Approximating AC^0 by Small Height Decision Trees and a Deterministic Algorithm for $\#\mathsf{AC}^0SAT$

Paul Beame[*]
Institute for Advanced Study
and the University of Washington
beame@cs.washington.edu

Russell Impagliazzo[†]
Institute for Advanced Study
and the University of California, San Diego
russell@cs.ucsd.edu

Srikanth Srinivasan[‡]
DIMACS, Rutgers University
srikanth@dimacs.rutgers.edu

December 8, 2011

Abstract

We show how to approximate any function in AC^0 by decision trees of much smaller height than its number of variables. More precisely, we show that any function in n variables computable by an unbounded fan-in circuit of AND, OR, and NOT gates that has size S and depth d can be approximated by a decision tree of height $n - \beta n$ to within error $\exp(-\beta n)$, where $\beta = \beta(S, d) = 2^{-O(d \log^{4/5} S)}$. Our proof is constructive and we use its constructivity to derive a *deterministic* algorithm for $\#\mathsf{AC}^0SAT$ with multiplicative factor savings over the naive $2^n S$ algorithm of $2^{-\Omega(\beta n)}$, when applied to any n-input AC^0 circuit of size S and depth d. Indeed, in the same running time we can deterministically construct a decision tree of size at most $2^{n-\beta n}$ that exactly computes the function given by such a circuit. Recently, Impagliazzo, Matthews, and Paturi derived an algorithm for $\#\mathsf{AC}^0SAT$ with greater savings over the naive algorithm but their algorithm is only randomized rather than deterministic.

The main technical result we prove to show the above is that for every family $\mathcal{F}$ of k-DNF formulas in n variables and every $1 < C = C(n) \leq \log^{poly(k)} |\mathcal{F}|$, one can construct a distribution on restrictions that each set at most n/C variables such that, except with probability at most $2^{-n/(2^{O(k)} C \log |\mathcal{F}|)}$, after application of the restriction, all formulas in F simultaneously reduce to $\log^{poly(k)} |\mathcal{F}|$-juntas where an s-junta is a function whose value depends on only s of its inputs. Previously, Ajtai showed simultaneous approximations for k-DNF formulas by juntas related to the one we show but with a dependence on $exp(k)$ rather than $poly(k)$, resulting in a weaker height-approximation tradeoff than ours.

[*] Research supported by NSF grants CCF-0830626 and DMS-0835373 and by the Ellentuck Fund
[†] Research supported by NSF grants DMS-0835373, CNS-0716790, CCF-0832797, etc.
[‡] Research supported by NSF grants CCF-0832797 and DMS-0835373.

1 Introduction

Williams has given a formal connection between circuit lower bounds for a circuit class and improved algorithms for the satisfiability of circuits from that class ([8, 9]). In light of this work, it is natural to consider using lower bound techniques to develop improved Satisfiability algorithms. One of the greatest success stories for circuit lower bounds are those for the class AC^0 of constant-depth, unbounded fan-in, Boolean circuits. A sequence of papers [1, 3, 10, 4] has proved strong lower bounds for computing natural functions like Parity with such circuits.

Santhanam ([7]) has given a template for such improved algorithms. Many of the circuit lower bounds are based on showing that the circuit can be simulated or approximated by another type of representation, such as a decision tree or a low-degree polynomial. In particular, for many of these representations, Satisfiability is trivial or at least relatively easy. Santhanam used a decision tree representation to give an improved algorithm for the satisfiability of formulas over the De Morgan basis. One can also think of Williams' Satisfiability algorithm for ACC^0 ([9]) as following this template, with the representation being low-degree polynomials.

This raises the question of what size of each representation is required for different classes of circuits. Interestingly, the question of size of decision tree needed to represent AC^0 circuits was considered early in the study of this class. Ajtai, in his original paper [1] which, independently of Furst, Saxe and Sipser [3], showed that Parity is not in AC^0, also showed that for any $\epsilon > 0$ and for sufficiently large n, AC^0 circuits have a correlation at most $2^{-n^{1-\epsilon}}$ with parity. The main tool used in the proof of this correlation bound is a result on approximating AC^0 circuits by decision trees that are much less than full height. More precisely, Ajtai shows that the value of the AC^0 circuit is exactly that of the decision tree on all but an exponentially small fraction of branches of the tree. Until very recently, this gave the best known such correlation bound for polynomial-size constant depth circuits.

Decision trees are a very natural, well-studied model of computation, with applications in many areas of Computer Science, including machine learning, proof complexity, circuit lower bounds, and the general study of the combinatorics of Boolean functions. In many cases, it is desirable to have a decision tree representation of a given boolean function since many properties of boolean functions that are hard to verify for Boolean functions represented by other means become easy to check in the decision tree model. Thus, this question of the minimal decision tree representation of classes of circuits is interesting in its own right.

Here, we revisit Ajtai's approximation technique, simplifying and strengthening it considerably. We then show how to use it to give a deterministic improved algorithm for Satisfiability of constant-depth circuits. We can also improve Ajtai's correlation bound.

In simultanenous recent work [6], Impagliazzo, Matthews and Paturi use a related representation, as a disjoint union of sub-cubes where the function is constant, to devise a zero-error randomized algorithm for $\#\mathsf{AC}^0 SAT$. The running time of this algorithm is considerably better than ours, but it seems inherently probabilistic. Their representations also give a tight correlation bound for approximating parity in AC^0. Hastad [5] also proved this correlation bound using similar techniques.

Like the simpler arguments in [3, 1], Ajtai's decision tree construction is based on iteratively converting the sub-circuits in the circuit to k-juntas (functions that depend on only k variables) and hence k-DNF formulas for constant k. However, instead of choosing a random set R of variables that are then queried obliviously, Ajtai's construction chooses the variables to query adaptively based on how setting a constant number of variables simultaneously simplifies the sets of k-DNF formulas that describe these sub-circuits. In particular Ajtai shows that when k is $O(1)$ one can choose a decision tree T of height $n/\log^{O(1)} n$ so that at all but $2^{-n/\log^{O(1)} n}$ fraction of leaves of T all formulas in a given polynomial-size collection of k-DNF formulas reduce to $\log^{O(1)} n$-juntas. Such a statement, which involves setting only a minority of variables, would not be possible using random restriction over a fixed set. In Ajtai's construction, the constant in the exponent of $\log n$ in the conversion of sets of k-DNF formulas to $\log^{O(1)} n$-juntas grow exponentially in k.

