CS 2401 - Introduction to Complexity Theory

Lecture #8: Fall, 2015

Lecturer: Toniann Pitassi

Scribe Notes by: Noah Fleming

1 The Switching Lemma

The aim of this lecture is to show strong lower bounds against a AC^0 circuits. In particular, we are going to show that small AC^0 circuits cannot compute the parity function. This will follow by a depth reduction style argument, where we attempt to reduce the depth of the circuit level by level, until only a single layer remains. The key to this argument is a prominent tool in circuit complexity lower bounds known as the switching lemma, originally proven by Hastad [**?**]. The switching lemma, allows one to "switch" CNFs to DNFs and visa versa, by restricting the size of their underlying decision trees. This is done by applying a restriction to the variables so that the underlying decision tree of those CNFs or DNFs is not too deep.

Definition Let f_n be a boolean function with variables $X = \{x_1, \dots, x_n\}$. A *restriction* ρ to the variables of f is an assignment of values to the variables of f_n, $\rho : X \to \{0, 1, *\}$, where $*$ denotes that the variable is left unassigned. We denote $f_n \restriction \rho$ to be boolean function f_n after ρ has been applied.

We will let $\mathcal{R}^l$ denote the set of all restrictions leaving exactly l variables unassigned. That is, $\rho \in \mathcal{R}^l$ assigns exactly l $*$'s. Since we are setting exactly $n - l$ variables,

$$|\mathcal{R}^l| = \binom{n}{l} 2^{n-l}.$$

We will aim to show that after the application of a restriction, with high probability, the depth of the decision tree computing the restricted function is not too large.

Definition Let f_n be a boolean function. A *decision tree* T for f is binary tree with internal nodes labeled with the variables of f. A path from the root to a leaf corresponds to an assignment to the variables labeled at the nodes along the path, where a variable is given an assignment of 0 if the path proceeds to it's left child, and 1 if it it visits the right child. Each leaf is labeled 0 or 1 corresponding to the evaluation of f under the truth assignment defined by the path from the root to that leaf.

The *depth* of a decision tree is the length of the longest path from the root to a leaf.

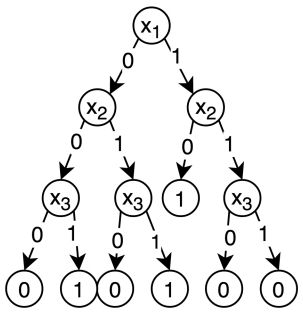
We say that a decision tree T represents a boolean function f if for any $x \in \{0, 1\}^n$, the leaf node at the end of the path in T defined by x is labeled with the value $f(x)$. In particular, we will be interested in the canonical decision tree representations of CNFs and DNFs.

Definition The *canonical decision tree* $T(f)$ for an r-DNF $f = t_1 \vee t_2 \vee \cdots \vee t_m$ is a decision tree constructed as follows:

1. Let t_1 be the first term in f and let $v_0 \wedge v_1 \wedge \cdots \wedge v_k$ be an ordered list of variables appearing in t_1. Create a depth k decision tree, where each node at level i is labeled with v_i.
2. Observe that each path from the root to a leaf l in the previous step defines an assignment (restriction) to the variables in that term ρ_l to the variables of f. For each leaf l, if $f \restriction \rho_l = 1$ or $f \restriction \rho_l = 0$, label the leaf with that constant. Otherwise attach the canonical decision tree for the restricted formula $T(f \restriction \rho_l)$ to that leaf.

The canonical decision tree for an r-CNF is constructed analogously.

For example, the canonical decision tree of the 2-DNF $f = x_1\bar{x_2} \vee \bar{x_1}x_3$ is:



There is a strong connection between the height of a decision tree for a formula and the size of it's corresponding DNF and CNF representations.

Proposition 1 *If a decision tree T representing a boolean function f has height r, then f can be written as both an r-CNF and an r-DNF.*

Proof We construct an r-DNF from T as follows. For each leaf l in T labeled 1, let π_l be the conjunction of the variables along the path from the root to l for which the path proceeds to that node's right child, and the negation of the variables for which the path proceeds to that node's left child. The corresponding DNF for f is then

$$DNF(f) = \bigvee_{l \text{ is labeled 1 in } T} \pi_l.$$

As T has height r, then each path can contain at most r variables; therefore $DNF(f)$ is an r-DNF. The construction of an r-CNF from T is similar.

