6.841/18.405J: Advanced Complexity **Monday, April 28th, 2003**

Lecture 20: Proof Complexity, part I

Instructor: Eli Ben-Sasson *Scribe: Samuel Fiorini*

1 Introduction

Suppose we are given a Boolean circuit $\mathcal{C}(x_1, \dots, x_n)$ (later $\mathcal{C}$ will be a CNF formula). Then any satisfying assignment $\alpha \in \{0,1\}^n$ is a short proof of the claim

$$\exists \alpha \in \{0,1\}^n : \mathcal{C}(\alpha) = 1.$$

Hence the language SAT has short membership proofs. By definition, any language in NP has short membership proofs. However if we were to prove the claim

$$\forall \alpha \in \{0,1\}^n : \mathcal{C}(\alpha) = 0,$$

then the simplest way would be to enumerate all possible assignments $\alpha \in \{0,1\}^n$. In other words, we can easily find exponential size membership proofs for co-SAT. Therefore a natural question is: "given a circuit $\mathcal{C} \in$ co-SAT, what is the size of a shortest proof that $\mathcal{C} \in$ co-SAT?". This is a central question in proof complexity. Let us now formally define proof systems.

Definition. A *proof system P* (for *co-SAT*) is a 2-tape Turing Machine $P(.,.)$ such that

1. (efficiency) P runs in polynomial time in both inputs,
2. (soundness and completeness) $\mathcal{C} \in$ co-SAT iff there exists a string π such that $P(\mathcal{C}, \pi) = 1$.

Observe that proof systems are not allowed to use randomness or interaction, and the size of the proof π is not required to be polynomial in the size of the circuit $\mathcal{C}$. Adding this latter requirement gives the notion of 'super' proof systems:

Definition. A proof system P is called *super* if there exists a positive integer k with the following property. For all circuits $\mathcal{C} \in$ co-SAT, there exists a proof $\pi_{\mathcal{C}}$ such that $|\pi_{\mathcal{C}}| \leq |\mathcal{C}|^k$ and $P(\mathcal{C}, \pi_{\mathcal{C}}) = 1$.

The main open question of proof complexity can be now stated as : "does there exist a super proof system?". This question is actually equivalent to NP =? co-NP.

Theorem 1 (Cook,Reckhow) *There exists a super proof system iff NP = co-NP.*

Proof. Assume there exists a super proof system. Then there exists a polynomial size proof that $\mathcal{C}$ is unsatisfiable for every circuit $\mathcal{C} \in$ co-SAT. Therefore co-SAT $\in$ NP, hence co-NP $\subseteq$ NP. Similarly it implies that SAT $\in$ co-NP, hence NP $\subseteq$ co-NP.

Conversely, if co-NP = NP, then co-SAT $\in$ NP and there exists a verifier $V(.,.)$ for co-SAT that runs in polynomial time in the first input. Hence V yields a super proof system. ■

The motivations of proof complexity include:

1. *Circuit design*: suppose that we would like to verify that a circuit $\mathcal{C}$ computes a given function f (e.g., that a certain chip really computes the product of two numbers). We would like to have an algorithm based on some proof system that could prove statements of the form $f \equiv \mathcal{C}$ as efficiently as possible.

2. *The* NP $\not\subseteq$ P/poly *question*: empirically it seems to be a difficult question. But can we prove that there is something inherently difficult in it? It turns out that finite versions of this statement can be shown to be hard to prove for some fixed proof system.

3. *Automated theorem proving and related questions*: for instance, suppose there exists a short proof for a given statement, can we efficiently find such a proof?

4. *Analysis of algorithms for NP-complete problems*: the execution of such an algorithm (even if it is exponential) can be used to prove that a certain circuit in co-SAT is not satisfiable. Therefore lower bounds on the length of a proof give rise to lower bounds on the running time of the algorithm.

2 Resolution

¿From now on, we restrict circuit $\mathcal{C}$ to be a CNF formula. Recall that a *literal* is either a variable x or its negation $\overline{x}$, a *clause* is a disjunction of literals (sometimes encoded as a set of literals), and a *CNF formula* is conjuction of clauses (which can also be encoded as a set of clauses).