In this paper, we follow the same basic strategy as Ajtai, but derive a much stronger version of such a construction in which the exponents depend only polynomially on k. Our key improvement over Ajtai's results is a much sharper and simpler lemma showing that with exponentially small failure probability one can find small hitting sets for the sets of k-terms in collections of k-DNF formulas after setting only a small fraction of variables. Moreover, our proof is substantially simpler.

We apply our new construction to show that any AC^0 circuit can be approximated in error-free manner by a decision tree of height $n - n/2^{O(\log^{4/5} n)}$ that produces an output value on all but a $2^{-n/2^{O(\log^{4/5} n)}}$ fraction of leaves. Our proofs are sufficiently constructive that they yield a deterministic algorithm running in time $2^{n-n/2^{O(\log^{4/5} n)}} n^{O(1)}$ that, given an n-input AC^0 circuit C, produces a decision tree that *exactly* computes the value of C. This immediately yields a deterministic $2^{n-n/2^{O(\log^{4/5} n)}} n^{O(1)}$ time algorithm that exactly counts the number of satisfying assignments of any AC^0 circuit, a time savings of $2^{n/2^{O(\log^{4/5} n)}}$ over brute force.

We state our main theorem formally below:

Theorem 1.1. *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by an unbounded fan-in circuit of size $S = n^c$ and depth d where $4c^{4/5}d \le \log_2^{1/5} n$. Then*

(a) there is a decision tree T_f of height $n - n/2^{2d\log_2^{4/5} S}$ such that for x chosen uniformly at random from $\{0,1\}^n$, $\Pr[f(x) \neq T_f(x)] \le \exp(-n/2^{2d\log_2^{4/5} S})$.

(b) there is a decision tree T'_f of size $2^{n+1-n/2^{2d\log_2^{4/5} S}}$ that exactly computes f.

Moreover, both trees T_f and T'_f can be constructed from the circuit for f by an algorithm in time $2^{n-n/2^{2d\log_2^{4/5} S}} S^{O(1)}$.

2 Preliminaries

For a set of formulas $\mathcal{F}$, we use $|\mathcal{F}|$ to denote its size and $||\mathcal{F}||$ to denote the total number of literal occurrences in $\mathcal{F}$. Fix n Boolean variables $x_1, \ldots, x_n$. A *restriction* on these variables is a function $\pi : \{x_1, \ldots, x_n\} \to \{0, 1, *\}$. Intuitively, for $b \in \{0,1\}$, π sets the values of variables in $\pi^{-1}(b)$ to b and leaves the variables in $\pi^{-1}(*)$ unset.

Recall that a *decision tree* T on n Boolean variables $x_1, \ldots, x_n$ is a full binary tree with internal nodes labelled by the variables, the two out-edges of each internal node labelled 0 and 1, and leaves labelled by Boolean values. The tree T defines a Boolean function in a natural way: we start at the root and at each internal node, we query the variable x labelling the internal node and follow the edge corresponding to the value of x. When a leaf is reached, we simply output the Boolean value that labels this leaf. The *height* of T is the length of the longest root-to-leaf path in T and the *size* of T is the number of leaves in T.

A *restriction decision tree* T on n Boolean variables $x_1, \ldots, x_n$ is a full binary tree with internal nodes labelled by the variables, and the two out-edges of an internal node labelled 0 and 1. (In contrast to the case of a decision tree, we don't require the leaves to output Boolean values.) Such a restriction defines a natural probability distribution $\mathcal{R}_T$ on restrictions: Choose a random root-to-leaf path in the tree by starting at the root and choosing a random child at each node; set the variables according to the answers on this path. (A path of length ℓ is chosen with probability $2^{-\ell}$.) The height and size of a restriction decision tree are defined similarly to that of decision trees.

An important family of restrictions in the context of AC^0 is the family of random restrictions $\mathcal{R}_{\ell,n}$, where we choose a random $R \subset [n]$ of size ℓ and set the values of all variables outside R uniformly at random. Note that this may be seen as choosing the random set R and then applying the tree restriction corresponding to the complete decision tree on the variables outside R.

For a set S of input indices, a Boolean function or formula is an *S-junta* iff its value depends only on inputs indexed by S. The S is called its *deciding set*. It is an *s-junta* iff it is an S-junta for some set S with $|S| = s$. We will produce deciding sets for formulas by combining hitting sets for their terms. A set H of variables is a *hitting set* for a collection of terms of a DNF formula iff it contains at least one variable from each term.

Proposition 2.1 (Azuma-Hoeffding)**.** *Let $X_1 \ldots X_i \ldots X_N$ be a difference sub-martingale with $|X_i| \leq M$ for all i. Then* $\Pr[\sum_{i=1}^{N} X_i \geq K] \leq exp(-K^2/(2M^2N))$.

Fix integers $k, n \in \mathbb{N}$ such that $k \leq n$. Recall that a *k-wise independent space* over $\{0,1\}^n$ is a multiset S with elements from $\{0,1\}^n$ such that for any subset $A = \{i_1, i_2, \ldots, i_k\}$ of $[n]$ and any Boolean values $b_1, \ldots, b_k$, we have $\Pr_{x \in_R S}[x_{i_1} = b_1 \wedge \cdots \wedge x_{i_k} = b_k] = \frac{1}{2^k}$. We will need efficient explicit constructions of k-wise independent sample spaces. We use the construction due to Alon, Babai, and Itai [2].

Lemma 2.2. *For any n, k such that $k \leq n$, there is a k-wise independent space S over $\{0,1\}^n$ with $|S| = O(n^{k/2})$. Moreover, S can be constructed by a deterministic algorithm running in time $n^{O(k)}$.*

3 Simultaneous Simplification for Families of k-DNF formulas

In this section we prove our main technical result, namely that given any family $\mathcal{F}$ of k-DNF formulas, we can construct a restriction decision tree $T_{\mathcal{F}}$ of height corresponding to only a small fraction of number of variables using time $|T_{\mathcal{F}}| \cdot ||\mathcal{F}||^{O(1)}$, such that, except with exponentially small probability, the restriction given by a random root-leaf path in $T_{\mathcal{F}}$ reduces every formula in $\mathcal{F}$ to a small junta, a function with a small set of variables that determines its value.

Usually, arguments for random restrictions apply the probabilistic method by showing that a restriction chosen from a simple distribution given *a priori* will work, because with positive probability it satisfies all the conditions required to simplify a given family of formulas. Instead, we build our distribution of restrictions constructively as a restriction decision tree based on the detailed properties of the family of formulas. Though we will find it convenient to express aspects of our construction using probabilities over paths in the restriction decision tree, there is actually no randomness required in its construction.