The switching lemma gives an upper bound on the probability that, after the application of a random restriction to a formula, the height of the canonical decision tree remains large. We call restrictions where the restricted decision tree has too large depth "bad" for that function. Precisely, we say that a restriction ρ is *bad* for f if the canonical tree $T(f \restriction_\rho)$ has height greater than or equal to some constant s. This means that, for such a bad restriction ρ, $f \restriction_\rho$ cannot be converted into a s-DNF or s-CNF. Define the set of all bad restrictions for f,

$$Bad_f(l, s) = \{\rho \in \mathcal{R}_n^l | T(f \restriction_\rho), \text{ has height } \geq s\}.$$

Lemma 2 *(Hstad's Switching Lemma [1]) Let f be an r-DNF and s be a positive integer, then*

$$Pr_{\rho \sim \mathcal{R}_n^l}(T(f \restriction \rho) \text{ has height } \geq s) \leq (8pr)^s$$

where $\rho \sim \mathcal{R}_n^l$ denotes ρ being chosen uniformly at random from $\mathcal{R}_n^l$.

Proof We will begin by showing that to prove the lemma, it suffices to prove that

$$|Bad_f(l,s)| \leq |\mathcal{R}_n^{l-s}|(2r)^s.$$

Observe that the probability of choosing a bad restriction from the set of restrictions is

$$\frac{|Bad_f(l,s)|}{|\mathcal{R}_n^l|} \leq \frac{|\mathcal{R}_n^{l-s}|(2r)^s|}{|\mathcal{R}_n^l|} = \frac{\binom{n}{l-s}2^{n-l+s}}{\binom{n}{l}2^{n-l}}\Big(2r\Big)^s = \frac{n!(n-l)!l!(2r)^s 2^s}{(n-l+s)!(l-s)!n!} \leq \left(\frac{l}{n-l}\right)^s \Big(4r\Big)^s.$$

Letting $l = pn$ gives

$$\left(\frac{pn}{n-pn}\right)^s \Big(4r\Big)^s = \left(\frac{p}{1-p}\right)^s \Big(4r\Big)^s \leq (8pr)^s$$

for $p \leq 2/3$.

We now focus on proving that $|Bad_f(l,s)| \leq |\mathcal{R}_n^l|(2r)^s$. The proof will be by a coding argument by Razborov [7]. The idea is that to prove that some set is not very large, we create a one-to-one mapping into another set which is already known to be small. To do this, we will create a function which "encodes" bad restrictions as elements of $\mathcal{R}_n^{l-s}$ plus some extra information, and give a way to recover these "encoded" bad restrictions, proving that the map is indeed one-to-one.

Encoding

Fix some bad restriction $\rho \in Bad_f(l,s)$ to f, and let $t_1, \ldots, t_k$ be the terms remaining in $f \restriction_\rho$; fix an ordering on the variables in each of these terms. Because ρ is a bad restriction, the canonical decision tree $T(f \restriction_\rho)$ must have height at least s. Let π be the left most path in $T(f \restriction_\rho)$ which has height at least s. If necessary, truncate the path π so that it has length exactly s. We will think of π as a restriction.

We will segment π into $\pi_1, \ldots, \pi_q$ sub-restrictions. We define π_i to be the assignment to the variables queried along path π for term t_i, which were not queried for any of the previous terms t_j, $j < i$. That is, π_i defines the assignments along the path π which have not yet been assigned in $\pi \restriction_{\rho\pi_1 \ldots \pi_{i-1}}$. For each π_i, let σ_i be the unique restriction to the same variables considered in π_i which sets $t_i \restriction_{\rho\pi_1 \ldots \pi_{i-1}\sigma_i}$ to 1. The final assignment π_q may not set all of the variables in its corresponding term t_q (because we truncated the height of the path to be exactly s). To accommodate this, let σ_q be the restriction to the variables which restricts the same variables as π_q and agrees with the 1 assignment to t_q. Therefore, we have that $\pi = \pi_1 \pi_2 \ldots \pi_q$.

We now describe the extra bits we will need in order to retrieve the encoded bad restrictions. Let γ be this string of extra bits; we will construct γ as follows. For $1 \leq i \leq q$, let t_i be the term in $f \restriction_\rho$ from which we constructed π_i and σ_i. For each variable in t_i which π_i sets, append to γ the index of that variable and a bit representing whether π_i set that variable to 0 or 1. This requires

at most $|\sigma_i|(\log s + 1)$ bits per restriction, and because π restricts at most r variables, the total length of these extra bits is at most $r(\log s + 1)$.

