CSC 2429 – Approaches to the P versus NP Question

Lecture #12: April 2, 2014

Lecturer: David Liu (Guest)

Scribe Notes by: David Liu

1 Introduction

The primary goal of complexity theory is to study the power of different computational resources: time, space, randomness, nondeterminism, and circuit size, to name a few. Yet despite decades of work from the best and brightest in the field, we seem to be very far from answering the most simple of questions, with P vs. NP being the most famous of these.

For the successes we have found, we have been excited not just by the results proven, but by the proofs themselves: *could this new technique X or idea Y be applied to larger questions*? Unfortunately, three times in the history of complexity theory, researchers have surveyed the current swath of techniques, distilled some common essences from them, and shown that no proof stemming from these bases could ever prove, for example, that $\mathsf{P} \neq \mathsf{NP}$. That is, they identified *proof barriers* to questions in complexity theory that would require fundamentally new techniques to overcome.

The first two barriers were *relativization* and *natural proofs*. The former concerns the notion of boolean oracles, which essentially give complexity classes the power to compute a single arbitrary boolean function. In 1975, Baker, Gill, and Solovay [2] observed that giving such oracles to P and NP could radically change their relative power, and so any resolutions of the P vs. NP question (with either outcome!) must involve techniques that were sensitive to such changes. The major significance of the observation was that *diagonalization*, a technique that had been used to great success in computability theory, could not be a viable approach to resolving this question. In 1994, Razborov and Rudich [8] took the existing circuit lower bound techniques and showed that they all consisted of finding some "hard property" for a given circuit class that was efficiently computable and shared by a non-trivial fraction of boolean functions. They showed that doing so for more powerful classes (like P) would disprove the existence for pseudorandom generators for these classes, contradicting a belief that is widely held to this day.

But even with these two barriers in place, researchers had reason to be hopeful: diagonalization, though relativizing, was not a natural proof. The proofs that $\mathsf{PH} \subseteq IP$ [7] and $\mathsf{IP} = \mathsf{PSPACE}$ [10] used a clever algebraic idea to overcome the relativization barrier, and the later proof that $\mathsf{MA_{EXP}} \not\subset \mathsf{P/poly}$ in [3] overcame both barriers!

In this lecture, we will discuss the third proof barrier, presented by Aaronson and Wigderson in [1]: *algebrization*. The rest of this lecture will be divided into four main parts, and can be seen as a very abridged version of the original paper: first, we define the notions relevant to algebrization; second, we show that the proof that $\mathsf{IP} = \mathsf{PSPACE}$ algebrizes; third, we show that any resolution

of the P vs. NP question will require non-algebrizing techniques; fourth, we give some interesting connections between algebrization and communication complexity.

2 What is Algebrization?

As mentioned in the introduction, relativization is a proof barrier rooted in the varying power of boolean oracles, while [10] overcame this barrier by "arithmetization", a conversion of boolean formulas into polynomials. The natural thing to do, then, is increase the strength of the barrier by replacing boolean oracles by algebraic oracles – thus *algebraic relativization*, shortened to *algebrization*, was born. We begin by reminding the reader about the standard boolean oracle case.

Definition A *boolean oracle* A is a family of boolean functions $\{A_m\}$, one for each input size. For a complexity class $\mathcal{C}$, the relativized class $\mathcal{C}^A$ with respect to the oracle is the set of boolean functions computable by $\mathcal{C}$-machines which can also make queries to the A_m as single steps in their computation.

Theorem (Baker, Gill, Solovay [2])**.** *There exists a (boolean) oracle A such that $\mathsf{P}^A = \mathsf{NP}^A$, and a (boolean) oracle B such that $\mathsf{P}^B \neq \mathsf{NP}^B$.*

The consequence of this result is that any proof that separates P and NP cannot also hold if the classes are augmented by an arbitrary boolean oracle, nor could any proof that $\mathsf{P} = \mathsf{NP}$.

2.1 Algebraic Oracles

Now, we extend this to the algebraic world.

Definition Given a boolean oracle $A = \{A_m\}$, an *algebraic extension oracle* $\tilde{A}$ is a collection of low-degree polynomials $\tilde{A}_{m,\mathbb{F}}$, one for every m and finite field $\mathbb{F}$, such that for all $\tilde{A}_{m,\mathbb{F}}$ and $x \in \{0,1\}^m$, $\tilde{A}_{m,\mathbb{F}}(x) = A_m(x)$.

