Non-Uniform ACC Circuit Lower Bounds

Ryan Williams*
IBM Almaden Research Center
San Jose, CA
ryanwill@us.ibm.com

***Abstract*—The class ACC consists of circuit families with constant depth over unbounded fan-in AND, OR, NOT, and MOD$_m$ gates, where $m > 1$ is an arbitrary constant. We prove:**

- **$\mathsf{NTIME}[2^n]$ does not have non-uniform ACC circuits of polynomial size. The size lower bound can be strengthened to quasi-polynomials and other less natural functions.**
- **E^{NP}, the class of languages recognized in $2^{O(n)}$ time with an NP oracle, doesn't have non-uniform ACC circuits of $2^{n^{o(1)}}$ size. The lower bound gives a size-depth tradeoff: for every d, m there is a $\delta > 0$ such that E^{NP} doesn't have depth-d ACC circuits of size 2^{n^δ} with MOD$_m$ gates.**

Previously, it was not known whether $\mathsf{EXP}^{\mathsf{NP}}$ had depth-3 polynomial size circuits made out of only MOD$_6$ gates. The high-level strategy is to design faster algorithms for the circuit satisfiability problem over ACC circuits, then prove that such algorithms can be applied to obtain the above lower bounds.

I. Introduction

Non-uniform computation allows the sizes of programs to grow with the sizes of inputs. A non-uniform computation can be naturally represented as an infinite family of *Boolean circuits*, one for each possible input length. A longstanding aim of complexity theory is to understand the power of non-uniform computation in relation to the usual uniform models which have fixed-size programs. One complication is that non-uniform computations can recognize *arbitrary* (even undecidable) languages by having a large enough circuit for each input length. Finding uniform computations that cannot be simulated by small non-uniform circuit families is an extremely difficult venture that is related to other major problems. For instance, $\mathsf{P} \neq \mathsf{NP}$ follows if one could provide an NP problem that cannot be solved by any circuit family where the size of the nth circuit is at most polynomial in n. Non-uniform lower bounds establish impossibility results for computation in the physical world: it could be that $\mathsf{P} \neq \mathsf{NP}$, yet NP problems can still be efficiently solved using "bloated" programs with sufficiently many lines of code for large inputs. Non-uniform circuit size lower bounds for NP could rule this out. (However, it is currently possible that every problem in NP has circuits of size $6n$.)

In the early 1980's, researchers began to carefully study the power of non-uniform *low depth* circuits. Intuitively, such circuits correspond to extremely fast parallel computations. The initial hope was that if some functions in NP were proved to require large, restricted circuit families, then by gradually lifting the restrictions over time, superpolynomial size unrestricted lower bounds for NP could be attained, proving $\mathsf{P} \neq \mathsf{NP}$. Furst, Saxe, and Sipser [FSS81] and independently Ajtai [Ajt83] showed that functions such as the parity of n bits cannot be computed by polynomial size AC^0 circuits, i.e., polynomial size circuit families of constant depth over the usual basis of AND, OR, and NOT gates, where each AND and OR may have arbitrarily many inputs. Yao [Yao85] improved the lower bounds to exponential size, and Håstad [Hås86] proved essentially optimal AC^0 lower bounds for parity. Around the same time, Razborov [Raz85] proved superpolynomial lower bounds for solving clique with *monotone* circuits (i.e., general circuits without NOT gates), and the bound was improved to exponential size by Alon and Boppana [AB87]. However, it was later shown [Raz89] that the monotone techniques probably would not extend to general circuits.

Encouraged by the progress on AC^0, attention turned to lower bounds for what seemed to be minor generalizations. The most natural generalization was to grant AC^0 the parity function for free. Razborov [Raz87] proved an exponential lower bound for computing the majority of n bits with constant-depth circuits made up of AND, OR, NOT, and MOD$_2$ gates. (A MOD$_m$ gate outputs 1 iff m divides the sum of its inputs.) Then Smolensky [Smo87] proved exponential lower bounds for computing MOD$_q$ with constant-depth circuits made up of AND, OR, NOT, and MOD$_p$ gates, for distinct primes p and q. Barrington [Bar89] suggested the next step would be to prove lower bounds for the class ACC, which consists of constant-depth circuit families over the basis AND, OR, NOT, and MOD$_m$ for arbitrary constant $m > 1$.[1] It is here that progress on strong lower bounds began to falter (although there has been progress on further restricted cases, cf. the Preliminaries). Although it was conjectured that the majority of n bits cannot have polynomial ACC circuits, strong ACC lower bounds remained elusive.

After failing to prove superpolynomial lower bounds for some time, the primary questions were weakened. Rather

* Supported by the Josef Raviv Memorial Fellowship.

[1]The class is also called ACC^0 in the literature. However, as ACC^i is hardly studied at all, for any $i > 0$, at the present time it makes sense to drop the superscript.

than trying to find simple functions that cannot be computed with weak circuits, perhaps we could rule out weak circuits for complicated functions. Could one prove that nondeterministic *exponential* time (NEXP) doesn't have polynomial size circuits? A series of papers [BFNW93], [KvM99], [IKW02] showed that even this sort of lower bound would imply derandomization results: in the case of NEXP lower bounds, it would imply that Merlin-Arthur games can be non-trivially simulated with nondeterministic algorithms. This indicated that proving circuit lower bounds for NEXP would already require significantly new ideas.

In this paper, we address two frontier questions concerning non-uniform circuit complexity:

1) *Does nondeterministic $2^{O(n)}$ time have non-uniform polynomial size* ACC *circuits?*
(Is $\mathsf{NTIME}[2^{O(n)}]$ in non-uniform ACC?)
2) *Does exponential time with an* NP *oracle have non-uniform polynomial size circuits?*
(Is $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}/\text{poly}$?)

Over the years, these questions have turned into notorious open problems, because it seems so obvious that the answers should be no. It was open if $\mathsf{EXP}^{\mathsf{NP}}$ could be recognized with depth-3 polynomial size circuits made out of only MOD_6 gates.[2] We make headway on these frontiers, giving a strong *no* answer to the first question.

Theorem 1.1: $\mathsf{NTIME}[2^n]$ does not have non-uniform ACC circuits of polynomial size.

Stronger size lower bounds hold (e.g. quasi-polynomial size); cf. the full version of the paper. For $\mathsf{EXP}^{\mathsf{NP}}$, we can prove exponential lower bounds.

Theorem 1.2 (Exponential Size-Depth Tradeoff): For every d and m, there is a $\delta > 0$ and a language in E^{NP} that fails to have non-uniform ACC circuits of depth d and size 2^{n^δ} with MOD_m gates.

Recall that the smallest complexity class for which we know exponential-size (unrestricted) circuit lower bounds is $\mathsf{EXP}^{\mathsf{NP}^{\mathsf{NP}}}$, the third level of the exponential hierarchy [MVW99].

Extending the approach of this paper to settle the second frontier question may be difficult, but this prospect does not look as implausible as it did before. If polynomial unrestricted circuits could be simulated by subexponential ACC circuits, or if one could improve just a little on the running time of algorithms for the circuit satisfiability problem, the second question would be settled.

A. An Overview of the Proofs

Let us sketch how these new lower bounds are proved, giving a roadmap for the rest of the paper. In recent work [Wil10], the author suggested a research program for proving non-uniform circuit lower bounds for NEXP. It was shown that for many circuit classes $\mathcal{C}$, sufficiently faster satisfiability algorithms for $\mathcal{C}$-circuits would entail new non-uniform lower bounds for $\mathcal{C}$-circuits. The objective of this paper is to carry out the proposed research program in the case of ACC circuits.