The high-level structure of our argument is similar to that of [1]. In section 3.1 we first show constructively how to produce a restriction decision tree for which the restrictions given by all but an exponentially small fraction of all root-leaf paths "reduce" each formula either by satisfying it or by yielding a small hitting set for its k-terms. The small hitting set property is useful for simplification because, for each choice of values for the variables in such a hitting set, the remaining formula would be reduced from a k-DNF to a $(k-1)$-DNF. Indeed, one can assign the variables in such a hitting set and take the union with all the small juntas found for the recursive application of the hitting set property to each of the resulting families of $(k-1)$-DNFs in order to obtain a small junta for each formula, an approach taken in [3] and the simple argument in [1]. Instead, in section 3.2 we use a more sophisticated argument from [1] which combines all the different $(k-1)$-DNF formulas produced from the k-DNF formulas into a single larger family of $(k-1)$-DNF formulas that we can recursively convert into small juntas use the union of all of these to produce the final junta.

3.1 Finding Small Hitting Sets for the k-Terms in Families of k-DNF Formulas

We first state out main lemma on hitting sets which shows how to find a restriction decision tree for a family of k-DNF formulas that on almost all of its branches yields either satisfying assignments or small hitting sets for the k-terms of all the formulas in the family.

Lemma 3.1. *Let $C \geq 1$ be such that $C \log_2 m$ is a non-negative integer multiple of 2^{-k}. If $\mathcal{F}$ is a set of k-DNF formulas on $\{0,1\}^n$ with $|\mathcal{F}| < m$, then there is a restriction decision tree $T_{\mathcal{F}}$ on $\{0,1\}^n$ of height n/C such that for π chosen according to a random root-leaf path in $T_{\mathcal{F}}$, the probability that for all formulas $F \in \mathcal{F}$, either $F|_\pi = 1$ or the k-terms of $F|_\pi$ have a hitting set of size at most $h(k,C,m) = 2^{k^2+5k+3}k^{3k}(C \log_2 m)^k$ is at least $1 - p(k,C,m,n)$ where $p(k,C,m,n) = k \log_2 m \exp(-2^{-2k-3}k^{-2}n/(C \log_2 m))$. Moreover, there is an algorithm with running time $2^{n/C}||\mathcal{F}||^{O(1)}$ that constructs $T_{\mathcal{F}}$ given $\mathcal{F}$ as input.*

The tree $T_{\mathcal{F}}$ in this lemma is constructed in stages. We will think of a formula as being successfully handled if it is either satisfied or has a small hitting set for its k-terms. In each stage we make sure that at almost all leaves of the restriction decision tree we will have successfully handled at least half the formulas in the family that were not handled at the previous stages. Since there are fewer than m formulas in the family, it will take at most $\log_2 m$ stages to handle all formulas. The construction for each stage is given by the lemma below; this is the heart of the argument.

Lemma 3.2. *Let k be a non-negative integer and $C \geq 1$ be an integer multiple of 2^{-k}. If $\mathcal{F}$ is a set of k-DNF formulas on $\{0,1\}^n$, then there is a restriction decision tree $T_{\mathcal{F}}$ on $\{0,1\}^n$ of height n/C such that for π chosen according to a random root-leaf path in $T_{\mathcal{F}}$, the probability that for fewer than $|\mathcal{F}|/2$ formulas $F \in \mathcal{F}$, $F|_\pi = 1$ or the k-terms of $F|_\pi$ have a hitting set of size at most $2^{k^2+5k+3}k^{3k}C^k$, is at most $k \exp(-2^{-2k-3}k^{-2}n/C)$. Moreover, there is an algorithm with running time $2^{n/C}||\mathcal{F}||^{O(1)}$ that constructs $T_{\mathcal{F}}$ given $\mathcal{F}$ as input.*

Before we prove this lemma we apply it in the obvious way to prove our main lemma.

Proof of Lemma 3.1. Repeatedly apply Lemma 3.2 with $C \log_2 m$ in place of C, replacing $\mathcal{F}$ at each leaf by the set of formulas $\mathcal{F}' \subset \mathcal{F}|_\pi$ that are not satisfied and do not have small hitting sets for their k-terms. Each application reduces $|\mathcal{F}|$ by at least a factor of 2. After $\log_2 m$ steps, all formulas will have been removed and at most n/C variables will have been assigned. The failure probability is a union bound over the steps. The constructivity follows from that in Lemma 3.2. □

The proof of Lemma 3.2 is quite subtle, though the basic ideas are relatively simple. Any k-DNF formula either has a small set of variables that hits all its k-terms or a large number of disjoint k-terms and in the latter case, it has many different ways of being satisfied. Indeed, this idea together with random restrictions that set most variables is how the argument in [3] and the simple argument in [1] both work. In order to set only a minority of variables we have to balance choices of assignments that satisfy one formula against the terms (and hence potential satisfying assignments) that they remove from other formulas.

In order to make this argument work, we need to orchestrate a careful march towards satisfaction (or small hitting sets) for all the formulas. To do so, we maintain a collection of disjoint, partially satisfied k-terms for each formula in the family. If all members of a collection of variables occurs frequently enough among these partially satisfied k-terms then setting those variables makes substantial progress towards satisfying many formulas (along at least some of its branches). There is also a cost incurred in doing so because any term in which these variables are set incorrectly is removed from the disjoint collection for that formula.

We therefore apply a form of potential argument that keeps track of the number of partially satisfied k-terms and how far they are from satisfaction. In order to argue that for a high fraction of branches, the wins will dominate the losses, we need to maintain a bound on the sizes of the potential losses and hence we need to also upper bound the number of occurrences of any particular collection of variables among these terms.

Intuitively, as terms get closer to being satisfied, their "likelihood" of being eventually satisfied increases exponentially. We handle this by having the threshold for the frequency required to choose a collection of variables depend exponentially on the distance from satisfaction that setting them would produce. (In general, setting some variables might create terms of many lengths but we only count the benefit from one particular length.) If one of a formula's k-terms does become satisfied then all of its other partially satisfied k-terms are useless and must be removed from our counts, hence we "freeze" the formula by limiting the terms we take into account to a number at which we can safely bet that the formula will eventually be satisfied. We now give the proof.

Proof of Lemma 3.2. Let $\mathcal{F} = \{F_1, \dots, F_m\}$. We can assume without loss of generality that each F_i consists only of exactly k-terms, since if either condition holds for the pure k-term part of F_i then it holds for F_i. Let $V = 2^{k+5}k^3C$, let $C_j = V^{j+1}$ for $j = 0, \dots, k-1$, and let $D_j = 12kC_{j-1}$ for $j = 0, \dots, k-1$.