Note the condition "low-degree" in this definition, which refers to a bound on the individual degree of each variable; in [1] this is simply a constant independent of m and $\mathbb{F}$, but it turns out they only need degree-2 polynomial extensions for all of their results. Also note that algebraic oracles are not unique: a single boolean function can be extended in different ways over a finite field, even when the degree is restricted to 2 (though the degree-1 multilinear extension is unique).

We can now define the notion of algebrization.

Definition Let $\mathcal{C}, \mathcal{D}$ be complexity classes. We say that the inclusion $\mathcal{C} \subseteq \mathcal{D}$ *algebrizes* if for all boolean oracles A and algebraic extension oracles $\tilde{A}$, $\mathcal{C}^A \subseteq \mathcal{D}^{\tilde{A}}$. Similarly, a separation $\mathcal{C} \not\subset \mathcal{D}$ *algebrizes* if for all $A, \tilde{A}$, $\mathcal{C}^{\tilde{A}} \not\subset \mathcal{D}^A$.

The strangest part of the above definition is the asymmetry in assigning the oracles: in both cases, only one class gets the algebraic oracle, while the other gets the boolean oracle. First, we note that because we can evaluate the boolean functions A_m by querying their polynomial extensions, $\mathcal{C}^A \subseteq C^{\tilde{A}}$; as a consequence of this, any inclusion or separation result that relativizes also algebrizes. However, the honest reason for this choice of definition is that this is the one for which we are currently able to prove results; Aaronson and Wigderson name it an interesting open problem to prove stronger results like $\mathsf{NP}^{\tilde{A}} \not\subset \mathsf{P}^{\tilde{A}}$.

3 Algebrizing Proof Techniques

Aaronson and Wigderson showed that several existing proven results that are non-relativizing are, in fact, algebrizing. This shows some limitations on the proof techniques like arithmetization that avoid relativization. Here, we focus on one such result: the fact that the existing "arithmetization" proof that $\mathsf{IP} = \mathsf{PSPACE}$ can be altered just slightly to prove the stronger algebrizing result. For the remainder of this paper, all numbered results refer to their original numbering in [1].

Theorem (Theorem 3.7)**.** *For all boolean oracles A and algebraic extensions $\tilde{A}$, $\mathsf{PSPACE}^A \subseteq \mathsf{IP}^{\tilde{A}}$.*

Proof. Because this proof is essentially the same as the (non-algebrized) proof given in [10], we will only give a sketch here. First, recall the basic ingredients of the proof that $\mathsf{PSPACE} \subseteq \mathsf{IP}$. The main idea is to give an interactive proof protocol for $TBQF$ (true quantified boolean formulas). We use "arithmetization": given a boolean formula F, we can extend this to a polynomial $P(x_1, \ldots, x_n)$ over some finite field $\mathbb{F}$. This reduces determining some quantified statement about F to evaluating some expression of sums and products of the polynomial's boolean values in the natural way.[1] Evaluating this expression directly requires exponentially many polynomial evaluations; the main trick of the proof is to using interaction with the prover to reduce the number of variables of the polynomial until the remaining expression has just one variable, for which can be substituted 0 and 1 directly. The key is that the prover does all the work, and this simplification is done without the verifier having to use either the original polynomial or formula. Then for soundness, the verifier checks the proposed single variable expression against the original at just two points (selected randomly by the verifier during the protocol); here is where the original polynomial is verified.

Adding the oracles A and $\tilde{A}$ changes just the first and last steps. Observing that $TQBF^A$ is PSPACE^A-complete, we begin with a QBF with A gates embedded in the formula; importantly, the arithmetization trick which turns formulas into polynomials can simply turn every instance of an A into the corresponding $\tilde{A}$. For example, that clause $x_1 \vee A_3(x_2, x_3, x_4)$ might become the polynomial $1 - (1 - x_1)(1 - \tilde{A}_{3,\mathbb{F}}(x_2, x_3, x_4))$. Then, in in the final step when the verifier must evaluate the arithmetized polynomial directly, it uses its (algebraic) oracle to do so. □

4 Non-algebrizing Relationships

In this section, we prove that the resolution of the P vs. NP question and the conjecture $\mathsf{NEXP} \not\subset \mathsf{P/poly}$ both do not algebrize, and therefore will require non-algebrizing proof techniques. To do this, we first develop a sequence of simple technical lemmas governing the limitations of querying an algebraic oracle.