The resolution mechanism starts with the set of clauses of a given CNF formula and sequentially derives new clauses from previous clauses until it shows that, in a satisfying assignment, a certain variable has to be set to 0 and 1 at the same time. This proves the unsatisfiability of the formula. Let $\mathcal{C}$ be a CNF formula over the variables $x_1, x_2, \ldots, x_n$, a resolution proof uses the two following rules to generate new clauses:

1. *Resolution Rule:* From any two clauses of the form $C \vee x_k$ and $D \vee \overline{x_k}$, derive $C \vee D$. This is often written
$$\frac{C \vee x_k \quad D \vee \overline{x_k}}{C \vee D}$$

2. *Weakening Rule:* From clause $\mathcal{C}$, derive clause $\mathcal{C} \vee \ell$, where ℓ is any literal.
$$\frac{C}{C \vee \ell}$$

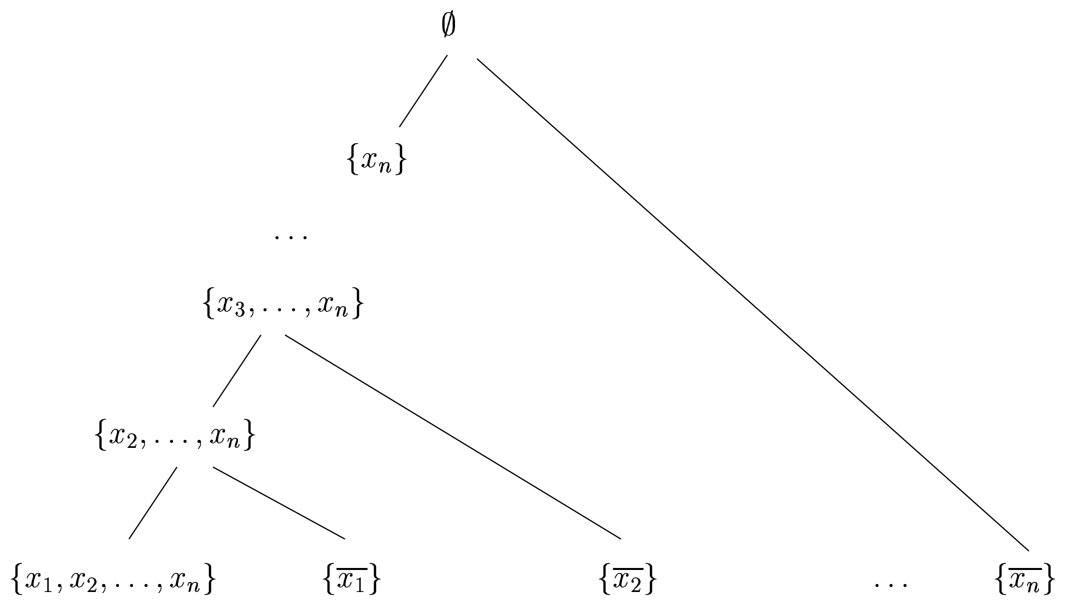
The weakening rule is not necessary to enforce the completeness of resolution, but it turns out to be useful in proofs. A *resolution proof* (or *refutation*) of *length s* for $\mathcal{C}$ is a sequence $\pi = (L_1, \ldots, L_s)$ of *lines* such that

(i) each line is a clause,

(ii) the last line L_s is the empty clause $\emptyset$ (which is unsatisfiable by definition),

(iii) each line L_i is either an *axiom* (that is, an element of $\mathcal{C}$) or derived from previous lines via the resolution or weakening rule.

Example. Consider the CNF formula $\mathcal{C}$ on variables $x_1, \ldots, x_n$ with clauses $C_0 = \{x_1, x_2, \ldots, x_n\}$, $C_1 = \{\overline{x_1}\}$, $C_2 = \{\overline{x_2}\}, \ldots, C_n = \{\overline{x_n}\}$. We can prove that $\mathcal{C}$ is unsatisfiable by resolution as follows. First, apply the weakening rule to C_0 and C_1 to obtain the clause $\{x_2, x_3, \ldots, x_n\}$. Then apply the weakening rule to this new clause and C_2 to obtain the clause $\{x_3, x_4, \ldots, x_n\}$. Continue in a similar fashion until you infer the empty clause. The corresponding resolution proof is $\pi = (L_1, \ldots, L_{2n+1})$ with
$$L_i = \begin{cases} C_{i-1} & \text{if } 1 \le i \le n+1, \\ \{x_1, x_2, \ldots, x_n\} \setminus \{x_1, \ldots, x_{i-n-1}\} & \text{if } n+2 \le i \le 2n+1. \end{cases}$$



A resolution proof gives rise to a DAG (Directed Acyclic Graph) called the *proof-DAG*. The nodes of the DAG are the lines of the proof, and a node has one edge coming in from each one of the premises it was derived from. Thus, the *sources* of the proof-DAG are the axioms, and the DAG has a single sink node, which is labeled $\emptyset$. We allow to label more than one node with the same axiom.