Though the tree $T_{\mathcal{F}}$ will depend deterministically on the set $\mathcal{F}$, we find it convenient to describe $T_{\mathcal{F}}$ by defining how we follow a random path in $T_{\mathcal{F}}$.

At any point t in the process, for each unsatisfied F_i, and each $1 \leq j \leq k$, we will have a set $S^t_{i,j}$ of j-terms and a restriction π_t given by the assignment along the path so far. We begin with $S^0_{i,k}$ being a maximal set of disjoint k-terms for F_i, and all other $S^0_{i,j}$ empty. π_0 will be the empty restriction. We will maintain the invariants that, for any i, the terms in the union of the $S^t_{i,j}$ are pairwise disjoint; that, for $j < k$, $|S^t_{i,j}| \leq D_j$; and that $S^t_{i,k}$ is a maximal set of k-terms from $F_i|_{\pi_t}$ that are disjoint from each other and from the other $S^t_{i,j}$. (Hence each variable appears in at most one term in each $S^t_{i,j}$.)

At each step, we determine the variables to branch on and how to maintain the sets $S^t_{i,j}$ by repeating the following sequence of steps until the process terminates.

Branching: For any $\ell \geq 1$ and $\{x_1, \dots, x_\ell\}$ a set of variables, let the *potential j-terms* for $\{x_1, \dots, x_\ell\}$ be the set of terms in the various $S^t_{i,j+\ell}$ containing all of $x_1, \dots, x_\ell$ and exactly j variables not among $x_1, \dots, x_\ell$. In other words, these are all terms of size $j + \ell$ that could become size j by restricting $x_1, \dots, x_\ell$. Find the smallest $j > 0$ so that there is a set of $1 \leq \ell \leq k - j$ variables $\{x_1, \dots, x_\ell\}$ so that the number of potential j-terms for $\{x_1, \dots, x_\ell\}$ in the $S^t_{i,j+\ell}$ is at least $C_j m/n$. If there is no such j, terminate. Otherwise, add branches that assign $x_1, \dots, x_\ell$ in all possible ways, choose a random assignment σ_t to $x_1, \dots, x_\ell$ and follow the branch corresponding to σ_t. Hence $\pi_{t+1} = \pi_t\sigma_t$.

We now describe how to build the set of $S^{t+1}_{i,j}$ based on the various $S_{i,j} = S^t_{i,j}$.

Restricting: For any term T in $S_{i,j+\ell}$ that contains all of $x_1, \dots, x_\ell$ such that $T' = T|_{\sigma_t} \neq 0$ add T' to $S_{i,j}$. For any term T in $S_{i,j'}$ containing any of $x_1, \dots, x_\ell$, remove T from $S_{i,j'}$.

Clean-up For any i, j with $|S_{i,j}| = D_j$, set all $S_{i,j'}$ for all $j' > j$ to $\emptyset$. Call such an *i frozen* at level j. In particular, if any term becomes satisfied, F_i is frozen at level 0, and all $S_{i,j}$ with $j > 0$ are empty. (Since after i is frozen at level j, we will never create any new terms of size j or larger, by Claim 3.3 below, this maintains the invariant that $|S^{t'}_{i,j}| \leq D_j$ for all t'.)

Replacement: For any i that is not frozen at any level $j < k$, while there is a term of size k in $F_i|_{\pi_{t+1}}$ disjoint from all terms in $S_{i,j}$ for $1 \leq j \leq k$, add it to $S_{i,k}$.

Define each $S^{t+1}_{i,j}$ to be the resulting set $S_{i,j}$ and increment t.

The above process repeats until it terminates. However, in our construction, we will prune all branches of length greater than n/C. We need to show that it terminates before this bound with high probability and we need to show that at termination at least half the formulas are either satisfied or have small hitting sets.

Claim 3.3. *For any t, i, and $j < k$, $|S^{t+1}_{i,j}| \le |S^t_{i,j}| + 1$.*

Proof. New terms of length $j < k$ can only be created in the Restricting step and this only happens if there is a term in $S^t_{i,j+\ell}$ that contains all of $x_1, \dots, x_\ell$. Since the terms in the sets $S^t_{i,j+\ell}$ are disjoint from each other, so at most one term can contain all of $x_1, \dots, x_\ell$. (Indeed, since all terms in $S^t_{i,j'}$ for various $j' < k$ are disjoint, for each fixed i and t there is at most one term created in all the $S^{t+1}_{i,j}$.) □

Claim 3.4. *For any step t and any formula F_i that is not frozen at any level $j < k$ at step t, the set of variables in $\bigcup_{j=1}^{k} S^t_{i,j}$ is a hitting set for the set of k-terms of $F_i|_{\pi_t}$.*

Proof. This is true initially by the maximality of $S^0_{i,k}$. By the replacement rule, for $t > 0$ the set $S^t_{i,k}$ is a maximal disjoint set of k-terms in $F_i|_{\pi_t}$ that is disjoint from $\bigcup_{j=1}^{k-1} S^t_{i,j}$ and hence the union of the variables in all of these sets of terms intersects every k-term of $F_i|_{\pi_t}$. □

Claim 3.5. *For each t, each $1 \le j < k$ and $\ell \le k - 1 - j$, no variables $x_1, \dots, x_\ell$ appear together in a term in $S^t_{i,j+\ell}$ for more than $2C_jm/n$ formulas F_i.*

Proof. We prove this by induction on the number of branching steps. Since we only consider $j + \ell < k$, at the start, all such $S^0_{i,j+\ell}$'s are empty. Assume that the claim is true for j and $x_1, \dots, x_\ell$ at the beginning of step $t + 1$. Note that, in creating $S^{t+1}_{i,j+\ell}$, we only add terms to any $S^t_{i,j+\ell}$ if we are creating terms at level $j + \ell$, branching on some $y_1, \dots, y_{\ell'}$, $\ell' \le k - j - \ell$. Because such a step did not branch on $x_1, \dots, x_\ell$, there were fewer than C_jm/n terms of size $j + \ell$ containing $x_1, \dots, x_\ell$ in all of the $S^t_{i,j+\ell}$. Therefore, if at the end of step $t + 1$, we have $2C_jm/n$ terms of size $j + \ell$ including $x_1, \dots, x_\ell$, more than C_jm/n of these must have also included all of $y_1, \dots, y_{\ell'}$ before restriction. That is, $x_1, \dots, x_\ell, y_1, \dots, y_{\ell'}$ appeared together in terms in more than C_jm/n sets $S^t_{i,j+\ell+\ell'}$. But then we would have created terms of size j rather than $j + \ell$. From this contradiction, we still have at most $2C_jm/n$ terms including $x_1, \dots, x_\ell$ among all the sets $S^{t+1}_{i,j+\ell}$ corresponding to all the formulas F_i at the end of step $t + 1$. The lemma thus follows by induction. □

Let $A_{i,j} = |\bigcup_t S^t_{i,j}|$ be the number of terms that are ever in $S^t_{i,j}$ throughout the process, and let $A_j = \sum_i A_{i,j}$.