The proof of the lower bound for E^{NP} (Theorem 1.2) is a combination of complexity-theoretic ideas (time hierarchies, compression by circuits, the local checkability of computation) and algorithmic ideas (fast matrix multiplication, dynamic programming, table lookup).

1. First, we show that satisfiability algorithms for subexponential size n-input ACC circuits with running time $O(2^n/n^k)$ imply exponential size ACC lower bounds for E^{NP} (Theorem 3.2), where k is sufficiently large. (The model of computation for the satisfiability algorithm is flexible; we may assume the multitape Turing machine or a random access machine. See the Preliminaries.) This step considerably strengthens earlier work, which could only show that an $o(2^{n/3})$ time algorithm for ACC circuit satisfiability implies lower bounds [Wil10]. The idea is to prove that, if there is a faster algorithm for ACC Circuit SAT, and there are subexponential ($2^{n^{o(1)}}$) size ACC circuits for E^{NP}, then every $L \in \mathsf{NTIME}[2^n]$ can be accepted by a nondeterministic algorithm in $O(2^n n^{10}/n^k)$ time. (Here, 10 is a substitute for a small universal constant.) When $k > 10$ this contradicts the nondeterministic time hierarchy theorem [SFM78], [Zak83], so one of the assumptions must be false.

Two known facts are applied in the proof. First, there is a polynomial-time reduction from any $L \in \mathsf{NTIME}[2^n]$ to the NEXP-complete problem SUCCINCT 3SAT such that every instance x of length n (for sufficiently large n) is reduced to a (unrestricted, not ACC) circuit C_x of size $O(n^5)$ with at most $n+5\log n$ inputs (Fact 3.1). That is, the string obtained by evaluating C_x on its $O(2^n n^5)$ possible assignments (in lex order) encodes a 3CNF formula F_{C_x} that is satisfiable iff $x \in L$. Informally, this says that $L \in \mathsf{NTIME}[2^n]$ have "succinct" reductions to exponentially long 3SAT instances.

Second, if E^{NP} is in subexponential-size ACC, then (given an x) there is some satisfying assignment to the formula encoded by C_x that can be described by an ACC circuit W of subexponential size (Fact 3.2). That is, the string obtained by evaluating W on all possible assignments encodes a satisfying assignment to the exponentially long F_{C_x}. Informally, this means that, if E^{NP} has subexponential ACC circuits, then every "succinct" satisfiable 3SAT instance has at least one "succinct" satisfying assignment: compressible satisfiable formulas have compressible satisfying assignments.

We can try to combine these two "succinct" facts, as follows. If C_x were an ACC circuit, then any $L \in \mathsf{NTIME}[2^n]$ could be accepted in $O(2^n n^5/n^k)$ nondeterministic time, by guessing a subexponential ACC circuit W and constructing an ACC circuit satisfiability instance D built of C_x and W, where D is satisfiable if and only if W does not encode

[2]Note that slightly larger classes such as MAEXP and $\mathsf{NEXP}^{\mathsf{NP}}$ are known to not have polynomial size circuits; see the Preliminaries.

a satisfying assignment to F_{C_x} (as shown in the author's prior paper). The circuit D has at most $n + 5\log n$ inputs and $2^{n^{o(1)}}$ size, so the assumed ACC satisfiability algorithm can handle D in $O(2^n n^5/n^k)$ time.

The above argument doesn't quite work, because we do not know how to produce a C_x that is an ACC circuit (indeed, it may not be possible). An ACC SAT algorithm will not work on D, because D contains a copy of an unrestricted C_x. However, assuming P has subexponential ACC circuits, we show how to *guess and verify* an equivalent ACC circuit C'_x in nondeterministic $O(2^n n^{10}/n^k)$ time using a slightly faster ACC SAT algorithm (Lemma 3.1). This makes it possible to prove ACC lower bounds even with weak ACC satisfiability algorithms. Furthermore, this part of the proof does not use any specific properties of ACC, so it may be useful for proving stronger lower bounds in the future.

2. Next, we show how satisfiability of subexponential ACC circuits of depth d and n inputs can be determined in $2^{n-\Omega(n^\delta)}$ time, for a $\delta > 0$ that depends on d (Theorem 4.1). Given any such circuit C, replace it with C' which is an OR of 2^{n^δ} copies of C, where the first n^δ inputs of each copy are substituted with a variable assignment. This ACC circuit C' has $n - n^\delta$ inputs, $2^{O(n^\delta)}$ size, and C is satisfiable if and only if C' is. Applying a powerful result of Yao, Beigel-Tarui, and Allender-Gore (Lemma 4.1), C' can be replaced by an equivalent depth-2 circuit C'' of $2^{n^{\delta 2^{O(d)}}}$ size, which consists of an efficiently computable symmetric function at the output gate and AND gates below it. Setting $\delta \ll 1/2^{O(d)}$, and exploiting the structure of the depth-2 circuit, C'' can be evaluated on all of its possible assignments in $2^{n-n^\delta}\text{poly}(n)$ time (Lemma 4.2). This concludes the sketch of the E^{NP} lower bound.

The only use of the full assumption "E^{NP} has ACC circuits" is in Fact 3.2. The lower bound for NEXP (Theorem 1.1) applies the result (which follows from Impagliazzo, Kabanets, and Wigderson [IKW02]) that if NEXP has polynomial size (unrestricted) circuits then satisfiable instances of SUCCINCT 3SAT already have polynomial size (unrestricted) circuits W encoding satisfying assignments to these instances (Theorem 5.1). But if P has ACC circuits, then these unrestricted circuits must have equivalent ACC circuits as well (Lemma 5.1). This lets us extend the E^{NP} lower bound to NEXP. However, the resulting size lower bound is not exponential: from $S(n)$-size circuits for NEXP one only obtains $S(S(S(n)^c)^c)^c$-size ACC circuits encoding satisfying assignments. This allows for "half-exponential" type improvements in the size lower bounds against NEXP.

Perhaps the most interesting aspect of the proofs is that only the satisfiability algorithm for ACC circuits relies on specific properties of ACC. *Improved exponential algorithms for satisfiability are the only barrier to further progress on circuit lower bounds for* NEXP. In general, this work weakens the algorithmic assumptions necessary to prove lower bounds, and strengthens the lower bounds obtained. Let $\mathcal{C}$ be a class of circuit families that is closed under composition (the composition of two circuit families from $\mathcal{C}$ is also a family in $\mathcal{C}$) and contains AC^0. Possible $\mathcal{C}$ include constant-depth threshold circuits, Boolean formulas, and unrestricted Boolean circuits. The arguments of Section III and Section V imply the following metatheorem.

Theorem 1.3: There is a $k > 0$ such that, if satisfiability of $\mathcal{C}$-circuits with n variables and n^c size can be solved in $O(2^n/n^k)$ time for every c, then $\mathsf{NTIME}[2^n]$ doesn't have non-uniform polysize $\mathcal{C}$-circuits.

II. Preliminaries

We presume the reader has background in circuit complexity and complexity theory in general. The textbook of Arora and Barak [AB09] covers all the necessary material; in particular, Chapter 14 gives an excellent summary of ACC and the frontiers in circuit complexity.