Lemma (Lemma 4.2)**.** *Let $Y \subseteq \mathbb{F}^n$. Then there exists a* multilinear *polynomial $p : \mathbb{F}^n \to \mathbb{F}$ such that:*

- *$p(y) = 0$ for all $y \in Y$*
- *$p(w) = 1$ for at least $2^n - |Y|$* boolean *points*

[1] For example, the statement $\forall x_1 \exists x_2 F(x_1, x_2)$ is true if and only if $\prod_{i=0}^{1} \sum_{j=0}^{1} P(i, j)$ is non-zero. There is a subtlety about *linearizing* the polynomials to keep the degree bounded that we will avoid discussing further; for details, see [10].

Proof. For each boolean point $z \in \{0,1\}^n$, we define the polynomial $\delta_z(x) = \prod_{z_i=1} x_i \prod_{z_i=0}(1-x_i)$, which is 0 on all boolean points except z. This forms a basis for the vector space of all multilinear polynomials of n variables over $\mathbb{F}$.

Then consider the multilinear polynomial $p(x) = \sum_{z\in\{0,1\}^n} a_z\delta_z(x)$. Setting it equal to 0 on Y generates a system of $|Y|$ linear equations over the a_z's, leaving at least $2^n - |Y|$ free variables that we can choose to equal 1. □

Lemma (Lemma 4.3)**.** *Let $Y \subseteq \mathbb{F}^n$. Then for at least $2^n - |Y|$ boolean points w, there exists a* multiquadratic *polynomial $p : \mathbb{F}^n \to \mathbb{F}$ such that*

- *$p(y) = 0$ for all $y \in Y$*
- *$p(w) = 1$, and for all boolean $z \neq w$, $p(z) = 0$*

Proof. First we take a multilinear polynomial $p(x)$ satisfying Lemma 4.2. Then for each boolean point w such that $p(w) = 1$, the polynomial $q(x) = p(x)\delta_w(x)$ satisfies the conditions of the lemma. □

To provide some context, suppose that an oracle machine is trying to determine if a boolean function A is identically false; in the standard regular query complexity setting, it is allowed to query A on arbitrary boolean values, and it's not hard to see that 2^n queries are required. Lemma 4.2 says that even when given access to a multilinear extension of A over some chosen field $\mathbb{F}$, the machine still requires many queries (think of $\mathcal{Y}$ as the set of queries made), as there is a function which is quite far from being identically 0 that is still consistent with being 0 on a significant number of points in $\mathbb{F}^n$. Lemma 4.3 says something even stronger: even having made several, there are actually many different boolean functions that are almost identically 0 (0 on only 1 boolean point), but have polynomial extensions that are consistent with a set of queries being 0.

These two lemmas are limited in two ways: first, they deal with a single fixed field, though our algebraic oracle model allows for queries to be made to polynomials over different fields; and second, they limit the query outcomes to 0, and the form of the boolean functions. Lemma 4.5 is a generalization of the previous two that resolves both of these shortcomings:

Lemma (Lemma 4.5)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and let $\mathcal{F}$ be a collection of fields. For each $\mathbb{F} \in \mathcal{F}$, let $Y_\mathbb{F} \subseteq \mathbb{F}^n$, and $p_\mathbb{F}$ a multiquadratic extension of f over $\mathbb{F}$.*

Then there exists $B \subseteq \{0,1\}^n$, $|B| \leq \sum |Y_\mathbb{F}|$ such that for every $\mathbb{F} \in \mathcal{F}$ and boolean f' agreeing with f on B, f' has a multiquadratic extension $p'_\mathbb{F}$ that agrees with $p_\mathbb{F}$ on $Y_\mathbb{F}$.

Intuitively, the $Y_\mathbb{F}$ are the queries that have been made by the algebraic oracle machine; note that this lemma allows for any possible set of values to be returned by these queries, rather than fixing them all to 0. What this lemma says is that if t queries in total have been made, there is a set of at least $2^n - t$ *boolean* points such that the underlying function A could have any possible values on these points, but still have an algebraic extension that is consistent with the queries made. Put in even plainer terms, the oracle machine doesn't know anything about the values of these $2^n - t$ points. (Note that this is obviously true if we restrict the oracle to making boolean queries).