Claim 3.6. *For $0 \le j \le k - 1$, $A_j \le (1 + 1/(6k))D_jm$.*

Proof. Let t^* be the index of the final step. Consider any term that is ever in some $S^t_{i,j}$. This term is either (1) in $S^{t^*}_{i,j}$ when the process terminates, (2) in $S^t_{i,j}$ the first time step t that F_i is frozen at some level $j' < j$, or (3) is removed from $S^t_{i,j}$ in some step t where it intersects with the branching variables $x_1, \dots, x_\ell$. Since we have $|S^t_{i,j}| \le D_j$, for any formula F_i and time step t, its total contribution to types (1) or (2) is at most D_j, the type of its contribution depending on whether it is frozen at some level $j' < j$ at termination. Since, at any time t, each branching variable can be in at most $2C_{j-1}m/n$ terms in some $S^t_{i,j}$, and there are at most n such variables, the number of terms of type (3) is at most $2C_{j-1}m$. So the total is at most $D_jm + 2C_{j-1}m \le (1 + 1/(6k))D_jm$ because $D_j = 12kC_{j-1}$. □

Claim 3.7. *For any j with $0 < j < k$, any run has at most $m/(4k)$ formulas frozen at level j and not at any level $j' < j$.*

Proof. If $f_j m$ formulas are frozen at level j, but not at any smaller level, they each had D_j terms in $S^t_{i,j}$ at the time t that they were frozen, for a total of $D_j f_j m$ such terms. Going through the categories again, none of these terms were removed for being frozen at smaller levels, by definition. At the end of the process, there must be fewer than $C_{j-1}m/j$ terms in all of the $S^{t^*}_{i,j}$, since otherwise there would be a variable that would appear in at least $C_{j-1}m/n$ such $S^{t^*}_{i,j}$ and we would branch on it. Finally, as above, at most $2C_{j-1}m$ of these terms are removed from some $S^t_{i,j}$ because they include a branching variable. Thus $(C_{j-1}/j + 2C_{j-1})m \geq f_j D_j m$ and hence $f_j \leq 3C_{j-1}/D_j \leq 1/(4k)$. □

Claim 3.8. *Any run ends with at least $m/2$ formulas either satisfied or with a hitting set for its terms of size at most $4\sum_{j=1}^{k} C_{j-1}$.*

Proof. At the last step t of the process, no variable appears in more than $C_{j-1}m/n$ terms in some $S^t_{i,j}$, or else we would branch on that variable. Thus, at least $3m/4$ of the formulas $F_i|_{\pi_t}$ have a total number of variables in $\bigcup_{j=1}^{k} S^t_{i,j}$ at most $4\sum_{j=1}^{k} C_{j-1}$. By Claim 3.4, if F_i is not frozen at any level, the variables in $\bigcup_{j=1}^{k} S^t_{i,j}$ form a hitting set for $F_i|_{\pi_t}$. By Claim 3.7, at most $m/4$ formulas end frozen at some level between 1 and $k-1$. So $3m/4 - m/4 \geq m/2$ formulas either have small hitting sets or are frozen at level 0, which is the same as being satisfied. □

So when the process terminates, we either have a constant fraction of formulas set to 1 or with small covers. Finally, we need to show that the process terminates in small height with high probability.

Let B_j be the number of steps where we create terms of size j.

Claim 3.9. *The probability that $B_j \geq n/(k^2C)$ is at most* $\exp(-2^{-2k-3}k^{-2}n/C)$.

Proof. Every time we create terms of size j, we expect to create at least $2^{-k}C_j m/n$ such terms. (More precisely, if we branch on ℓ variables we expect to create at least $2^{-\ell}C_j m/n$ such terms.) We have no matching upper limit, but we can artificially restrict our attention to any subset of exactly $C_j m/n$ potential terms. Let Y_i be the number of terms created from among these potential terms in the i-th step in which we create terms of size j. Let $X_i = 2^{-k}C_j m/n - Y_i$. Then the X_i form a difference sub-martingale ($\mathbb{E}[X_i|X_1 \cdots X_{i-1}] \leq 0$) and each $|X_i| \leq C_j m/n$, so by the Azuma-Hoeffding inequality with $M = C_j m/n$, for any fixed N such that $0 \leq N \leq B_j$, setting $K = 2^{-k-1}C_j mN/n = 2^{-k-1}MN$ we have $\Pr[\sum_{i=1}^{N} X_i \geq K] \leq \exp(-K^2/(2M^2N) = \exp(-2^{-2k-3}N)$. By our definitions, $\sum_{i=1}^{N} Y_i \leq K$ implies that $\sum_{i=1}^{N} X_i \geq K$ so we have $\Pr[\sum_{i=1}^{N} Y_i \leq K] \leq \exp(-2^{-2k-3}N)$. Assume that $B_j \geq N = n/(k^2C)$. In this case $K = 2^{-k-1}MN = 2^{k-1}k^{-2}C_j m/C$ and the probability that at most K terms of size j are created is at most $\exp(-2^{-2k-3}k^{-2}n/C)$. We claim that this event must indeed happen on any branch. For $j = 0$, at most m such 0-terms can be created on any branch, but since $C_0 = V = 2^{k+5}k^3C$ we have $K = 16km$ which is much larger than m. For $1 \leq j \leq k-1$ we have $K = 2^{k-1}k^{-2}C_j m/C = 16kC_{j-1}m = \frac{4}{3}D_j m$ and by Claim 3.6 at most $\frac{7}{6}D_j m < \frac{4}{3}D_j m$ terms of size j can have been created during the first N such steps. □

Corollary 3.10. *With all but probability* $k\exp(-2^{-2k-3}k^{-2}n/C))$. *the number of variables queried is less than n/C.*