Machine model: An important point about this work is that the choice of uniform machine model is not crucial to the arguments. We show that if large classes have small non-uniform ACC circuits, then $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[o(2^n)]$ (in fact, $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[o(2^n/n^k)]$ for sufficiently large k), which is a contradiction in all computational models we are aware of. Moreover, Gurevich and Shelah proved that the nondeterministic machine models are tightly related in their time complexities. For example, let $\mathsf{NTIME}_{RTM}[t(n)]$ be the languages recognized by nondeterministic $t(n)$ time random-access Turing machines, and let $\mathsf{NTIME}_{TM}[t(n)]$ be the class for multitape TMs.

Theorem 2.1 (Gurevich and Shelah [GS89]): $\bigcup_{c>0} \mathsf{NTIME}_{RTM}[n \log^c n] = \bigcup_{c>0} \mathsf{NTIME}_{TM}[n \log^c n]$.

As a consequence, even if we showed $\mathsf{NTIME}_{TM}[2^n] \subseteq \mathsf{NTIME}_{RTM}[2^n/n^k]$ for sufficiently large k, we would still obtain the desired contradiction. (Note that such a result is *not* known for the deterministic setting.) A random access Turing machine can also simulate a log-cost random access machine with only constant factor overhead [PR81]. Hence in our proof by contradiction, we may assume that the source algorithm we're simulating is only a multitape TM, while the target algorithm has all the power we need to perform typical computations from the literature.

Notation: Inside of an algorithm description, the integer n refers to the length of the input to the algorithm. For a function $f : \mathbb{N} \to \mathbb{N}$, we use $\text{poly}(f(n))$ to denote a growth rate of the form $cf(n)^c$ for a constant c.

The size of a circuit refers to the number of wires in it. However, since attention shall be restricted to circuits with at least polynomially many gates, the distinction between the number of wires and gates does not matter. An *unrestricted* circuit has gate types AND/OR/NOT, and each gate has fan-in two. (That is, an unrestricted circuit is the generic variety used in the definition of P/poly.) All circuit size functions S considered in this work are assumed to be monotone nondecreasing, i.e., $S(n+1) \geq S(n)$ for all n.

We say that a *circuit class* $\mathcal{C}$ is a collection of circuit families that (a) *contains* AC^0 (for every circuit family in AC^0, there is an equivalent circuit family in $\mathcal{C}$) and (b) *is closed under composition*: if $\{C_n\}$ and $\{D_n\}$ are families in $\mathcal{C}$, then for every c, the circuit family consisting of circuits which take n bits of input, feed them to $n^c + c$ copies of circuits from C_n, and feed the outputs to the inputs of D_{n^c+c}, is also a circuit family in $\mathcal{C}$. Essentially all classes studied extensively in the literature (AC^0, ACC, TC^0, NC^1, NC^2, P/poly, etc.) are circuit classes in this sense. For classes allowing for superpolynomial size circuits, "$n^c + c$" in the above may be relaxed appropriately.

For a complexity class $\mathcal{C}$, the class i.o.-$\mathcal{C}$ consists of languages $L \subseteq \Sigma^\star$ such that there is a language $L' \in \mathcal{C}$ where $L \cap \Sigma^n = L' \cap \Sigma^n$ holds for infinitely many n.

When the expression "$O(1)$" appears inside of the time bound for a complexity class, this is shorthand for the union of all classes where the $O(1)$ is substituted by a constant. For example, the class $\mathsf{TIME}[2^{n^{O(1)}}]$ is shorthand for $\bigcup_{c \geq 0} \mathsf{TIME}[2^{n^c}]$.

Other prior work: Kannan [Kan82] showed in 1982 that for any superpolynomial constructible function $S : \mathbb{N} \to \mathbb{N}$, the class $\mathsf{NTIME}[S(n)]^{\mathsf{NP}}$ does not have polynomial size circuits. Another class known to not have unrestricted polynomial size circuits is MAEXP [BFT98]. Later it was shown that the MAEXP lower bound can be improved to *half-exponential* size functions f which satisfy $f(f(n)) \geq 2^n$ [MVW99]. Kabanets and Impagliazzo [KI04] proved that $\mathsf{NEXP}^{\mathsf{RP}}$ *either* doesn't have polynomial size Boolean circuits (over AND, OR, NOT), *or* it doesn't have polynomial size arithmetic circuits (over the integers, with addition and multiplication gates). Note that $\mathsf{NEXP}^{\mathsf{RP}} \subseteq \mathsf{MAEXP}$.

A line of work has studied ways of representing ACC circuits by certain depth-two circuits which will play a critical role in this work. Define a SYM^+ circuit to be a depth-two circuit which computes some symmetric function at the output gate, and computes ANDs of input variables on the second layer. Yao [Yao90] showed that every ACC circuit of s size can be represented by a probabilistic SYM^+ circuit of $s^{O(\log^c s)}$ size, where c depends on the depth, and the ANDs have $\text{poly}(\log s)$ fan-in. Beigel and Tarui [BT94] showed how to remove the probabilistic condition. Allender and Gore [AG94] showed that every subexponential *uniform* ACC circuit family can be simulated by subexponential *uniform* SYM^+ circuits; as an application, the Permanent does not have *uniform* ACC circuits of subexponential size. Later, Allender [All99] improved the Permanent lower bound to polynomial size uniform TC^0 circuits. However, these proofs require uniformity, and the difference between uniformity and non-uniformity may well be vast (e.g., it is clear that $\mathsf{P} \neq \mathsf{NEXP}$, but open whether $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$). Green *et al.* [GKRST95] showed that the symmetric function can be assumed to be the specific function which returns the *middle bit* of the sum of its inputs. This representation may also be used in the lower bounds of this work.

There has also been other substantial work on representing ACC [BT88], [AAD00], [Han06], [KH09] as well as many lower bounds in restricted cases [BST90], [Thé94], [YP94], [KP94], [BS95], [Cau96], [Gro98], [GT00], [CGPT06], [CW09]. Significant work has gone into the *constant degree hypothesis* [BST90] that a certain type of low-depth ACC circuit requires exponential size to compute AND. The hypothesis is still open. These prior works on non-uniform ACC lower bounds attack the problem in a "bottom-up" way. Lower bounds have been proved for highly restricted circuits and the restrictions have been very gradually relaxed over time. The strategy of this paper is "top-down", finding the smallest complexity classes for which it is still possible to prove strong ACC lower bounds. This is in line with the goal of eventually proving circuit lower bounds for NP.

As mentioned before, this paper builds on the author's prior work which showed that mildly faster SAT algorithms can sometimes imply lower bounds. Let us briefly review the prior state-of-the-art for Circuit SAT algorithms. It is known that CNF SAT can be solved in $2^{n-\Omega(n/\ln(m/n))}\text{poly}(m)$ time, where m is the number of clauses and n is the number of variables [Sch05], [CIP06], [DH08]. Calabro, Impagliazzo, and Paturi have recently shown that AC^0 SAT can be solved by a randomized algorithm in $2^{n-n^{1-o(1)}}$ time, on circuits with $n^{1+o(1)}$ gates [CIP09]. Recently, Santhanam [San10] applied ideas inspired by formula size lower bounds to show that for a fixed k, Boolean formula SAT can be determined in $O(2^{n-n/c^k})$ time on formulas of size cn. Unfortunately, these upper bounds are not strong enough to prove new circuit lower bounds.