A proof is said to be tree-like if its DAG is a tree. For instance, the above proof is tree-like. In general, resolution proofs are not necessarily tree-like. Although there are contradictions $\mathcal{C}$ which admit small (linear length) resolution proofs but need long (exponential length) tree-like resolution proofs, tree-like resolution is very interesting from a theoretical and practical point of view. For instance, most known SAT algorithms (such as the DLL and DP algorithms) use tree-like resolution.

3 Complexity measures

Let $\mathcal{C}$ be a CNF formula. Then we define

$$\begin{aligned} S_R(\mathcal{C}) &= \text{min size of a resolution proof for } \mathcal{C}, \\ S_T(\mathcal{C}) &= \text{min size of a tree-like resolution proof for } \mathcal{C}, \\ \text{width}(\mathcal{C}) &= \min_{\pi \text{ proof for } \mathcal{C}} (\text{max width of a line in } \pi), \end{aligned}$$

where the *size* of a proof is the number of symbols in it (this measure is polynomially related to the length for resolution proofs), the *width* of a clause is the number of literals in it. By convention, all above quantities equal ∞ if $\mathcal{C}$ is satisfiable. At first sight, it may seem that $S_T(\mathcal{C})$ and $S_R(\mathcal{C})$ are the most interesting parameters, but it turns out that width($\mathcal{C}$) is also very interesting in that lower bounds on width($\mathcal{C}$) are often easier to obtain and yield lower bounds on $S_T(\mathcal{C})$ and $S_R(\mathcal{C})$.

4 Completeness of proof by resolution

Theorem 2 *A CNF formula is unsatisfiable if and only if it admits a resolution proof.*

Proof. The backwards implication follows directly from the soundness of the resolution and weakening rules, so let's do the forward implication. Let $\mathcal{C}$ be a CNF formula on variables x_1, …, x_n. If $n = 0$ then $\mathcal{C}$ has only one clause, namely the empty clause, so it has a one line resolution proof.

Now assume that $n \geq 1$ and every unsatisfiable CNF formula with $n-1$ variables has a resolution proof. We will construct a CNF formula $\mathcal{C}'$ such that (i) $\mathcal{C}'$ has $n-1$ variables, (ii) $\mathcal{C}'$ is derived from $\mathcal{C}$ by the resolution rule (we won't use the weakening rule), and (iii) $\mathcal{C}'$ is unsatisfiable. Let

$$\mathcal{C}' = \bigwedge_{\alpha \in \{0,1\}^{n-1}} C'_\alpha$$

for some clauses C'_α to be determined. Fix $\alpha \in \{0,1\}^{n-1}$. Because $\mathcal{C}$ is unsatisfiable, it must possess clauses C_0 and C_1 such that $C_0(\alpha, 0) = C_1(\alpha, 1) = 0$.

Case 1. If x_n is not a literal of C_0 then we set $C'_\alpha = C_0$ ($\overline{x_n}$ cannot be a literal of C_0).

Case 2. If $\overline{x_n}$ is not a literal of C_1 then we set $C'_\alpha = C_1$ (x_n cannot be a literal of C_1).

Case 3. Otherwise $C_0 = C'_0 \vee x_n$ and $C_1 = C'_1 \vee \overline{x_n}$ for some clauses C'_0 and C'_1 on $x_1, \ldots, x_{n-1}$. We then derive $C'_\alpha = C'_0 \vee C'_1$ by one application of the resolution rule.

This concludes the construction of $\mathcal{C}'$. By induction, we know that $\mathcal{C}'$ has a resolution proof π'. We can find a proof for $\mathcal{C}$, for instance, by inserting in π' all the axioms of $\mathcal{C}$ and then the clauses of $\mathcal{C}'$ which are derived from two axioms of $\mathcal{C}$ by the resolution rule. ■

Example. For $i \in 1, \ldots, n$ and $\alpha \in \{0,1\}^n$, let $\ell_{\alpha,i} = x_i$ if $\alpha_i = 0$ and $\ell_{\alpha,i} = \overline{x_i}$ if $\alpha_i = 1$. Then define

$$C_\alpha = \bigvee_{i=1}^{n} \ell_{\alpha,i} \qquad \text{and} \qquad \mathcal{C} = \bigwedge_{\alpha \in \{0,1\}^n} C_\alpha .$$

The 'universal formula' $\mathcal{C}$ has 2^n clauses, each of width n. Each of its clauses C_α satisfies all assignments except α. It follows that $\mathcal{C}$ has no resolution proof of length less than 2^n.