Proof. The height is at most k times the number of branching steps. Each branching step contributes to B_j for some $0 \leq j \leq k-1$. Each branching step sets at most k variables. The bound then follows by Claim3.9. □

Finally, we can complete the proof of Lemma 3.2. By Claim 3.8 at least $m/2$ formulas in $\mathcal{F}|_\pi$ are either satisfied or have hitting sets of size at most $4\sum_{j=1}^{k} C_{j-1} = 4\sum j = 1^k V^j < 8V^k = 2^{k^2+5k+3}k^{3k}C^k$. Pruning all branches of the tree at level n/C gives the claimed restriction decision tree and the time to construct it is immediate from the process we used to define it. □

3.2 Converting k-DNF Families to Small Juntas

Using our substantially improved construction to achieve small hitting sets our argument now follows the same lines as the argument in Ajtai's original paper, though we make small changes to improve its constructivity. As noted earlier, the key idea is a recursive construction in which we use the hitting sets for the k-terms to produce a much larger family of $(k-1)$-DNF formulas for which we can recursively find small deciding sets that can be combined with the hitting sets to produce a small deciding set for each formula in the family.

Theorem 3.11. *Let $\mathcal{F}$ be a set of k-DNF formulas on $\{0,1\}^n$ with $|\mathcal{F}| < m$ for $m \geq n$, let $C > 1$, and let $s = 2^{k^4/2}k^{6k^3}(C\log_2 m)^{k^3/2}$. Then there is a restriction decision tree $T_\mathcal{F}$ on $\{0,1\}^n$ of height n/C such that for π chosen according to a random root-leaf path in $T_\mathcal{F}$, the probability that for all formulas $F \in \mathcal{F}$, $F|_\pi$ is not an s-junta is at most $4k\log_2 m\exp(-2^{-2k-4}k^{-3}n/(C\log_2 m))$.*

Moreover, there is an algorithm with running time $2^{n/C}||\mathcal{F}||^{O(1)}$ that constructs $T_\mathcal{F}$ given $\mathcal{F}$ as input and computes for each formula F and successful root-leaf path π, a set $S = S_{F,\pi}$ of size s such that $F|_\pi$ is an S-junta.

Proof. The proof is by induction on k. The base case $k = 0$ is immediate since all such formulas are satisfied and hence do not depend on any inputs. In the inductive step we apply Lemma 3.1 with C replaced by kC to find a tree $T_\mathcal{F}$ so that for almost all restrictions π defined by $T_\mathcal{F}$, every formula $F \in \mathcal{F}$ either has $F|_\pi = 1$ or F has a hitting set for its k-terms of size at most $h = h(k, kC, m) = 2^{k^2+5k+3}k^{3k}(kC\log_2 m)^k$. For each such good branch π, define the set of formulas $\mathcal{F}^\pi$ as follows:

For each $F \in \mathcal{F}$ such that $F|_\pi \neq 1$, let $H_{F,\pi}$ be the hitting set given by Lemma 3.1. Let $A_{F,\pi}$ be the set of partial assignments σ defined on $H_{F,\pi}$ with $|\sigma| = k-1$. Observe that $|A_{F,\pi}| = \binom{h}{k-1}2^{k-1}$. For each $\sigma \in A_{F,\pi}$ define the formula F_π^σ to be the $(k-1)$-DNF formula whose terms are all $T|_\sigma$ such that T is a term of $F|_\pi$ and $T|_\sigma$ does not have any variables in $H_{F,\pi}$. $\mathcal{F}^\pi$ is the set of all such F_π^σ. The fact that F_π^σ is a $(k-1)$-DNF formula follows since every k-term of $F|_\pi$ has at least one variable in $H_{F,\pi}$. We now apply the inductive hypothesis with C replaced by $kC/(k-1)$ to $\mathcal{F}^\pi$ which has at most $m\binom{h}{k-1}2^{k-1}$ formulas and obtain that for almost all extensions τ of π of , each of the formulas in $\mathcal{F}^\pi$ is an s'-junta for some s' given by the inductive statement for $k-1$ with m replaced by $m\binom{h}{k-1}2^{k-1}$. Assume that τ is such a good extension. For each formula F_π^σ, let $S_{\sigma,\tau}$ be the set of size s' such that $F_\pi^\sigma|_\tau$ is an $S_{\sigma,\tau}$-junta.

Claim 3.12. *For $S = H_{F,\pi} \cup \bigcup_{\sigma\in A_{F,\pi}} S_{\sigma,\tau}$ the formula $F|_\tau$ is an S-junta.*

Proof. To see that the claim holds, fix an assignment ρ to S that is consistent with τ. If $F|_\tau$ is not falsified by ρ then there is some term T of $F|_\pi$ that is consistent with ρ and τ. If the variables of T are entirely contained in $H_{F,\pi}$ then ρ must satisfy T. It follows that ρ satisfies $F|_\pi$ and hence it also satisfies $F|_\tau$. Otherwise, T has at most $k-1$ variables in $H_{F,\pi}$. Let σ be any assignment to exactly $k-1$ variables of $H_{F,\pi}$ that contains the variables in T and is consistent with ρ. Then $T|_\sigma$ is a (non-zero) term in F_π^σ and hence ρ does not falsify $(F_\pi^\sigma)|_\tau$. Since $(F_\pi^\sigma)|_\tau$ is an $S_{\sigma,\tau}$-junta and ρ assigns values to all of $S_{\sigma,\tau}$, ρ must satisfy $(F_\pi^\sigma)|_\tau$. Therefore there is some term T' of $F|_\pi$ such that $(T'|_\sigma)|_\tau$ is satisfied by ρ. Since ρ extends σ, ρ also satisfies $T'|_\tau$. Therefore ρ satisfies $F|_\tau$. It follows that in either case ρ forces $F|_\tau$ to a constant. □

It remains to work out the parameters. For $j = k, \ldots, 1$ define h_k and m_k recursively by $m_k = m$, let $h_j = h(j, kC, m_j) = 2^{j^2+5j+3} j^{3j} (kC \log_2 m_j)^j$ where $h_j = h(j, kC, m_j)$ is the upper bound in Lemma 3.1 on the size of the hitting set for the j-terms at most leaves of the tree of height $n/(kC)$ for sets $\mathcal{F}$ of j-DNF formulas with $|\mathcal{F}| < m_j$. and $m_{j-1} = \binom{h_j}{j-1} 2^{j-1} * m_j \leq (2h_J)^{j-1} m_j$. Let $p_j = p(j, kC, m_j, n) = j \log_2 m_j \exp(-2^{-2j-3} j^{-2} n/(kC \log_2 m_j))$ be the failure probability in that lemma. For $j = 0, \ldots, k$, define s_j by $s_0 = 0$ and $s_j = h_j + \binom{h_j}{j-1} 2^{j-1} s_{j-1}$. Then, by our construction, the probability that for a random path π in $T_{\mathcal{F}}$ there is a formula F in $\mathcal{F}$ such that $F|_\pi$ is not an s_k-junta is at most $\sum_{j=1}^k p_j$.