How do we avoid the barriers?: There are several well-known formal barriers to proving lower bounds. Let us say a little about their relation to this work. Intuitively, we circumvent the natural proofs barrier [RR97] because of the use of diagonalization. More precisely, we rely heavily on strong completeness properties of a specific NEXP language, namely SUCCINCT 3SAT, to prove that it cannot have small ACC circuits. So it looks unlikely that one may extract any P-natural or NP-natural properties from the proof. (Furthermore, there is little evidence that ACC contains pseudorandom functions, so natural proofs may not be a barrier for ACC after all.) It is hard to formally rule out that a proof cannot possibly be made natural, without showing either an algorithmic lower bound (there is no efficient algorithm with certain properties) or a circuit upper bound (the circuit class under discussion has pseudorandom functions). Nevertheless, statements like "Satisfiability of $\mathcal{C}$ circuits is in $O(2^n/n^{10})$ time" do appear to be far weaker than statements like "There are no strong pseudorandom functions implementable with $\mathcal{C}$ circuits".

More conclusively, the approach of this work definitely avoids relativization [BGS75] and algebrization [AW09] because there are oracles A relative to which $\mathsf{NEXP}^{A} \subset \mathsf{ACC}^{A}$, and even $\mathsf{NEXP}^{\tilde{A}} \subset \mathsf{ACC}^{A}$ (Scott Aaronson, personal communication). The ACC SAT algorithm used in the lower bounds relies on non-relativizing properties of ACC circuits. In general, the approach of using slightly-faster SAT algorithms to prove lower bounds appears fruitful for circumventing oracle-based barriers, because all known improved SAT algorithms break down when oracles (or algebraic extensions thereof) are added to the instance. That is, improvements over exhaustive search necessarily exploit structure in instances that black-box methods cannot see.

III. A Strengthened Connection Between SAT Algorithms and Lower Bounds

In this section, we prove that if one can achieve a very minor improvement over exhaustive search in satisfying ACC circuits, then one can prove lower bounds for ACC. The required improvement is so minor that we are able to achieve it, in the sequel. However, all results in this section hold equally well for other circuit classes as well: we only require basic properties of ACC that practically all robust circuit classes satisfy.

Define the ACC Circuit Sat problem to be: *given an* ACC *circuit C, is there an assignment of its inputs that makes C evaluate to* 1*?* In recent prior work [Wil10], the author proved a relation between algorithms for ACC Circuit Sat and lower bounds for ACC circuits:[3]

Theorem 3.1 ([Wil10]): Let $s(n) = \omega(n^k)$ for every k. If ACC Circuit Sat instances with n variables and n^c size can be solved in $O(2^{n/3}/s(n))$ time for every c, then E^{NP} does not have non-uniform ACC circuits of polynomial size.

We shall sharpen this theorem considerably. Let $S : \mathbb{N} \to \mathbb{N}$ be a monotone nondecreasing function such that $S(n) \geq n$. Let $\mathcal{C}$ be a circuit class as defined in the Preliminaries. ($\mathcal{C}$ can be ACC, TC^0, NC^1, P/poly, etc.) Define the $\mathcal{C}$-Circuit Sat problem to be: *given a circuit C from class $\mathcal{C}$, is there an assignment of its inputs that makes C evaluate to* 1*?*

Theorem 3.2: Let $S(n) \leq 2^{n/4}$. There is a $c > 0$ such that, if $\mathcal{C}$-Circuit Sat instances with at most $n + c \log n$ variables, depth $2d + O(1)$, and $O(n\ S(2n) + S(3n))$ size can be solved in $O(2^n/n^c)$ time, then E^{NP} does not have non-uniform $\mathcal{C}$ circuits of depth d and $S(n)$ size.

The constant c depends on the model of computation, but for all typical models, c is not large (less than 10). For us, the important corollary is this: if ACC SAT has a slightly faster algorithm on circuits that are mildly larger than $S(n)$, then E^{NP} does not have ACC circuits of $S(n)$ size. In what follows, we prove Theorem 3.2 only for ACC circuits, but the proof also works for other circuit classes. (The reader can verify that the only two properties of ACC used are that the class contains AC^0, and the class is closed under composition of circuit families.)

[3]In fact a more general result for any circuit class was proved, which implies Theorem 3.1.

To understand the difficulty behind proving Theorem 3.2, let us recall the proof of Theorem 3.1 to see why it needed such a strong assumption. The generic proof idea is to derive a contradiction from assuming small circuits for E^{NP} and a faster algorithm for Circuit Sat. In particular, it is shown that under the two assumptions, every language $L \in \mathsf{NTIME}[2^n]$ can be recognized in $\mathsf{NTIME}[o(2^n)]$, which is false by the nondeterministic time hierarchy [SFM78], [Zak83]. The contradiction is derived from stitching together several facts about circuits and satisfiability.

Define Succinct 3SAT as the problem: *given a circuit C on n inputs, let F_C be the 2^n-bit instance of 3-SAT obtained by evaluating C on all of its possible inputs in lexicographical order. Is F_C satisfiable?*

That is, given a *compressed encoding* of a 3-CNF formula, the task is to determine if the underlying decompressed formula is satisfiable. Call F_C the *decompression* of C, and call C the *compression* of F_C. The Succinct 3SAT problem is a canonical NEXP-complete problem [PY86].

Fact 3.1: There is a constant $c > 0$ such that for every $L \in \mathsf{NTIME}[2^n]$, there is a reduction from L to Succinct 3SAT which on input x of length n runs in poly(n) time and produces a circuit C_x with at most $n + c \log n$ inputs and $O(n^c)$ size, such that $x \in L$ if and only if the decompressed formula F_{C_x} of $2^n \cdot \text{poly}(n)$ size is satisfiable.

Fact 3.1 follows from several prior works concerned with the complexity of the Cook-Levin theorem.

Theorem 3.3 ([Tou01], [FLvMV05]): There is a $c > 0$ such that for all $L \in \mathsf{NTIME}[n]$, L reduces to 3SAT in $O(n(\log n)^c)$ time. Moreover there is an algorithm (with random access to its input) that, given an instance of L with length n and an integer $i \in [dn(\log n)^c]$ in binary (for some d depending on L), outputs the ith clause of the resulting 3SAT formula in $O((\log n)^c)$ time.

The proofs in the above references build on even earlier work of Schnorr, Cook, Gurevich-Shelah, and Robson [Sch78], [Coo88], [GS89], [Rob91]. In a nutshell, all of these proofs exploit the *locality of computation*: every nondeterministic computation running in linear time can be represented with a nondeterministic circuit of size $O(n \cdot \text{poly}(\log n))$ which has a highly regular and efficiently computable structure. This circuit can be easily modeled as a 3-CNF formula using the Tseitin transformation that assigns a variable to each circuit wire and uses 3-CNF clauses to model the input-output relationships for each gate.

The value of c in Theorem 3.3 depends on the underlying computational model; typically one can take $c \leq 4$. A standard padding argument (substituting 2^n in place of n) yields Fact 3.1. In more detail, given $L \in \mathsf{NTIME}[2^n]$, we apply Theorem 3.3 to the language $L' = \{x01^{2^{|x|}} \mid x \in L\}$, which is in $\mathsf{NTIME}[n]$. On an input x, this generates an equivalent 3SAT instance of length $O(2^{|x|}|x|^c)$. As it is easy to simulate random accesses to an input of the form $x01^{2^{|x|}}$ with a uniform poly$(|x|)$ size circuit, one can simulate the $O((\log n)^c)$ time algorithm of Theorem 3.3 on L', with a uniform poly$(|x|^c)$ size circuit.

Using Fact 3.1, one can prove that every compressible satisfiable formula output by the SUCCINCT 3SAT reduction has a compressible *satisfying assignment*.