The above formula has no short resolution proof in terms of n, but is itself very long. Can we find short CNF formulas that have long resolution proofs ? Random CNF formula have been proved to be hard for resolution, but there are also explicit examples of CNF formulas which require long resolution proofs, as we will see now.

5 Formulas which are hard for resolution

The *pigeonhole principle* says that it is impossible to squeeze $n+1$ pigeons in n holes if no two pigeons fit in one hole. We can express the pigeonhole principle by a CNF formula PHP_n as follows. PHP_n lies and says "there is a way to squeeze $n+1$ pigeons in n holes". For each pigeon $i \in \{1, \ldots, n+1\}$ and hole $j \in \{1, \ldots, n\}$, formula PHP_n has a variable x_{ij}. The interpretation of x_{ij} is: $x_{ij} = 1$ if pigeon i is in hole j. Then we can express "pigeon i occupies at least one hole" by a clause $P_i = \bigvee_{j=1}^{n} x_{ij}$ and "pigeons i and i' are not both in hole j" by a clause $H_{i,i',j} = \overline{x_{ij}} \vee \overline{x_{i'j}}$. Then

$$\mathrm{PHP}_n = \left(\bigwedge_{1 \leq i \leq n+1} P_i \right) \wedge \left(\bigwedge_{\substack{1 \leq i \neq i' \leq n+1 \\ 1 \leq j \leq n}} H_{i,i',j} \right)$$

expresses what we want.

Theorem 3 (Haken 1989) $S_R(\mathit{PHP}_n) = 2^{\Omega(n)}$.

So PHP_n is indeed hard for resolution. But there are more CNF formulas which are hard for resolution:

Theorem 4 (Raz 2002) *Resolution cannot efficiently prove statements of the type "function f is not in P/poly".*

There are two ways to interprete this results: (i) it is hard to prove circuit lower bounds, (ii) resolution is weak. (Interestingly, Ran Raz derived the latter result from a lower bound on the minimum length of a resolution proof for the *weak pigeonhole principle.*) This is related to work by Razborov and Rudich on 'natural proofs'. They showed that all known circuit lower bounds have certain 'natural' properties, and that the existence of hard functions contradicts the existence of natural proofs of general circuit lower bounds. Thus, a circuit lower bound for arbitrary circuits might require some totally new proof techniques. The ultimate goal in this direction, which currently seems well beyond our reach, is to prove the *independence* of the claim P $\neq$ NP from some strong first order theory, such as ZFC.

6 Two techniques for obtaining lower bounds

In the next lecture, we will prove lower bounds for the resolution proof system. Given a CNF formula $\mathcal{C}$, we want to know what is it about the structure of $\mathcal{C}$ that makes it hard to prove for a fixed proof system like resolution. There are two general approaches:

1. investigate the combinatorics of DAG-proofs;
2. use the interpolation method.

The interpolation method implements a natural idea that fails at some point: connect CNF formulas to circuits and use circuit lower bounds. The basic argument here is "if this CNF formula has a short proof, then some function can be computed by a small circuit". Consider an unsatisfiable CNF formula $\mathcal{C}$ of the form $\mathcal{C} = \mathcal{C}_0(x, y) \wedge \mathcal{C}_1(x, z)$. Krajíček's theorem says that if we have a short resolution proof for $\mathcal{C}$ then there exists a small circuit that determines which one of $\mathcal{C}_0$ or $\mathcal{C}_1$ is not satisfiable when the x variables have been fixed to some given truth values.

Theorem 5 (Krajíček's interpolation theorem) *If $\mathcal{C} = \mathcal{C}_0(x, y) \wedge \mathcal{C}_1(x, z)$ has a resolution proof of length s (so, in particular, $\mathcal{C}$ is unsatisfiable), then there exists a circuit $S(x)$ of size at most s that computes, for each $\alpha \in \{0, 1\}^{|x|}$, an index $i = S(\alpha)$ with the following property: if $i = 0$ then $\mathcal{C}_0(\alpha, y)$ is unsatisfiable; if $i = 1$ then $\mathcal{C}_1(\alpha, z)$ is unsatisfiable.*

Note that the interpolation method fails for proof systems which are stronger than resolution.