Using this result, we can prove the following theorem. The significance of this result is that any proof that $\mathsf{P} = \mathsf{NP}$ will be non-algebrizing.

Theorem (Theorem 5.3)**.** *There exist oracles A, $\tilde{A}$ such that* $\mathsf{NP}^A \not\subset \mathsf{P}^{\tilde{A}}$.

Proof. The proof we give is very similar to the one of Baker, Gill, and Solovay in [2]; the only difference is that Lemma 4.5 is required limit the power of the queries to the algebraic oracle.

Let L be the language of strings 1^n for which A_n is not identically 0. This is certainly in NP^A, since the machine could just guess some $w \in \{0,1\}^n$ and verify that $A_n(w) = 1$ using the (boolean) oracle. Let $M_1, M_2, \ldots$ be an enumeration of the $\mathsf{NTIME}(n^{\log n})$ algebraic oracle machines. We'll construct the oracles $A, \tilde{A}$ so that for all i, there exists an n_i such that $M_i(n_i)$ incorrectly answers whether or not $1^{n_i} \in L$. We proceed iteratively on the M_i, each time fixing some of the boolean functions A_m and corresponding extensions $\tilde{A}_{m,\mathbb{F}}$ *for every* $\mathbb{F}$. That is, after simulating M_i, we'll have fixed at least one collection $\{\tilde{A}_{m,\mathbb{F}} \mid \mathbb{F} \text{ finite}\}$ for at least one m.

For M_i, pick the smallest number n_i such that the $\tilde{A}_{n_i,\mathbb{F}}$ haven't been fixed, and $2^{n_i} > n_i^{\log n_i}$. We simulate $M_i(n_i)$, doing the obvious thing when choosing values returned by the oracle: every time M_i queries some $\tilde{A}_{m,\mathbb{F}}$ which has already been fixed, we return the fixed value; otherwise, we return 0. In particular, every time M_i queries some $\tilde{A}_{n_i,\mathbb{F}}$, we return 0. When this ends, we first fix all the $\tilde{A}_{m,\mathbb{F}}$, $m \neq n_i$, that had not yet been fixed at previous iterations.

If $M_i(n_i)$ outputs 1, we can clearly make this incorrect by choosing the $\tilde{A}_{n_i,\mathbb{F}}$ to all be identically 0, which is certainly consistent with the queries returned. On the other hand, if $M_i(n_i)$ outputs 1, then because the machine made at most $n_i^{\log n_i} < 2^i$ queries to the $\tilde{A}_{n_i,\mathbb{F}}$, by Lemma 4.5 there exists at least one boolean point w and choice of A_{n_i} so that $A_{n_i}(w) = 1$, and yet we can choose an extension of A_{n_i} to be consistent with the queries made by $M_i(n_i)$. □

Of course, most of us think that $\mathsf{P} \neq \mathsf{NP}$; we can prove that this requires non-algebrizing results as well. Note the flipped assignment of the oracles determined by the subtle asymmetry of the definition of algebrization; were this given the other way, it would follow immediately from the Baker-Gill-Solovay result, as $\mathsf{P}^A \subseteq \mathsf{P}^{\tilde{A}}$.

Theorem (Theorem 5.1)**.** *There exist oracles $A, \tilde{A}$ such that* $\mathsf{NP}^{\tilde{A}} \subseteq \mathsf{P}^A$.

Finally, though we didn't have time to go over this proof in lecture, Aaronson and Wigderson proved the following theorem, which shows that any separation of NEXP and $\mathsf{P/poly}$ requires non-algebrizing techniques.

Theorem (Theorem 5.6)**.** *There exist oracles $A, \tilde{A}$ such that* $\mathsf{NTIME}^{\tilde{A}}(2^n) \subset \mathsf{SIZE}(n)^A$.

5 Ties to Communication Complexity

We now turn our attention to some links between algebrization and communication complexity. First, we observe that Lemmas 4.2, 4.3, and 4.5 actually yield lower bounds on the *algebraic query complexity* of problems: that is, the worst-case number of queries necessary to compute some property of a boolean function A by querying some low degree polynomial extensions $\tilde{A}$.