Fact 3.2: If E^{NP} has ACC circuits of size $S(n)$, then there is a constant c such that for every language $L \in \mathsf{NTIME}[2^n]$ and every $x \in L$ of length n, there is an ACC circuit W_x of size at most $S(3n)$ with $k \le n + c\log n$ inputs such that the variable assignment $z_i = W(i)$ for all $i = 1, \ldots, 2^k$ is a satisfying assignment for the formula F_{C_x}, where C_x is the circuit obtained by the reduction in Fact 3.1.

Proof of Fact 3.2. Consider the E^{NP} machine:

$N(x,i)$: *Compute the* SUCCINCT 3SAT *reduction from x to C_x in polynomial time. Decompress C_x, obtaining a formula F of $O(2^{|x|}|x|^c)$ size. Let k be the number of inputs to C_x. Binary search for the lexicographically smallest satisfying assignment A to F, by repeatedly querying:* given (F, A) where $|A| \le 2^k$, is there an assignment $A' \le A$ that satisfies F? *Then output the ith bit of A.*

Note the queries can be answered in NP, and N needs $O(2^k)$ queries to the oracle. By assumption, N has ACC circuits of size $S(n)$. It follows that for every $x \in L$ there is some satisfying assignment to F which is encoded by a circuit of size $S(|\langle x,i\rangle|) \le S(3|x|)$, where $\langle\cdot,\cdot\rangle$ is a polynomial-time computable pairing function. ■

With these two facts, we may try to recognize any $L \in \mathsf{NTIME}[2^n]$ with a $o(2^n)$ nondeterministic algorithm (a contradiction), as follows. Given a string x of length n, compute the SUCCINCT 3SAT circuit C_x in polynomial time and nondeterministically *guess* a $S(3n)$ size circuit W. We must verify that W succinctly encodes a satisfying assignment for the underlying formula F_{C_x}. To verify this, the algorithm constructs a CIRCUIT SAT instance D. The circuit D has $n + c\log n$ inputs fed to $O(n)$ copies of C_x, so that when i is input to D, the copies altogether print the ith clause of the 3CNF formula F_{C_x}. These copies output three variable indices of length at most $n + c\log n$, along with sign bits (whether or not the variables are negated in the clause). Then D feeds each index to a copy of W, which prints a bit. Finally D compares the sign bits with the three bits printed by the copies of W, and outputs 0 iff the assignment encoded by W satisfies the ith clause. Observe D has poly$(n) + O(S(3n))$ size. Running a fast enough CIRCUIT SAT algorithm lets us determine the satisfiability of D in $o(2^n)$ time. Finally, this algorithm for L accepts x iff D is unsatisfiable. To see that this algorithm is correct, observe there is a size-$S(3n)$ circuit W such that D is unsatisfiable, if and only if there is such a W encoding a satisfying assignment for F_{C_x}, if and only if $x \in L$.

The above argument cannot be carried out directly to prove ACC circuit lower bounds from ACC CIRCUIT SAT algorithms, because of Fact 3.1. Given an instance x of L, the circuit C_x produced in the reduction from L to SUCCINCT 3SAT can be constructed in polynomial time, however it looks hard (perhaps impossible) to implement C_x directly with ACC circuits. As C_x is a component of the circuit D, it follows that D itself would not be an ACC circuit, so an ACC CIRCUIT SAT algorithm would not seem to be useful for determining the satisfiability of D.

In the proof of Theorem 3.1 in the author's prior work [Wil10], this problem was fixed by settling for a weaker reduction from L to SUCCINCT SAT, which generates an AC^0 circuit C'_x with $3n + O(\log n)$ inputs rather than $n + O(\log n)$. Unfortunately this constant factor makes a huge difference: to determine satisfiability of the resulting circuit D' in $o(2^n)$ time, a $2^{n/3}/n^{\omega(1)}$ time algorithm for ACC CIRCUIT SAT is needed, instead of $2^n/n^{\omega(1)}$ time. Algorithms of the former type are not known even for 3SAT; algorithms of the latter type are much more plentiful.

While it is unlikely that these C_x circuits can be implemented in ACC, note we already *assume* that ACC is powerful: in a proof by contradiction, we assume many functions have small ACC circuits! Since the function computed by C_x is computable in P, then even if we assume P has ACC circuits, there still *exists* a circuit C'_x which is ACC and equivalent to C_x, but it is from a non-uniform family, and therefore may be difficult to construct. However, we can use nondeterminism in the algorithm recognizing L in $\mathsf{NTIME}[o(2^n)]$, so at the very least we can *guess* this elusive C'_x. We also have a good algorithm for ACC CIRCUIT SAT at our disposal. By guessing two more ACC circuits to help, it turns out that we can always generate a correct ACC circuit C'_x that is equivalent to C_x in $o(2^n)$ time.

Lemma 3.1: There is a fixed $d > 0$ with the following property. Assume P has ACC circuits of depth d' and size at most $S(n)$, and assume ACC CIRCUIT SAT on circuits with $n + c\log n$ inputs, depth $2d' + O(1)$, and at most $O(S(3n) + S(2n)n)$ size can be solved in $O(2^n/n^c)$ time, for large enough $c > 2d$. Then for every $L \in \mathsf{NTIME}[2^n]$, there is a nondeterministic algorithm $\mathcal{A}$ such that:

- $\mathcal{A}$ runs in $O(2^n/n^c + S(3n)\cdot \text{poly}(n))$ time,
- for every x of length n, $\mathcal{A}(x)$ either prints *reject* or it prints an ACC circuit C'_x with $n + d\log n$ inputs, depth d', and $S(n + d\log n)$ size, such that $x \in L$ if and only if C'_x is the compression of a satisfiable 3-CNF formula of $2^n \cdot \text{poly}(n)$ size, and
- there is always at least one computation path of $\mathcal{A}(x)$ that prints the circuit C'_x.

That is, given an instance x, the algorithm $\mathcal{A}$ nondeterministically generates an equivalent SUCCINCT 3SAT instance C'_x which is an ACC circuit.

Proof of Lemma 3.1. We describe $\mathcal{A}$ in detail. On input x of length n, $\mathcal{A}$ guesses an ACC circuit C'_x of size $S(n+d\log n)$, and constructs the SUCCINCT 3SAT circuit C_x with $n + d\log n$ inputs and $kn^d + k$ size (of Fact 3.1) in polynomial time, for some d independent of L. By Fact 3.1, $x \in L$ if and only if C_x is the compression of a satisfiable formula F_{C_x} of $O(2^n n^d)$ length. We must efficiently verify that C'_x and C_x compute the same function.

WLOG, the unrestricted circuit C_x has gate types AND, OR, NOT, and INPUT, where each AND and OR has fan-in two. By definition an INPUT gate has no inputs, and the output value of an INPUT is the appropriate input bit itself. The gates are indexed by the numbers $1, \ldots, kn^d + k$, where the first $n + d\log n$ indices correspond to the $n + d\log n$ INPUT gates, and the $(kn^d + k)$th gate is the output gate.

Since the map $x \mapsto C_x$ is polynomial time computable, the following function f is polynomial-time computable:

Given x, and a gate index $j = 1, \ldots, kn^d + k$, $f(x, j)$ outputs the gate type (AND, OR, NOT, INPUT) of the jth gate in the circuit C_x. Furthermore, if the gate type is NOT, then f outputs the gate index j_1 in C_x whose output is the input to j; if the gate type is an AND or OR, then f outputs the two gate indices j_1 and j_2 in C_x whose outputs are the two inputs of j.

Consider the decision problem D_f: *given x, j, and $i = 1, \ldots, 2d\log n + O(1)$, decide if the ith bit of $f(x, j)$ is* 1. D_f is solvable in P and hence has $O(S(n + d\log n + O(\log\log n)))$-size, d'-depth ACC circuits, by assumption.