Therefore our encoding function will be

$$\delta : \rho \to (\rho\sigma_1\sigma_2 \dots \sigma_q, \gamma).$$

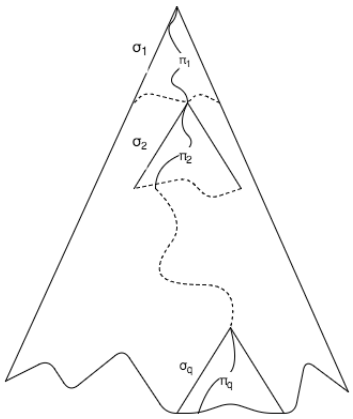


Figure 1: The canonical decision tree $T(f \restriction_\rho)$ labeled with $\pi_1 \dots \pi_q$ and $\sigma_1 \dots \sigma_q$.

Decoding

Our goal is to show that the function δ which we have constructed is one-to-one. To do this, it is enough to show that we can reconstruct any bad restriction ρ given it's encoding $\delta(\rho) = (\rho\sigma_1\sigma_2 \dots \sigma_q, \gamma)$ and the original boolean function f. To retrieve ρ perform the following process:

1. Apply the restriction $\rho\sigma_1\sigma_2 \dots \sigma_q$ to f. We can recover the term associated with σ_1 by noting that it will be the first term of $f \restriction_{\rho\sigma_1\sigma_2\dots\sigma_q}$ which is not set to 0. This is we have assumed that $f \restriction_\rho$ is not the constant 1 function, and each of the σ_i set their corresponding terms to 1; this term will be t_1.

2. Use the first $|\sigma_1| \log(r + 1)$ bits in the auxiliary bits δ in order to recover the restrictions σ_1 and π_1. We can determine which bits these are, by examining the variables appear t_1. We may then remove the restriction σ_1 in $f \restriction_{\rho\sigma_1\sigma_2\dots\sigma_q}$ by using the information about which variables were assigned by σ_1 in order to un-set them.

3. Repeat steps 1 through 3 with $f \restriction_{\rho\sigma_1\sigma_2\dots\sigma_q}$ replaced with $f \restriction_{\rho\sigma_2\sigma_3\dots\sigma_q}$, unless ρ has been fully decoded.

This process will allow us to recover $\pi_1, \dots, \pi_q$, $\sigma_1, \dots, \sigma_q$, and in particular the original restriction ρ. Observe that the restrictions $\rho\sigma_1 \dots \sigma_q$ belong to $\mathcal{R}_n^{l-s}$ because they restrict exactly $l - s$ variables. Therefore the encoding function δ which we have constructed is one-to-one.

2 AC^0 Circuit Lower Bounds

The switching lemma allows us to prove strong lower bounds on a class of circuits known as AC^0 circuits.

Definition The complexity class AC^0 is the class of all decision problems computable by families of circuits with constant depth and unbounded fanin $\vee$ and $\wedge$ gates, and $\neg$ gates occurring only at the leaves.

Such constant depth unbounded fanin circuits are known as AC^0 *circuits*. Taking any such constant depth circuit and duplicating the nodes with fanout greater than 1, we can assume that each node has fanout at most 1. Combining similar gates, and introducing dummy gates, we can assume that each constant depth circuit has the following form.

Proposition 3 *If C is a depth-d circuit of size M computing some function f, then C can be converted to a depth-d circuit such that*

- *All of the negations occur at the inputs.*
- *Every gate has fanout 1 except for the input gates.*
- *The circuit is composed of alternating levels of unbounded-fanin AND and OR gates.*
- *The number of gates is at most $(4M)^{2d}$.*

Note that because AC^0 circuits are of constant depth, applying proposition 3 incurs only a polynomial blowup in the number of gates. The switching lemma allows us to simplify a circuit by applying random restrictions to it, therefore in order to prove our desired AC^0 lower bound, we look for a function which is very robust, and therefore cannot be simplified in this way.

Definition The parity function $Parity_n$ takes as input n bits and computes their $\mod 2$ sum

$$Parity_n(x_1, \ldots, x_n) = (x_1 + \cdots + x_n) \mod 2.$$

It is simple to see that the parity function cannot be computed without knowing the value of all of its inputs. In particular, this implies the parity function requires a canonical decision tree of depth n, as it must query the value of each of the inputs in order to compute $Parity_n$.

