line in this proof is a disjunctive clause, and the proof starts with the clauses in the CNF formula $\neg\varphi$. There is one inference rule - called *resolution*,

$$\frac{A \vee x \qquad\qquad B \vee \overline{x}}{A \vee B}$$

which takes two clauses that share a variable x, but appearing with different sign, and outputs the join of the clauses without x appearing. Notice that this rule is sound, since if both $A \vee x$ and $B \vee \overline{x}$ are satisfied, then so is $A \vee B$. Therefore, we can see that at each step, if $\neg\varphi$ is satisfiable, then there is an assignment that simultaneously satisfies all possible lines derived by this inference rules. Thus, we deduce that $\neg\varphi$ is not satisfiable if we can derive the empty clause from $\neg\varphi$. Such a derivation of the empty clause is called a *refutation*. In addition to the resolution rule, we will at times use the *weakening rule*:

$$\frac{A}{A \vee x}$$

where x is any literal. Intuitively, this should not provide us with any additional power. This can be proven formally - for an unsatisfiable formula ϕ, if there is a proof using resolution and weakening with width w and length ℓ, then there is also a proof using only resolution that uses width at most w and length at most ℓ. However, it will at times make our arguments simpler if we allow weakening.

The above shows that resolution is sound. We now show that resolution is complete. Given an unsatisfiable formula, we wish to construct a resolution refutation (a sequence of resolutions ending in the empty clause). Consider a tree formed by a brute force search for a satisfying assignment to $\neg\varphi$ which is pruned when a clause is found to be violated by a partial assignment (see Figure 1). We know that every path on the tree will eventually be pruned, because we are assuming an un-satisfiable formula. We can use this tree to construct a refutation. We will associate with each leaf node the clause that is violated by that partial assignment. For each internal node, we will resolve the clauses associated with its children on the variable of the internal node. It is easy to show by induction that we can always do this (we may have to use the weakening rule if a clause does not contain the appropriate variable), and that each node in the tree will have an associated formula that: is unsatisfiable given the partial assignment up to that point, and only contains variables that have been assigned up to that point. When we get to the root of the tree, the last resolution will yield the empty clause.

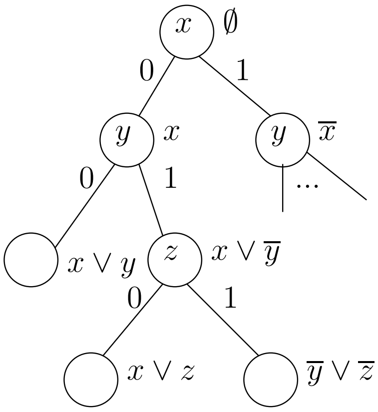

Figure 1: Example of construction of a tree that can always be used to give a refutation of an un-satisfiable formula. This example is for the DNF $\{x \vee y, x \vee z, y \vee z, \overline{x} \vee \overline{y}, \overline{x} \vee \overline{z}, \overline{y} \vee \overline{z}\}$

(2) **Geometry based systems**– Another class of proof systems treats clauses as geometric objects. One system in particular is the "cutting planes" system, which is another refutation system (like resolution). Here, we interpret each clause as a linear inequality defining a hyperplane. For example, $x \vee \overline{y} \vee z$ gives rise to $x + (1 - y) + z \geq 1$, which can be rewritten as $x - y + z \geq 0$. As inference rules, we can add two inequalities, multiply by a positive integer, or divide by a common divisor of the left-hand coefficients, rounding the quotient on the right-hand side up. More formally, the last rule is: if there is an integer $c > 0$ such that $c|a_i$ for all i, we have the inference rule

$$\frac{(\sum_i a_i x_i) \geq b}{(\sum_i \frac{a_i}{c} x_i) \geq \lceil \frac{b}{c} \rceil}$$

This rule is critical to the power of this system. A refutation proof in the cutting planes system is a derivation of the inequality $0 \geq 1$.