Let $D(x, j)$ be an ACC circuit implementing the functionality of f. Note we may assume the size of D is $O(S(n+O(\log n))\log n)$, by simply taking $2d\log n + O(1)$ copies of the $S(n + O(\log n))$-size circuit solving the decision problem D_f. (By convention, let us assume that when D is printing the gate information for an INPUT gate, it prints all-zeroes strings in place of j_1 and j_2, and when D is printing the information for a NOT gate, it prints all-zeroes in place of j_2.)

The nondeterministic algorithm $\mathcal{A}$ guesses D, and verifies that D is correct on the given input x in time

$$\begin{aligned} &O(n^d S(n + O(\log n)) \cdot \text{poly}(\log S(n + O(\log n)))) \\ &\qquad \le n^d \cdot S(2n) \cdot \text{poly}(\log S(2n)) \le O(2^{2n/3}), \end{aligned}$$

by evaluating $D(x, \cdot)$ on all $j = 1, \ldots, kn^d + k$, checking that the outputs of D correspond to the gates of C_x. If D does not output all gates of C_x correctly, then $\mathcal{A}$ *rejects*.

Next, consider the problem:

Given x, an input i of $n + d\log n$ bits, and a gate index $j = 1, \ldots, kn^d + k$, output the bit value on the output wire of the jth gate when C_x is evaluated on i.

By assumption, this problem also has ACC circuits, since C_x can be constructed and evaluated on any input i in polynomial time. Let $E(x, i, j)$ be an ACC circuit of size $S(n + (n + d\log n) + d\log n + O(1)) \le S(3n)$ and depth d' with this functionality.

Now $\mathcal{A}$ guesses E and wishes to verify its correctness on x. To do this, $\mathcal{A}$ constructs a circuit VALUE(i, j) built out of D and E, where i has $n + d\log n$ bits and $j = 1, \ldots, kn^d + k$. Intuitively, VALUE(i, j) will output 0 if and only if E produces a sensible output for the jth gate of C_x evaluated on input i.

First, VALUE(i, j) feeds j to the circuit $D(x, \cdot)$, producing gate indices j_1, j_2, and a gate type g. VALUE then computes $v_1 = E(x, i, j_1)$, $v_2 = E(x, i, j_2)$ and $v = E(x, i, j)$. (Depending on g, these j_1 and j_2 may be all-zeroes, but this does not matter to us.)

If $g =$ INPUT, then VALUE outputs 0 if and only if $j \in \{1, \ldots, n + d\log n\}$ and the jth bit of i equals v. This behavior can be implemented with an AC^0 circuit of $O(n\log S(n))$ size: an AND over all $2n$ choices of a bit from input i along with a bit v, of ORs of fan-in $\log S(n) + O(1)$.

If $g =$ NOT, VALUE outputs 0 if and only if $v_1 = \neg v$.

If $g =$ AND, VALUE outputs 0 if and only if $v_1 \wedge v_2 = v$.

If $g =$ OR, VALUE outputs 0 if and only if $v_1 \vee v_2 = v$.

Each of the above three conditions can be implemented with $O(1)$ gates, given the values g, v_1, v_2, and v. It follows that VALUE can be implemented as an ACC circuit.

Since $\mathcal{A}$ has not rejected, D is correct, so we know that for all i, j, the gate types g and input connections j_1 and j_2 are correct. Therefore VALUE$(i, j) = 1$ if and only if E asserts that the output of gate j in $C_x(i)$ equals v, and E asserts the inputs to j have values v_1, v_2, but the gate type g dictates that the output of j should be $\neg v$. It follows that VALUE is an unsatisfiable circuit if and only if E prints correct values for all gates in $C_x(i)$, over all i.

Therefore, by calling ACC SAT on VALUE$(\cdot, \cdot)$, $\mathcal{A}$ determines whether E is correct. The algorithm $\mathcal{A}$ *rejects* if E is deemed incorrect. The circuit VALUE(i, j) has $n + 2d\log n + O(1)$ inputs, depth $2d' + O(1)$, and $O(S(3n) + S(2n)\log S(2n) + n\log S(n)) \le O(S(3n) + S(2n)n)$ size. By assumption, the assumed ACC SAT algorithm runs in $O(2^n/n^c)$ time for c chosen to be greater than $2d$.

After checking that E is a correct guess, the question of whether C'_x is equivalent to C_x can now be verified. (Alternatively, at this point we may simply print the circuit $E(\cdot, kn^d + k)$ as a valid circuit that is equivalent to $C_x(\cdot)$.) First, if E is correct, then for all i, $C_x(i) = E(x, i, kn^d + k)$. Therefore it suffices to set up an ACC circuit EQUIV(i) which outputs 1 if and only if $C'_x(i) \neq E(x, i, kn^d + k)$, and determine if EQUIV is satisfiable using the algorithm for ACC CIRCUIT SAT. Since EQUIV(i) has $n + d\log n$ inputs, depth $d' + O(1)$, and size $O(S(n + O(\log n)))$, the circuit SAT call runs in $O(2^n/n^c)$ time by assumption. If EQUIV is satisfiable, then $\mathcal{A}$ *rejects*.

Finally, $\mathcal{A}$ prints its guessed circuit C'_x if the algorithm did not reject on any of the above steps. ■

Remark 1: The proof of the lemma does not require specific properties of ACC. We only need that the underlying circuit class $\mathcal{C}$ contains AC^0 and is closed under composition of two circuit families. The same goes for the proof of Theorem 3.2 below.

With Lemma 3.1, the proof of Theorem 3.2 closely follows the author's prior work (Theorem 3.1), except the circuit C'_x is substituted in place of C_x. Let us give the details, using the specific example of ACC in place of a generic circuit class $\mathcal{C}$.

*Reminder of Theorem 3.2 Let $S(n) \leq 2^{n/4}$. There is a $c > 0$ such that, if $\mathcal{C}$-*CIRCUIT SAT *instances with at most $n + c \log n$ variables, depth $2d + O(1)$, and $O(n\ S(2n) + S(3n))$ size can be solved in $O(2^n/n^c)$ time, then E^{NP} does not have non-uniform $\mathcal{C}$ circuits of depth d and $S(n)$ size.*

Proof of Theorem 3.2. Suppose ACC CIRCUIT SAT instances with $n + c \log n$ variables, depth $2d + O(1)$, and $O(n\ S(2n) + S(3n))$ size can be solved in $O(2^n/n^c)$ time for a sufficiently large c. Further suppose that E^{NP} has non-uniform ACC circuits of depth d and $S(n)$ size. The goal is to show that $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[o(2^n)]$, contradicting the nondeterministic time hierarchy [SFM78], [Zak83].

Let $L \in \mathsf{NTIME}[2^n]$. We describe a fast nondeterministic algorithm $\mathcal{B}$ deciding L. As discussed earlier (Lemma 2.1), we may assume L is accepted by a nondeterministic multitape TM in $O(2^n)$ time, and we only need to simulate L on a nondeterministic RAM in $O(2^n/n^c)$ time for large enough c to obtain the contradiction.

On input x of length n, $\mathcal{B}$ first runs the nondeterministic algorithm $\mathcal{A}$ of Lemma 3.1. Using the ACC CIRCUIT SAT algorithm and the fact that P has ACC circuits, $\mathcal{A}$ runs in $O(2^n/n^c + S(3n) \cdot \mathrm{poly}(n)) \leq O(2^n/n^c)$ time, and for some computation path, $\mathcal{A}$ produces an ACC circuit C'_x of $S(n + c \log n)$ size and $n + c \log n$ inputs such that $x \in L$ if and only if C'_x is the compression of a satisfiable formula $F_{C'_x}$.