By the switching lemma and proposition 1 we know that with high probability, under a random restriction, any small-width DNF can be simplified into a low-height decision tree, which in turn can be expressed as a small-width CNF, and visa versa. Therefore, the switching lemma allows us with high probability to "switch" a small-width DNF in to a small-width CNF. The key idea is to swap adjacent CNFs and DNFs and then merge the resulting adjacent similar gates. We will use this together with the robustness of the $Parity_n$ function in order to prove a contradiction.

Lemma 4 *Let f be a boolean function computed by a depth d AC^0 circuit with M gates. Let $\rho \in \mathcal{R}_n^p$ be a restriction with $p = \frac{1}{16s}$, then*

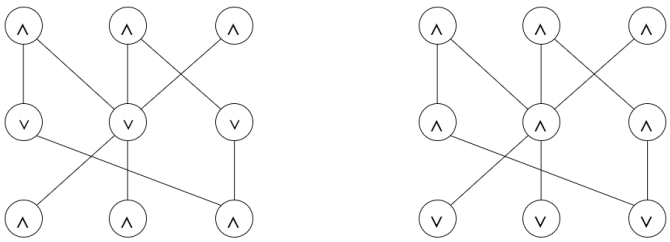
$$Pr[T(f \restriction_\rho) \text{ has height } > s] \leq M2^{-s}$$

Proof Let $\mathcal{C}$ be a depth d size M AC^0 circuit computing the function f. By the proposition 3, we can assume that the circuit is composed of alternating levels of $\vee$s and $\wedge$s. Without loss of generality, assume that the gates on the first level (next to the inputs) are $\vee$ gates; a symmetric argument to the one we will give holds for the case when they are $\wedge$ gates. We will think of each of these $\vee$ gates as a 1-DNF; that is, we will introduce a $\wedge$ gate in between the input variable and the corresponding $\vee$ gate. This only increases the depth by 1, and is necessary to ensure that the width of the DNFs to which we apply the switching lemma is bounded.

The goal is now to apply the switching lemma so that the probability of being unable to convert to each 1-DNF to an s-CNF is bounded above by 2^{-s}. To this end, we set the parameters of lemma 2 which we will be using for the remainder of this proof as follows; let

$$s = r = n^{1/d}/16,$$
$$p = \frac{l}{n} = \frac{1}{16r}.$$

By the switching lemma with these parameters, the probability that each of the 1-DNFs on the first level has a canonical decision tree of depth greater than s is at most 2^{-s}. If m_1 is the number of gates on level 1, then the probability that all of the 1-DNFs on this level, after a random restriction, have decision tree height greater than s is bounded above by $m_1 2^{-s}$ by the union bound. In particular, by the probabilistic method, this implies that there is a restriction under which all of the 1-DNFs on the first level have height at most s. Applying this restriction, we can transform each of the 1-DNFs in to an s-CNF. After doing so, we merge the resulting adjacent similar gates. An example of this can be seen in figure 2



(a) Before (b) After

Figure 2: The effect of an application of the switching lemma to a layer of DNFs. After application, the top two laters can be compacted into a single layer.

We apply the same argument for the s-CNFs at level 2, "switching" them with s-DNFs, and again reducing the depth. If we repeat this process $d-1$ times, this would result in a decision tree of height at most s for f. If each level i contains at m_i gates, then the probability that we are unable to convert to such a depth s decision tree would be bounded above by

$$(m_1 + m_2 + \cdots + m_{d-2})2^{-s} \leq M2^{-s}.$$

Any restriction of parity is either parity or its negation. As we have noted above, any decision tree requires the maximal number of queries to its variables in order to compute $Parity_n$. In order to obtain our lower bound, we will use this fact to show that if the size of our original circuit $\mathcal{C}$ computing parity was small, then under the random restrictions of lemma 4 we can obtain a small

depth decision tree computing a restriction parity on less than the number of remaining variables. As any restriction of $Parity_n$ is parity or its negation on fewer variables, then this would be a contradiction, and the original circuit $\mathcal{C}$ could not have computed $Parity_n$.

Theorem 5 *$Parity_n$ cannot be computed by AC^0 circuits of size less than $2^{\frac{n^{1/d}}{16}}$.*

Proof Suppose $\mathcal{C}$ is a size M depth d AC^0 circuit computing $Parity_n$. Moreover, assume $\mathcal{C}$ has the form of proposition 3. By corollary 4, there exists a restriction ρ such that $f \restriction \rho$ has a decision tree of height at most s with probability at least $1 - M2^{-s}$. In order to obtain this result, we want to set the parameters M and $s = r$ so that the number of remaining variables is more than the height s of the resulting decision tree after the restriction ρ.