If $(2h_j)^{j-1} < n \leq m \leq m_j$ then $m_{j-1} \leq m_j^2$ and hence

$$h_{j-1} \leq 2^{(j-1)^2+5(j-1)+3} (j-1)^{3(j-1)} (2kC \log_2 m_j)^{j-1} < h_j/16.$$

Therefore $(2h_{j-1})^{j-2} < n$ and hence inductively the h_j are bounded by a geometric series with largest term h_k. In particular, this means that

$$s_k \leq (2h_k)^{\sum_{j=1}^k (j-1)} = (2h_k)^{\binom{k}{2}} \leq 2^{k^4/2} k^{6k^3} (C \log_2 m)^{k^3/2} < n^{1/k}.$$

Also, by construction, for every j, $m_j \leq (2h_k)^{\binom{k}{2} - \binom{j}{2}} m < nm$ Hence $\log_2 m_j < 2logm$ and therefore $p_j \leq 2j \log_2 m \exp(-2^{-2j-4} j^{-2} n/(kC \log_2 m))$. Hence the total failure probability $\sum_{j=1}^k p_j \leq 2p_k \leq 4k \log_2 m \exp(-2^{-2k-4} k^{-3} n/(C \log_2 m))$. The constructivity follows from that in Lemma 3.1, the fact that the sets $S_{F,\pi}$ are easily defined as the tree construction proceeds and, for each π and σ, the formulas $\{F_\sigma^\pi \mid F \in \mathcal{F}\}$ are constructible in $||\mathcal{F}||^{O(1)}$ time given $\mathcal{F}$. □

4 A Deterministic Algorithm for $\#\mathsf{AC}^0 SAT$

In this section we derive our algorithm for $\#\mathsf{AC}^0 SAT$ by deterministically constructing a decision tree of size much less than 2^n that exactly computes the function given by the input AC^0 circuit and then simply walking through that decision tree to compute the number of satisfying assignments. We produce that decision tree by iteratively combining restriction decision trees constructed in the previous section for families of formulas corresponding to levels in the AC^0 circuit. The combination of these trees will only determine the value of the circuit on almost all branches. On any branch where the value is not determined so far, we simply add a complete decision tree for assignments on that branch. This will be a rare enough occurrence that it will not add much to the total size of the tree.

The basic idea in all restriction arguments for AC^0 bounds is to find a simple representation for the input-level subcircuits and then to propagate that simplification to make the next level of the circuit amenable to the same simple representation. Unfortunately, if we apply Lemma 3.11 iteratively, if we start with a collection of m k-DNF formulas and use the junta property for the next level of the circuit, the resulting formula is only an s-junta for $s = \log^{poly(k)} m$ which is much larger than k, and immediately would be too large to iterate. For this iterative application we need to apply a further restriction which will set most of the variables. This restriction will be chosen pseudorandomly which is sufficiently constructive for our needs. The argument is given in the next subsection.

4.1 Reducing the Junta Size

Lemma 4.1. *Let $\mathcal{F}$ be a set of k-DNF formulas on $\{0,1\}^n$ with $|\mathcal{F}| < m$ and suppose that $k \leq k' \leq (\log_2 m)^{1/5}$. Then there is a restriction decision tree $T_{\mathcal{F}}$ on $\{0,1\}^n$ of height $n - m^{-2/k'} n$ such that for π*

chosen according to a random root-leaf path in $T_{\mathcal{F}}$, the probability that for some formula $F \in \mathcal{F}$, $F|_\pi$ is not a k'-junta is at most $4k \log_2 m \exp(-2^{-2k-5}k^{-3}n/\log_2 m)$. Moreover, there is an algorithm with running time $2^{n-m^{-2/k'}n}n^{O(k')}||\mathcal{F}||^{O(1)}$ that constructs $T_{\mathcal{F}}$ given $\mathcal{F}$ as input.

Proof. We apply Theorem 3.11 with $C = 2$ to get a restriction decision tree $T'_{\mathcal{F}}$ of height $n/2$ for which on at most a $4k \log_2 m \exp(-2^{-2k-5}k^{-3}n/\log_2 m)$ fraction of root-leaf paths π some formula $F|_\pi$ is not an s-junta for $s = 2^{k^4/2}k^{6k^3}(2\log_2 m)^{k^3/2}$. Let $n' = m^{-1/k'}n/(2s)$. For each π such that all formulas become s-juntas, we will choose a fixed set $R = R(\pi)$ of n' variables and add a complete restriction decision tree on the variables not in R. Consider choosing R k'-wise independently and uniformly among all sets of size n' on the remaining $n/2$ variables. For a fixed set S of size s, $\Pr[|R \cap S| > k'] < \binom{s}{k'}(2n'/n)^{k'}/ \leq \left(\frac{2n's}{n}\right)^{k'} < 1/m$. Since this is less than $1/|\mathcal{F}|$ we can fix a set R such that for all formulas F in $\mathcal{F}$, R does not intersect any of the sets $S = S_{F,\pi}$ such that $F|_\pi$ is an $S_{F,\pi}$-junta in more than k elements.

Since $k \leq k' \leq (\log_2 m)^{1/5}$ we have $2^{k^4} \leq 2^{(k')^4} \leq m^{1/k'}$, and hence $2s < 2^{k^4/2}(\log_2 m)^{2k^3} \leq m^{-1/k'}$ and so $n' = m^{-1/k'}n/(2s) \leq m^{-2/k'}n$.

For constructivity, we begin with the constructivity of Theorem 3.11. We then observe that since, by Lemma 2.2, there are explicit k'-wise independent probability distributions on n-bit strings with support only $n^{O(k')}$ so we simply try all possibilities in the family. In order to test whether or not a given R succeeds we use each of the sets $S_{F,\pi}$ such that $F|_\pi$ is an $S_{F,\pi}$-junta, given by the algorithm in Theorem 3.11. For each leaf π, checking that R succeeds amounts to checking that $|R \cap S_{F,\pi}| \leq k'$ for each such $F \in \mathcal{F}$ which can be done in time $||\mathcal{F}||^{O(1)}$. □

4.2 Converting AC^0 circuits to Decision Trees and the $\#\mathsf{AC}^0 SAT$ Algorithm

We now apply Lemma 4.1 iteratively to derive both small height approximations and small size exact representations of AC^0 as decision trees. The constructivity of both representations is an essential feature.