Then $\mathcal{B}$ nondeterministically guesses a $S(3n)$-size circuit W. By Fact 3.2, there exists such a W that encodes a satisfying assignment for $F_{C'_x}$ if and only if $x \in L$.

Next, $\mathcal{B}$ constructs an ACC CIRCUIT SAT instance D to verify that W is correct (just as in the proof of Theorem 3.1). The circuit D has $n + c \log n$ inputs fed to $O(n)$ copies of C'_x, so that when i is input to D, the ith clause of the 3CNF formula $F_{C'_x}$ is printed on $O(n)$ bits of output. The $O(n)$ bits encode three variable indices along with sign bits for each variable. For the three variables, an assignment is computed for them by evaluating the indices on three copies of W. Finally, D compares the sign bits with the bits output by the copies of W, and outputs 0 iff the variable assignment encoded by W satisfies the ith clause.

Observe that D has $O(n\ S(2n) + S(3n))$ size, depth $2d + O(1)$, and $n + c \log n$ inputs. By assumption, the satisfiability of D can be determined in $O(2^n/n^c)$ time, hence $\mathcal{B}$ decides if $x \in L$ in $O(2^n/n^c)$ time. ■

IV. A SATISFIABILITY ALGORITHM FOR ACC CIRCUITS

Now we present an algorithm for ACC SAT that runs slightly faster than the 2^n runtime of exhaustive search. There are two components in the algorithm: a nice representation of ACC circuits, and a method for evaluating this representation quickly on all of its inputs.

It follows from the work of Yao [Yao90], Beigel and Tarui [BT94], and Allender and Gore [AG94] that, given any ACC circuit of size s, one can produce a $s^{O(\log^c s)}$ size SYM^+ circuit in $\mathrm{poly}(s^{O(\log^c s)})$ time that has equivalent functionality, and very special properties. (For more background, see the Preliminaries.)

Lemma 4.1: There is an algorithm and function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that given an ACC circuit with MOD_m gates of depth d and size s, the algorithm outputs an equivalent SYM^+ circuit of $s^{O(\log^{f(d,m)} s)}$ size. The algorithm takes at most $s^{O(\log^{f(d,m)} s)}$ time. Furthermore, given the number of ANDs in the circuit that evaluate to 1, the symmetric function itself can be evaluated in $s^{O(\log^{f(d,m)} s)}$ time.

The function $f(d, m)$ is estimated to be no more than $m^{O(d)}$. Allender and Gore show that given a *uniform* ACC circuit (with an efficiently computable connection language), there is a similarly *uniform* SYM^+ circuit of the appropriate size. Their proof gives an efficient, deterministic algorithm computing the transformation, and it works equally well if it is simply given any ACC circuit as input (not necessarily uniform). A proof of the lemma is in the full version.

A. Rapid evaluation of an ACC circuit on all of its inputs

The other component of the ACC SAT algorithm is a method for rapidly evaluating a given SYM^+ circuit on all of its possible satisfying assignments:

Lemma 4.2 (Evaluation Lemma): There is an algorithm that, given a SYM^+ circuit of size $s \leq 2^{0.1n}$ and n inputs with a symmetric function that can be evaluated in $\mathrm{poly}(s)$ time, runs in $(2^n + \mathrm{poly}(s)) \cdot \mathrm{poly}(n)$ time and prints a 2^n-bit vector V which is the truth table of the function represented by the given circuit. That is, $V[i] = 1$ iff the SYM^+ circuit outputs 1 on the ith variable assignment.

That is, any SYM^+ circuit can be evaluated on all 2^n assignments in *polynomial amortized time per assignment*. Brute force search would take $2^n \cdot \mathrm{poly}(s)$ time, but the algorithm manages to use roughly $2^n + \mathrm{poly}(s)$ time instead.

Unfortunately we do not have space to include the proof of Lemma 4.2 here. However we can give some intuition. Representing a function f with a size-s SYM^+ circuit amounts to representing f as a composition of two functions $f = g \circ h$, where $g : \{0, \ldots, s\} \to \{0, 1\}$ is arbitrary, and $h : \{0, 1\}^n \to \{0, \ldots, s\}$ is a multilinear polynomial with at most s monomials. The h is provided in its coefficient representation: all coefficients for the monomials are specified. All we need is to efficiently convert this representation into the *point representation*, giving the value of h on all points in $\{0,1\}^n$. With the point representation of h, it is easy to evaluate $g \circ h$ on all 2^n points in $O(2^n \log s)$ time, by storing g as a lookup table.

There are multiple ways to efficiently perform the conversion; here is a simple one. We may write h as $h = x_1 \cdot h_1(x_2, \ldots, x_n) + h_2(x_2, \ldots, x_n)$ for multilinear h_1 and h_2. This suggests the strategy: recursively obtain 2^{n-1}-vectors R_1 and R_2 which point-represent h_1 and h_2, then output the 2^n point representation of h as the concatenation of R_2 and $R_1 + R_2$. (If $n = 2$, then compute the representation directly.) This recursive strategy can be implemented in $2^n \cdot \text{poly}(n) + \text{poly}(s)$ time, even on multitape TMs.

B. The final algorithm

Theorem 4.1: For $d > 1$ and $m > 1$ there is an $\varepsilon \in (0,1)$ such that satisfiability of depth-d ACC circuits with MOD_m gates, n inputs, and 2^{n^ε} size can be determined in $2^{n-\Omega(n^\delta)}$ time for some $\delta > \varepsilon$ that depends only on d and m.

Proof: Let ℓ, ε be parameters to set later. Suppose we are given a depth-d ACC circuit C of $s = 2^{n^\varepsilon}$ size and n inputs. Make a circuit C' with $s \cdot 2^\ell$ size and $n - \ell$ inputs which is obtained by producing 2^ℓ copies of C, plugging in a different possible assignment to the first ℓ inputs of C in each copy, and taking the OR of these copies. Observe C' is a depth-$(d+1)$ ACC circuit, and C is satisfiable if and only if C' is satisfiable.

Applying Lemma 4.1, a circuit C'' equivalent to C' can be produced, where C'' consists of a symmetric gate connected to $s'' \leq s^{e(\ell^e \log^e s)}$ ANDs of variables, for some constant e that depends on the depth d and modulus m. Producing C'' from C' takes only $s^{O(\ell^e \log^e s)}$ steps. When $s = 2^{n^\varepsilon}$, $s'' \leq 2^{en^\varepsilon(\ell^e n^{\varepsilon e})}$. Set $\ell = n^{1/(2e)}$, and observe $s'' \leq 2^{n^{2/3}}$ for all sufficiently large n and sufficiently small ε.

By the evaluation lemma (Lemma 4.2) and the fact that the symmetric function of C'' can be evaluated in $\text{poly}(s'')$ time, C'' can be evaluated on all of its possible assignments in $O(2^{n-\ell} \cdot \text{poly}(n)) \leq 2^{n-\Omega(n^{1/(2e)})}$ time, hence the satisfiability of C can be determined within this time. ■

V. ACC Lower Bounds

Combining the results of the previous two sections, non-uniform lower bounds for ACC can be proved.