5.1 Communication to Algebrization

It turns out that algebraic query complexity has a natural connection to communication complexity:

Theorem (Theorem 4.11)**.** *Let f be a property of boolean functions and A some boolean function such that $f(A)$ can be computed using at most T queries to $\tilde{A}$, the multilinear extension of A over the finite field $\mathbb{F}$.*

Consider the communication problem where Alice is given the truth table of $A|_{x_1=0}$ and Bob is given the truth table of $A|_{x_1=1}$. Then Alice and Bob can evaluate $f(A)$ using $O(Tn \log |\mathbb{F}|)$ bits of communication.

Proof. The protocol is simply for Alice and Bob to simulate an oracle machine computing $f(A)$ by using their inputs to answer the T algebraic queries the machine might make. To do this, we use the fact that because $\tilde{A}$ is multilinear, for any $y \in \mathbb{F}^n$, we can write $\tilde{A}(y) = \sum_{z \in \{0,1\}^n} \delta_z(y) A(z)$, where the δ_z are the same basis polynomial defined in the proof of Lemma 4.2.

So then the protocol is very straightforward: every time the machine makes some query $\tilde{A}(y)$, Alice sends Bob y, Bob computes his part of the sum, sends it to Alice, who then computes the value of $\tilde{A}(y)$ using the above sum identity. Alice and Bob exchange T numbers in $\mathbb{F}^n$ (the y values being queried) and T numbers in $\mathbb{F}$ (Bob's partial sums); this requires a total of $O(Tn \log |\mathbb{F}|)$ bits of communication. □

Theorem 4.11 can be leveraged to "transfer" the known separations among communication complexity classes to the algebrization setting. Though we didn't discuss this in lecture, we present the main theorem to illustrate the usefulness of this connection.

Theorem (Theorem 5.11)**.** *There exist $A, \tilde{A}$ such that*

(i) $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\tilde{A}}$

(ii) $\mathsf{coNP}^A \not\subset \mathsf{MA}^{\tilde{A}}$

(iii) $\mathsf{P}^{\mathsf{NP}^A} \not\subset \mathsf{PP}^{\tilde{A}}$

(iv) $\mathsf{NP}^A \not\subset \mathsf{BQP}^{\tilde{A}}$

(v) $\mathsf{BQP}^A \not\subset \mathsf{BPP}^{\tilde{A}}$

(vi) $\mathsf{QMA}^A \not\subset \mathsf{MA}^{\tilde{A}}$

5.2 Algebrization to Communication

Next, we show how the ideas of algebrization can be applied to get a somewhat surprising result in communication complexity.

First, we define a canonical hard problem in communication complexity.

Definition The *Inner Product* function IP takes as input two n-bit numbers x and y, and outputs the sum $\sum_i x_i y_i$.

Next, we define a variation of the standard communication model.

Definition A MA-*protocol* for a function $f(x, y)$ is one where Alice gets x, Bob gets y, and in the first step of the protocol, an all-powerful prover sends Alice and Bob a message. After this step, Alice and Bob may only communicate with each other. The *communication cost* of the protocol is the total number of bits exchanged, including the initial message from the prover.

In 2003, Klauck [6] found a $\Omega(\sqrt{n})$ lower bound for MA-protocols computing the Inner Product. Aaronson and Wigderson show that this bound is nearly optimal:

Theorem (Theorem 7.4)**.** *There is an* MA*-protocol for Inner Product using* $O(\sqrt{n}\log n)$ *bits of communication.*

Proof. To begin, we assume x and y represent the truth tables of functions $a, b : [\sqrt{n}] \times [\sqrt{n}] \to \{0,1\}$. The goal is then to compute the sum $\sum_{x,y} a(x,y)b(x,y)$. The protocol fixes some prime $q \in [n, 2n]$, and Alice and Bob extend a and b to polynomials $\tilde{a}, \tilde{b}$ over $\mathbb{F}_q[x,y]$ of degree at most $\sqrt{n}-1$ in each variable.

As the first step, the prover sends Alice $s(x)$, claiming $s(x) = \sum_y \tilde{a}(x,y)\tilde{b}(x,y)$. If this polynomial is correct, Alice can simply compute $\sum_x s(x)$ as the final answer. To verify that this polynomial is indeed correct, Bob chooses some random $r \in \mathbb{F}_q$, and sends Alice r and $\tilde{b}(r,y)$ for every y. Alice then checks if $s(r) = \sum_y \tilde{a}(r,y)\tilde{b}(r,y)$.

