Notes on Complexity Theory Last updated: May, 2015

Lecture on Håstad's Switching Lemma

Jonathan Katz

1 Introduction and Background

We have already seen an "algebraic" approach to proving that computing parity requires exponential size AC^0 circuits. Here we give a more combinatorial proof. Besides being interesting as a different technique, it also gives a better lower bound on the size of AC^0 circuits needed to compute parity.

Recall that AC^0 is the set of languages/problems decided by *constant-depth*, polynomial-size circuits (with gates of unbounded fan-in). We consider the basis consisting of AND, OR, and NOT gates, though we do not count NOT gates when measuring the depth or size of the circuit.

A *DNF formula* on n variables is a disjunction of terms, each of which is a conjunction of literals. E.g.,

$$f(x_1, \ldots, x_n) = (x_1 \wedge \bar{x}_2) \vee (x_7 \wedge \bar{x}_8 \wedge \bar{x}_{11})$$

is a DNF formula. Analogously, a *CNF formula* is a conjunction of terms, each of which is a disjunction of literals. The *size* of a DNF/CNF formula is the number of terms, and its *width* is the maximum number of literals in any term.

A *decision tree* is a directed, acyclic graph with a designated start vertex having in-degree 0. Each vertex other than the leaves has out-degree two. Each non-leaf vertex is labeled with a variable, and has one outgoing edge labeled '0' and one outgoing edge labeled '1'. Each leaf vertex is labeled either '0' or '1'. A decision tree computes a function in the natural way. The *depth* of a decision tree is the maximum path length from the start to a leaf, and its *size* is the number of leaves. For a function f, we write $\mathrm{DT}_{\mathrm{depth}}(f)$ to denote the smallest depth of any decision tree computing f. Note that any function on n variables always has $\mathrm{DT}_{\mathrm{depth}}(f) \leq n$.

2 The Switching Lemma

Let $f : \{0,1\}^n \to \{0,1\}$ be a function on n variables. An *s-restriction* α of f fixes $n - s$ of the variables to 0 or 1, and leaves the remaining s variables "free." We write $f|_\alpha$ for the reduced function obtained. Note that if α is an s-restriction, then $f|_\alpha$ is a function from $\{0,1\}^s$ to $\{0,1\}$. When we say that we pick a uniform s-restriction, we mean that we choose a random subset of $n - s$ variables and fix each of those variables uniformly to 0 or 1.

The switching lemma states, roughly, that if f can be computed by a small-width DNF formula, then a random restriction of f (with small number of free variables) can likely be computed with a small-depth decision tree. Formally:

Theorem 1 *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a DNF formula of width at most w. Let α be a random s-restriction with $s = \sigma n \leq n/5$. Then for any $d \geq 0$,*

$$\Pr_\alpha\left[DT_{depth}(f|_\alpha) > d\right] \leq (10\sigma w)^d.$$

Proof Fix some d and let $\mathcal{B}$ be the set of "bad" s-restrictions, i.e., those s-restrictions β for which $\mathrm{DT}_{\mathrm{depth}}(f|_\beta) > d$. We show that each bad restriction can be encoded using a "small" number of bits, and thus $\mathcal{B}$ itself is "small" (at least in comparison to the set of all possible s-restrictions).

Let f be computed by the DNF formula $T_1 \vee \cdots \vee T_\ell$, where each T_i contains at most w literals. Restriction α *kills* a term T_i if it sets one of the literals in T_i to 0. (In that case, that term will be removed from the DNF formula for $f|_\alpha$.) We say that α *fixes* a term T_i if it sets all the literals in T_i to 1. (In that case, T_i—and hence the entire formula—becomes equal to the constant 1.) If $\beta \in \mathcal{B}$ is a bad restriction, then we know that β does not fix any of the terms of f, nor does it kill all the terms of f—if it did, then $f|_\beta$ would have a constant-depth decision tree.

Given some bad restriction β, we define a canonical decision tree for $f|_\beta$ as follows. Take the first term T_{i_1} not killed by β, and say it has d_1 free variables. Form the complete, depth-d_1 decision tree over those variables, considering the variables in order. The (unique) 1-leaf of that sub-tree becomes a 1-leaf in the canonical decision tree. For each of the 0-leaves of that sub-tree, continue the process by fixing the variables according to the path to that leaf (thus defining a new restriction β'), taking the first term not killed by β', etc. (If what remains is the constant function, then that leaf simply becomes a leaf in the canonical decision tree.)

Since β is bad, the canonical decision tree for $f|_\beta$ must, in particular, have depth greater than d. Take a path of length exceeding d, and let P be the first d steps on that path. Fix the d variables involved in P to their values taken on that path. Call the resulting $(s-d)$-restriction (which consists of β plus d additional fixed variables) π.

Say P traverses sub-trees associated with clauses $T_{i_1}, T_{i_2}, \ldots, T_{i_\ell}$ involving $d_1, d_2, \ldots$ free variables, with π possibly ending in the middle of T_{i_ℓ}. We encode β via an $(s-d)$-restriction γ plus some auxiliary information. We determine γ in the following way: start with β. Then fix the d_1 variables in T_{i_1} to the unique values such that T_{i_1} is fixed, the d_2 variables in T_{i_2} to the unique values such that T_{i_2} is fixed, etc. (We stress that γ does *not* correspond to π, since π does *not* fix $T_{i_1}, \ldots$.) As auxiliary information, we include for each clause (1) which variables in the clause are the ones being fixed in each iteration, and (2) how those variables are set in π. The first of these can be encoded using $d_i \cdot \lceil \log(w+1) \rceil \le d_i \log w + d_i$ bits (we use an alphabet of size $w+1$ that can encode each possible position plus a special "termination" character), and the second can be encoded using d_i bits.