Proof sketch of Theorem 1.1. Without loss of generality we can assume that the circuit has alternating layers of $\vee$ gates and $\neg$-gates. The bottom level of the circuit is a 1-DNF. The basic idea of the argument is to apply Lemma 4.1 iteratively with $k = k'(\log_2 m)^{1/5} = c^{1/5}\log_2^{1/5} n$ and $m = S = n^c$ to the set of formulas that approximate a given level. When all those formulas become k-juntas they can be negated and plugged into the $\vee$ gate at the next level to produce the new family of k-DNFs. The trees are built by appending the new trees at each leaf of the tree for the previous level. The rest is working out parameters, and the full details are given in the appendix. □

We note that since any incomplete branch in a decision tree has no correlation with Parity, from Theorem 1.1 (a), we immediately obtain that any AC^0-circuit of size S and depth d has correlation at most $2^{-n/2^{2d\log_2^{4/5} S}}$ with Parity, though this is a much weaker bound than that shown in [6, 5].

Our main consequence of Theorem 1.1 is the following algorithm for $\#\mathsf{AC}^0 SAT$.

Corollary 4.2. *Given an unbounded fan-in circuit C on n inputs of size $S = n^c$ and depth d where $4c^{4/5}d \leq \log_2^{1/5} n$, we can errorlessly exactly compute $|C^{-1}(1)|$ by an algorithm in time $2^{n-n/2^{2d\log_2^{4/5} S}}S^{O(1)}$.*

Proof. Given a decision tree T we can visit the leaves of T to compute the size of $T^{-1}(1)$ in time proportional to the size of T. The corollary is then immediate from Theorem 1.1 (b). □

References

[1] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[2] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, December 1986.

[3] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[4] J. Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, Berkeley, CA, May 1986.

[5] J. Håstad. On parity. Manuscript., 2011.

[6] R. Impagliazzo, M. Matthews, and R. Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, Kyoto, Japan, January 2012.

[7] R. Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, pages 183–192, Las Vegas, NV, October 2010. IEEE.

[8] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, pages 231–240, Cambridge, Ma, June 2010.

[9] R. Williams. Non-uniform ACC circuit lower bounds. In *Proceedings Twenty-Sixth Annual IEEE Conference on Computational Complexity*, pages 115–125, San Jose, CA, June 2011.

[10] A. C. Yao. Separating the polynomial hierarchy by oracles: Part I. In *26th Annual Symposium on Foundations of Computer Science*, pages 1–10, Portland, OR, October 1985. IEEE.

A Proof of Theorem 1.1

We give the full details of the proof sketched in Section 4:

Proof of Theorem 1.1. Without loss of generality the circuit consists of d layers, each of which consists of alternating $\neg$ gates and unbounded fan-in $\vee$ gates. We write $S = n^c$ and iteratively apply Lemma 4.1 with $m = S$ and $k = k' = (\log_2 m)^{1/5} = c^{1/5}(\log_2 n)^{1/5}$ to sets of k-DNF formulas that approximate a given level of $\vee$-gates in the circuit where we begin the levels starting with the inputs. Observe that each gate at the first level of $\vee$ gates is given by a 1-DNF formula. Lemma 4.1 shows that there is a restriction decision tree T_1 of height $n - n/S^{2/k}$ such that for almost all root-leaf paths π of T_1, after restriction by π, the functions computed by all level-1 $\vee$-gates of the circuit depend on only k-variables. Therefore this also holds for their negations and hence, after restriction by π, each level-2 $\vee$-gate of the circuit is a k-DNF formula. Therefore we can apply Lemma 4.1 with a reduced number of variables (though the same k) to the set of level-2 gates after this restriction to derive a new restriction decision tree T_2^π. We say that a leaf of T_i (or a restriction defined by that leaf) to be *good* iff all level-i $\vee$-gates reduce to k-juntas after the restriction. The tree T_2 will be derived by taking all good π and appending the tree T_2^π at the leaf of T_1 reached by π. We repeat this again for T_2 and the level-2 $\vee$-gates and for each of the remaining levels up until we have built tree T_{d-1}. Since there is only one $\vee$-gate at level d, it suffices to apply Theorem 3.11 with $m = 2$ and $C = 2$ and append the complete decision tree on the variables of the s-junta (note here that this is a decision tree and not a *restriction* decision tree) computing the one formula associated with each good leaf of T_{d-1} to obtain a decision tree T_d. Clearly, at each good leaf of T_d the decision tree exactly computes the value of the $\vee$-gate at level d. The probability that a branch in T_d is in error is at most the sum of the probabilities of error associated with each of the levels.

It remains to compute the values of the parameters for the construction. Let the height of T_i be $n - n_i$. Then $n_0 = n$ and $n_i = n_{i-1} \cdot S^{-2/k}$ for $i \leq d-1$. Then since $4cd \leq c^{1/5}(\log_2 n)^{1/5} = k$, $n_{d-1} = n \cdot S^{-2(d-1)/k} \geq n^{1/2}$. Since $m = C = 2$ and n_{d-1} replaces n in the application of Theorem 3.11 for level d we have $s = 2^{k^4/2} k^{6k^3} 2^{k^3/2} < n^{c^{4/5}/\log_2^{1/5} n} < n_{d-1}/4$ and thus we can set $n_d > n_{d-1}/2 - s \geq n_{d-1}/4 > n \cdot S^{-2d/k}$. Because the n_i decrease rapidly, the probabilities that nodes are not good are bounded above by a quickly increasing geometric series whose largest term is that associated with the constructions of T_d. The failure probability in that case is: is at most $4k\exp(-2^{-2k-4}k^{-3}n_{d-1}/2)$ and thus the overall failure probability is at most $\exp(-n/S^{2d/k})$. The final bounds follow by observing that since $k = \log_2^{1/5} S$, $S^{1/k} = 2^{\log_2^{4/5} S}$.

The tree T_f is either T_d or T_d with its leaf values complemented, depending on the output gate of the circuit. To build the tree T'_f we simply take the tree T_f and append a complete decision tree on all leaves of T_f that are not good. The constructivity of T_f and T'_f follows from the constructivity given by Lemma 4.1 and the fact that we have sufficient leeway in the error probability to absorb the $n^{O(k)}$ term in other exponents. □