In lemma 4, each random restriction leaves $l = pn$ variables unset. As we are applying $d - 1$ restrictions, the number of remaining variables is $p^{d-1}n$. In order to ensure that the number of remaining variables is greater than the height of the resulting decision tree, we need to set s so that

$$p^{d-1}n > s.$$

In corollary 4, we set $p = \frac{1}{16s}$. Substituting this in,

$$\frac{1}{(16s)^{d-1}}n > s,$$
$$\frac{n^{1/d}}{16^{1-1/d}} > s.$$

Therefore, we set $s = r = \frac{n^{1/d}}{16}$. In order to ensure that such a restriction exists. By corollary 4 we require that

$$M2^{-s} < 1,$$
$$M < 2^{\frac{n^{1/d}}{16}}.$$

If this holds, then there exists a restriction ρ to $\mathcal{C}$, such that the resulting canonical decision tree has height $s < p^{d-1}n$. Therefore, as any restriction of $Parity_n$ is parity or its negation, by assumption this restricted circuit would be computing parity on $p^{d-1}n$ input variables. But as the canonical decision tree queries only at most s variables, less than the total number of the variables, this must be a contradiction because the parity function depends on all of its variables. Therefore, the original circuit cannot $\mathcal{C}$ cannot be computing $Parity_n$, and so we require that $M \geq 2^{\frac{n^{1/d}}{16}}$.

By a slight adjustment of the parameters, we can obtain an even stronger result. Namely that no small AC^0 circuit can correlate with $Parity_n$ with probability more than one half plus an inverse exponential.

Theorem 6 *Let $\mathcal{C}$ be a size M depth d AC^0 circuit on n variables, $M \leq 2^{n^{1/d}/32}$. Then*

$$Pr_{x \sim \{0,1\}^n}[\mathcal{C}_n(x) = Parity_n(x)] \leq \frac{1}{2} + 2^{-s/2},$$

Proof To begin, observe that any function f which can be computed by a decision tree of height $s < n$ has correlation

$$Pr_{x\sim\{0,1\}^n}[f(x) = Parity_n(x)] = \frac{1}{2}.$$

This is because any path of the decision tree can query at most $s < n$ variables. We know that in order to correctly compute parity, we must query all n of the variables. To any path of the decision tree, there are an equal number of extensions to that path which set the function to even and odd.

Suppose that

$$Pr_{x\sim\{0,1\}^n}[\mathcal{C}_n(x) = Parity_n(x)] > \frac{1}{2} + 2^{-s/2}.$$

The idea of the proof will be to use the above fact, and to set our parameters in such a way that when applying a restriction to $\mathcal{C}$ using lemma 4, a large fraction of the restrictions will result in a height $s < n$ decision tree. We will use this in order to upper bound the correlation of the circuit $\mathcal{C}$ with parity function marginalized over a set of restrictions, and show that this contradicts our initial assumption.

To this end, set $M2^{-s} \leq 2^{-s/2}$, as we want a large fraction of restrictions to restrict the tree to height $< s$. therefore, we require that $M \leq 2^{s/2} = 2^{n^{1/d}/32}$. We know that after an application of any restriction ρ given by lemma 4 $p^{d-1}n$ variables will be left left unset. Marginalizing over the restrictions which set $k = 1 - p^{d-1}n$ variables,

$$\begin{aligned} Pr_{x\sim\{0,1\}^n}[\mathcal{C}_n(x) = Parity_n(x)] &= \mathop{\mathbb{E}}_{\rho\in\{0,1\}^k}[Pr[f\restriction_\rho(x) = Parity\restriction_\rho(x)]] \\ &= \sum_{\rho\in\{0,1\}^k} Pr[f(x) = Parity_n(x) | (x_{i1},\ldots,x_{ik}) = \rho] \Pr_{\rho\in\{0,1\}^k}[\rho] \\ &= \sum_{\rho\in\{0,1\}^k} Pr[f\restriction_\rho(x) = Parity_n\restriction_\rho(x)]\frac{1}{2^k}. \end{aligned} \quad (1)$$