We show that this encoding allows recovery of β, given f. Given the encoding, we find the first clause T_{i_1} of f that is fixed under γ. The auxiliary information tells us what variables in that clause were fixed when extending β, as well as how those variables are set in π. This process is continued until we get a list of all variables that were set in forming γ (equivalently, π), thus allowing us to recover the original restriction β.

How many bits did we use to encode β? We encoded β using an $(s-d)$-restriction γ plus $d \log w + 2d$ additional bits. So the total number of bad restrictions is at most $\binom{n}{s-d} \cdot 2^{n-s+d} \cdot (4w)^d$, and the *fraction* of bad restrictions is at most

$$\begin{aligned}
\frac{\binom{n}{s-d} \cdot 2^{n-s+d} \cdot (4w)^d}{\binom{n}{s} \cdot 2^{n-s}} &\le \left(\frac{s}{n-s+d}\right)^d (8w)^d \\
&\le \left(\frac{\sigma}{1-\sigma}\right)^d \cdot (8w)^d \le (10\sigma w)^d,
\end{aligned}$$

using $\sigma < 1/5$. ■

Note that a similar proof applies when f is computed by a CNF formula. For our application in the next section, it will be useful to rephrase the switching lemma based on the following observation.

Lemma 2 *If $DT_{depth}(f) \leq d$, then f has a width-d DNF formula and a width-d CNF formula.*

Proof Given a depth-d decision tree for f, we get a width-d DNF for f by simply taking the disjunction of all the paths leading to 1-leaves.

Since f has a depth-d decision tree, so does $\neg f$. Hence $\neg f$ has a width-d DNF. But then f has a width-d CNF by applying de Morgan's law. ■

Corollary 3 *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a DNF formula (resp., CNF formula) of width at most w. Let α be a random s-restriction with $s \leq n/5$. Then $f|_\alpha$ can be computed by a CNF formula (resp., DNF formula) of width w except with probability at most $(10sw/n)^w$.*

3 A Lower Bound for Parity

We use the switching lemma to derive a lower bound for the size of AC^0 circuits computing parity. We rely on the following easy lemma.

Lemma 4 *Any DNF (resp., CNF) formula computing parity (or its negation) on n-bit inputs must have width n.*

Proof We focus on DNF formulae; the case of CNF formulae is handled similarly. Say there is a term T with fewer than n literals. Set the values of the variables so T evaluates to 1, and hence the DNF formula evaluates to 1 regardless of how the remaining variables are set. But toggling the value of any variable not in T should toggle the value of the function, a contradiction. ■

Theorem 5 *For sufficiently large n, any depth-d circuit that computes parity on n-bit inputs must have size at least $2^{\Omega(n^{1/(d-1)})}$.*

Proof Say we have a depth-d circuit of size S computing parity. We will assume the following about this circuit:

- NOT gates are only at the inputs.
- The circuit if *layered*, with gates at one layer feeding only into the next layer, and all gates at a layer having the same type.
- Each gate has fanout 1. (The inputs can have unbounded fanout.)

The above are without loss of generality here, in the sense that any circuit of depth d and size S can be converted to an equivalent circuit of the above form without increasing in the depth and with size increasing to $O((dS)^d)$. Since d here is a constant, this does not affect the theorem statement.

Let $w = 20 \log S$. Say for concreteness that the inputs feed into AND gates at the top level. (The case of OR gates can be handled analogously.) We claim that we may assume without loss of generality that every gate at the top level has fanin at most w. The reason is that, if not, we can apply a random restriction in which each variable is fixed with probability $c = 2 - \sqrt{2} \approx 0.6$ (and, if so, is fixed to 0 or 1 with half probability each); one can show that, with positive probability, the resulting circuit has no gates at the top level with fanin greater than w, and at least $n/4$ variables remain free. Note that the resulting restricted function computes parity or its negation. Since the number of variables is reduced by only a constant factor, this does not affect the theorem statement.

Having dispensed with various technicalities, we now come to the heart of the proof. Set $n_0 = n$ and let $n_i = n_{i-1}/20w$ for $i = 1, \ldots, d-2$. The gates at the second layer of the circuit each compute a DNF formula of width at most w. Focusing on any particular such gate, and applying Corollary 3 with an n_1-restriction, we see that the output of that gate (after the restriction) can be computed by a width-w CNF formula except with probability at most $(10n_1w/n_0)^w = 2^{-20\log S} \ll 1/S$. Since there are at most S gates, a union bound shows that there exists an n_1-restriction for which all level-2 Choosing any such restriction, the DNF sub-circuits can then be swapped for width-w CNF sub-circuits. But then the AND gates at levels 2 and 3 can then be coalesced, reducing the depth by 1. Note that the restricted function still computes parity or its negation, now on n_1-bit inputs.

Continuing, we repeatedly apply Corollary 3 for $i = 2, \ldots, d-2$. This gives a width-w CNF or DNF formula computing parity on n_{d-2}-bit inputs, where $n_{d-2} = n \cdot (1/20w)^{d-2} = n/(400 \log S)^{d-2}$. But then, using Lemma 4, we must have $w = n_{d-2}$. This completes the proof. ■

Bibliographic Notes

The general proof strategy used here is due to Furst, Saxe, and Sipser, with the bound claimed here due to Håstad [1]. The simplified proof of the switching lemma given here is due to Razborov. Our presentation is based on the notes by O'Donnell [2].

References

[1] J. Håstad. Computational Limitations of Small-Depth Circuits. MIT Press, 1987. (Published version of the author's PhD thesis.)

[2] R. O'Donnell. The Switching Lemma. *Lecture 14 of 15-855 Intensive Intro to Complexity Theory*, Spring 2009. `http://www.cs.cmu.edu/~odonnell/complexity`