This protocol is clearly complete; for soundness, note that if the verifier sends an incorrect polynomial, the only way the check could fail is if Bob picks a root of the (non-zero) polynomial $s(x) - \sum_y \tilde{a}(x,y)\tilde{b}(x,y)$ which has degree at most $2\sqrt{n}$; this occurs with probability $\frac{2\sqrt{n}}{q} \leq \frac{1}{3}$, as $q \geq n$.

Finally, we compute the communication cost of this protocol. The prover sends to Alice a degree $\sqrt{n}$ polynomial with coefficients in $\mathbb{F}_q$; this requires $O(\sqrt{n}\log q) = O(\sqrt{n}\log n)$ bits of communication. Bob then sends Alice $\sqrt{n}+1$ numbers in $\mathbb{F}_q$, which again costs $O(\sqrt{n}\log n)$ bits. □

Finally, we state an interesting connection that arises between computational complexity classes as a result of the algebrizing results. In the statement below, an IP-protocol is similar to an MA-protocol, except Alice and Bob can talk to the prover multiple times during the protocol.

Lemma (Lemma 7.1)**.** *If* $f : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ *is in* NL*, then* f *has an* IP*-protocol with communication cost* $O(\text{polylog}\, N)$*.*

Theorem (Theorem 7.2)**.** *Suppose Alice and Bob hold 3-SAT instances, and that no* IP*-protocol with communication cost* $O(\text{polylog}\, n)$ *can decide whether their instances have a common satisfying assignment. Then* NL $\neq$ NP*.*

6 Conclusion

In this lecture, we have seen a new proof barrier: algebrization. We have seen just a small sample of the results in [1]; other highlights include showing that the separations PromiseMA $\not\subset$ SIZE(n^k) for all k from [9] algebrizes and that NP $\subseteq$ BPP does not algebrize, investigations of polynomial extensions over $\mathbb{Z}$, and proving that the zero-knowledge GMW result NP $\subseteq$ CZK [4] algebrizes. Aaronson and Wigderson also posed many new and interesting questions, including: what other

conjectured complexity class relationships algebrize or fail to algebrize; and can the definition of algebrization be made more robust (or elegant) by proving statements like $\mathsf{PSPACE}^{\tilde{A}} \subseteq \mathsf{IP}^{\tilde{A}}$?

Algebrization has generated much interest among researchers since its genesis. We give two notable connections with later work here. In 2009, Impagliazzo, Kabanets, and Kolokolova [5] gave an axiomatic approach to algebrization and derived the main theorems in [1]. In 2011, Williams' separation of NEXP and ACC [11] was in fact both non-naturalizing and non-algebrizing; this has understandably generated great interest in his program.

7 References

References

[1] S. Aaronson and A. Wigderson, *Algebrization: a new barrier in complexity theory*, in STOC, 2008, pp. 731–740.

[2] T. P. Baker, J. Gill, and R. Solovay, *Relativizatons of the p =? np question*, SIAM J. Comput., 4 (1975), pp. 431–442.

[3] H. Buhrman, L. Fortnow, and T. Thierauf, *Nonrelativizing separations*, in IEEE Conference on Computational Complexity, 1998, pp. 8–12.

[4] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems*, J. ACM, 38 (1991), pp. 691–729.

[5] R. Impagliazzo, V. Kabanets, and A. Kolokolova, *An axiomatic approach to algebrization*, in STOC, 2009, pp. 695–704.

[6] H. Klauck, *Rectangle size bounds and threshold covers in communication complexity*, in IEEE Conference on Computational Complexity, 2003, pp. 118–134.

[7] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, J. ACM, 39 (1992), pp. 859–868.

[8] A. A. Razborov and S. Rudich, *Natural proofs*, in STOC, 1994, pp. 204–213.

[9] R. Santhanam, *Circuit lower bounds for merlin-arthur classes*, in STOC, 2007, pp. 275–283.

[10] A. Shamir, *IP = PSPACE*, J. ACM, 39 (1992), pp. 869–877.

[11] R. Williams, *Non-uniform acc circuit lower bounds*, in IEEE Conference on Computational Complexity, 2011, pp. 115–125.