(3) **Algebraic systems**– This class of proof systems treats clauses as algebraic objects. A particular refutation system in this class is the "polynomial calculus". This system uses an arithmetization of clauses to encode each as a multi-linear polynomial over some field, which evaluates to zero under a 0/1 assignment when the clause is satisfied. For example, $x \vee \overline{y} \vee z$ is encoded as $(1-x)y(1-z)$. For each variable, we also include axioms $x(1-x) = 0$ to enforce that x is given a 0/1 assignment. The key idea for this system is Hilbert's Nulstellensatz, which says that a list of multivariate polynomials over some algebraically closed field does not have a common zero if and only if 1 is in the ideal generated by these polynomials (i.e. 1 can be written as some linear combination of the polynomials with arbitrary polynomial coefficients). Therefore, we have inference rules which allow us to generate the elements of the

ideal as lines in our proof, namely, addition of polynomials and multiplication by variables, so that a refutation consists of a derivation of the identically 1 polynomial.

The parameter we are most concerned about within each system is the proof sizes. Other parameters often come into play, such as the maximum degree of a line in a polynomial calculus proof, or the maximum clause width in a resolution proof. The latter parameter has a strong connection to resolution proof size, and in fact, next time we show an exponential lower bound for proof size of certain tautologies based on a lower bound for clause width.

It has been established that many of the proof systems we named are not polynomially bounded. For example, we have exponential lower bounds for the resolution proof size of various tautologies, as well as for bounded-depth Frege systems in general. However, for general Frege systems and extended Frege systems, the question is still open. Cutting plane systems also have been shown to require exponential size proofs for certain tautologies, and so are also not polynomially bounded.

2 Specific Families of Contradictions

We now introduce two families of contradictions for which we will prove exponential lower bounds on the resolution proof size next time.

(1) **Pigeonhole principle**– We denote by PHP_n^m the formula encoding the pigeonhole principle with m pigeons and n holes (which tries to assign m pigeons into n holes so that no more than one pigeon is in each hole). So PHP_n^m is a contradiction for $m > n$. We can easily encode PHP_n^m with variables x_{ij}, where the meaning is that x_{ij} is true if and only if pigeon i is in hole j. Then PHP_n^m consists of the following clauses:

$$\bigvee_{j=1}^{n} x_{ij},$$

for $1 \leq i \leq m$, which enforces that every pigeon is in at least one hole, and

$$\overline{x_{ij}} \vee \overline{x_{i'j}},$$

for $1 \leq i < i' \leq m$ and $1 \leq j \leq n$, which enforces that no hole has two pigeons. Intuitively, it should be hardest to refute PHP_n^m when $m = n + 1$, since there is only one place where any assignment of pigeons to holes goes wrong. Therefore it should be easier to prove lower bounds on proof sizes for the *strong pigeonhole principle*, where we enforce $m = n+1$, than for the weak pigeonhole principle (where m is arbitrary). Indeed, there is a lower bound of $2^{\Omega(n)}$ on the size of resolution refutations for the strong pigeonhole principle, and $2^{\Omega(n^{1/3})}$ for the weak pigeonhole principle. However, even the lower bound for the strong pigeonhole principle isn't as strong as we'd like, since the exponent is not linear in the number of variables, n^2.

(2) **Tseitin contradictions**– We will be able to derive exponential size lower bounds for resolution in which the exponent is linear in the number of variables by considering statements about edge/vertex parities on expander graphs. Namely, for any graph $G = (V, E)$ and assignment of charges to the vertices $\sigma : V \to \{0, 1\}$, we consider the formula over variables x_e for $e \in E$:

$$\forall v \in V, \oplus_{e \text{ incident to } v} x_e = \sigma(v).$$

We can see that this forms a contradiction if and only if the parity of σ is odd– the reverse direction follows since adding all edges incident to each vertex counts each edge twice, resulting in a sum (mod 2) of zero for any assignment to the edges. The other direction follows easily by induction.

Next time, we will show that when G is an expander, there is a resolution refutation size lower bound of $2^{\Omega(N)}$. Since expanders are constant degree, the number of variables becomes $O(N)$, and so the exponent of the lower bound is indeed linear in the number of variables.