Where $x_{i1},\ldots,x_{ik}$ are the k variables set by the restriction ρ. Now because we set $M2^{-s} \leq 2^{-s/2}$, at least a $1 - 2^{-s/2}$ fraction of the 2^k restrictions will result in a decision tree of height at most s. Using this fact, and upper bounding the remaining probabilities by 1, we can rewrite equation 1 as

$$\begin{aligned} \sum_{\rho\in\{0,1\}^k} Pr[f\restriction_\rho(x) = Parity_n\restriction_\rho(x)]\frac{1}{2^k} &= \left[2^k(1-2^{-s/2})(1/2) + 2^k(2^{-s/2})(1)\right]\frac{1}{2^k}, \\ &= \frac{1}{2} + 2^{-s/2-1}. \end{aligned}$$

This contradicts our original assumption, as

$$\frac{1}{2} + 2^{-s/2-1} < \frac{1}{2} + 2^{-s/2}.$$

3 Applications of the AC^0 Parity Lower Bound

- **Fourier Concentration** The Fourier expansion of a boolean function is it's representation as a low-degree polynomial. Using Hastad's switching lemma 2, Linial, Mansour, and Nisan [4] prove that the Fourier expansion of any function in AC^0, has all of it's larger coefficients concentrated on it's low order Fourier coefficients; AC^0 functions can be approximated by low degree polynomials.

- **Pseudo-Random Generators for AC^0** Derandomization studies the possibility of removing or reducing the amount of randomization used by randomized algorithms while still maintaining their efficiency and correctness. Nisan and Wigderson [5] use the correlation lower bound, theorem 6 above, to show that the randomized analogues of AC^0, RAC^0 and $BPAC^0$, can be de-randomized in poly-logarithmic space and quasi-polynomial time.

- **AC^0-Circuit SAT and #-SAT Algorithms** In lecture 2 we studied the Circuit Satisfiability problem. We can consider a restricted domain of this problem and define the AC^0-Circuit SAT problem; given an AC^0 circuit, determine whether there exists an input x which evaluates that circuit to 1. In [2] Impagliazzo, Matthew and Paturi show that Hastad's switching lemma 2 can be used in order to give a non trivial algorithm for AC^0-Circuit SAT. Their technique also implies an algorithm for the much harder problem $\#AC^0$-SAT of determining the number of inputs x which satisfy a given AC^0 circuit.

- **Compression Algorithms for AC^0** Given the truth table representation of a boolean function f which can be computed by some unknown small circuit of a known circuit class, the compression problem is to find, in time 2^{cn} for $c > 0$, a circuit computing f of size less than the trivial size $2^n/n$ circuit. Using a modification of the switching lemma 2, Kabanets and Kolokolova [3] are able to obtain such a compression algorithm for AC^0 circuits. Moreover, they show that their compression results can be extended to get non-trivial #SAT algorithms. These #SAT algorithms agree with the AC^0-#SAT algorithms [2] mentioned above.

- **AC^0-Frege Lower Bounds** *Proof Complexity* studies the lengths of proofs of propositional tautologies. Originally proposed as an approach to separating NP from coNP, proof complexity formalizes the structure of these propositional proofs in terms of *proof systems* which specifies the rules of inference that we allow in a proof. AC^0-Frege is a proof system in which we each line of the proof can be expressed as an AC^0 circuit. Pitassi, Beame and Impagliazzo [6] prove exponential lower bounds for AC^0-Frege by proving an analogous switching lemma in the space of bipartite matchings. Urquhart and Fu [8] give a simplified proof of this result.

References

[1] J. Hastad, *Computational limitations for small depth circuits*, PhD thesis, MIT, 1986.

[2] R. Impagliazzo, W. Matthews, and R. Paturi, *A satisfiability algorithm for ac0*, in Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, SIAM, 2012, pp. 961–972.

[3] V. Kabanets and A. Kolokolova, *Compression of boolean functions*, Electronic Colloquium on Computational Complexity (ECCC), 20 (2013), p. 24.

[4] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, fourier transform, and learnability*, J. ACM, 40 (1993), pp. 607–620.

[5] N. Nisan and A. Wigderson, *Hardness vs randomness*, Journal of Computer and System Sciences, 49 (1994), pp. 149 – 167.

[6] T. Pitassi, P. Beame, and R. Impagliazzo, *Exponential lower bounds for the pigeonhole principle*, Computational Complexity, 3 (1993), pp. 97–140.

[7] A. A. Razborov, *Bounded arithmetic and lower bounds in boolean complexity*, in Feasible Mathematics II, Birkhauser, 1993, pp. 344–386.

[8] A. Urquhart and X. Fu, *Simplified lower bounds for propositional proofs*, Tech. Rep. TR-293-95, University of Toronto (CA), 1995.