Reminder of Theorem 1.2 For every d and m, there is a $\delta > 0$ and a language in E^{NP} that fails to have non-uniform ACC *circuits of depth d and size 2^{n^δ} with MOD_m gates.*

Proof: Theorem 4.1 states that for every d and m there is an $\varepsilon > 0$ so that satisfiability of depth-d ACC circuits with MOD_m gates, n inputs, and $2^{O(n^\varepsilon)}$ size can be solved in $2^{n-\Omega(n^\delta)}$ time, for some $\delta > \varepsilon$. Theorem 3.2 says there is a $c > 0$ such that, if ACC Sat instances with $n + c\log n$ variables, depth $2d + O(1)$, and $s = n\ 2^{O(n^\varepsilon)}$ size can be solved in $O(2^n/n^c)$ time, then E^{NP} does not have ACC circuits of depth d and 2^{n^ε} size. The lower bound follows, as $2^{(n+c\log n)-\Omega((n+c\log n)^\delta)} \ll O(2^n/n^c)$ for every c. ■

Note this lower bound can be "padded down":

Corollary 5.1: For every constant $d \geq 2$, the class $\mathsf{QuasiP}^{\mathsf{NP}} = \mathsf{TIME}[n^{\log^{O(1)} n}]^{\mathsf{NP}}$ does not have non-uniform ACC circuits of depth d and polynomial size.

It is known that $\mathsf{NTIME}[n^{\log^{O(1)} n}]^{\mathsf{NP}}$ does not have polynomial size (unrestricted) circuits [Kan82]. Superpolynomial ACC lower bounds for NEXP are also provable. First we need a theorem established in prior work: if NEXP has (unrestricted) polynomial size circuits, then every satisfiable formula output by the Succinct 3SAT reduction in Fact 3.1 has some *satisfying assignment* that can be represented with a polynomial size unrestricted circuit.

More precisely, say that Succinct 3SAT *has succinct satisfying assignments* if there is a constant c such that for every language $L \in \mathsf{NTIME}[2^n]$ and every $x \in L$ of length n, there is a circuit W_x of $\text{poly}(n)$ size with $k \leq n + c\log n$ inputs such that the variable assignment $z_i = W(i)$ for all $i = 1, \ldots, 2^k$ is a satisfying assignment for the formula F_{C_x}, where C_x is the circuit obtained by the Succinct 3SAT reduction in Fact 3.1. Say that W_x *is a succinct satisfying assignment for C_x.*

Theorem 5.1 ([Wil10]): Suppose NEXP has polynomial size circuits. Then Succinct 3SAT has succinct satisfying assignments.

Theorem 5.1 is not explicitly proved in the paper, however it follows immediately from another theorem. Say that NEXP has *universal witness circuits of polynomial size* if for every $L \in \mathsf{NEXP}$ and every correct exponential time verifier for L, there is a $c > 0$ such that for every $x \in L$, there is a circuit of size at most $|x|^c + c$ which encodes a witness for x that is accepted by the verifier. (For more formal definitions, see [Wil10].) The following directly implies Theorem 5.1:

Theorem 5.2 ([IKW02], [Wil10]): If $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$ then every language in NEXP has universal witness circuits of polynomial size.

The proof of Theorem 5.2 follows an argument by Impagliazzo, Kabanets, and Wigderson [IKW02]. The second ingredient is a simple folklore lemma.

Lemma 5.1 (Folklore): Let $\mathcal{C}$ be any circuit class. If P has non-uniform $\mathcal{C}$ circuits of $S(n)^{O(1)}$ size, then there is a $c > 0$ such that every $T(n)$-size circuit family (uniform or not) has an equivalent $S(n+O(T(n)\log T(n)))^c$-size circuit family in $\mathcal{C}$.

Proof: If P has non-uniform $S(n)^{O(1)}$-size $\mathcal{C}$ circuits, then for some $c > 0$, the Circuit Eval problem has $S(n)^c$-size circuits. (Recall the Circuit Eval problem is: *given an arbitrary Boolean circuit C and input x, evaluate C on x and output the answer.*) Let $\{D_n(\cdot,\cdot)\}$ be a $S(n)^c$-size circuit family for this problem. Now let $\{C_n\}$ be an arbitrary $T(n)$-size circuit family. To obtain an equivalent $\mathcal{C}$-circuit family $\{C'_n\}$ of $S(n + O(T(n)\log T(n)))^c$ size, define $C'_{|x|}(x) = D_{n_1}(C_{|x|}, x)$ for an appropriate length $n_1 \leq n + O(T(n)\log T(n))$. ■

Note if $S(n)$ and $T(n)$ are polynomials, then $S(n + O(T(n)\log T(n)))^c$ is also polynomial.

Reminder of Theorem 1.1 $\mathsf{NTIME}[2^n]$ *does not have non-uniform* ACC *circuits of polynomial size.*

Proof: First, we claim that if $\mathsf{NTIME}[2^n]$ has polysize ACC circuits, then every language in NEXP has polysize ACC circuits. A sketch of this implication is in the full version of the paper, but it is not hard to deduce.

By Lemma 5.1 and Theorem 5.1, it follows that SUCCINCT 3SAT has succinct satisfying assignments that are polynomial size ACC circuits. We claim that a contradiction can be obtained by examining the proof of Theorem 1.2 (the lower bound for E^{NP}). There, the only place requiring the full assumption "E^{NP} has non-uniform ACC circuits of size $S(n)$" is inside the proof of Theorem 3.2. In particular, the assumption is needed in Fact 3.2, where it is shown that for every satisfiable instance of SUCCINCT 3SAT, at least one of its satisfying assignments can be encoded in a size-$S(3n)$ ACC circuit. (The only other part of Theorem 3.2 where the assumption is applied is Lemma 3.1, but there it is only required that P has non-uniform ACC circuits.) But from the above, we already have that SUCCINCT 3SAT has succinct satisfying assignments which are ACC circuits.

The ACC CIRCUIT SAT instance D constructed in Theorem 3.2 with the witness circuit W has size polynomial in its $n+c\log n$ inputs. The Circuit SAT algorithm of Theorem 4.1 can determine satisfiability of any $n + c\log n$ input, n^c size ACC circuit in $O(2^{n-\log^2 n})$ time, for *every* c. Therefore unsatisfiability of D can be determined in $O(2^n/n^c)$ time for every constant c, and the desired contradiction follows from the nondeterministic time hierarchy. ■

It follows that problems complete under AC^0 reductions for NEXP such as SUCCINCT 3SAT require superpolynomial size ACC circuits.

VI. CONCLUSION

This paper demonstrates that it is possible to make progress in circuit complexity by designing faster algorithms for analyzing circuits. The reader is strongly urged to see the full version of this paper for many more results and details.

Further work will surely improve the results. Three natural next steps are: replace ACC with TC^0 circuits in the lower bounds, or replace NEXP with EXP, or extend the exponential lower bounds from E^{NP} to NEXP.

The results of Section III and Lemma 5.1 show that one only has to find a very minor improvement in algorithms for TC^0 satisfiability in order to establish non-uniform TC^0 lower bounds for NEXP. The author sees no serious impediment to the existence of such an algorithm. The evaluation lemma for SYM^+ circuits is key to the ACC SAT algorithm, and it would be very interesting to find similar lemmas for TC^0 or NC^1. It is plausible that the characterization of NC^1 as width-5 branching programs [Bar89] could lead to an analogous evaluation lemma for Boolean formulas, which would imply nontrivial depth lower bounds for NEXP. (Note that permutation branching programs of width 4 *can* be simulated in ACC [BT88], while width 5 captures NC^1.) Along the lines of the author's prior work [Wil10], Oded Goldreich and Or Meir (personal communication) observed that the consequence of Theorem 3.2 holds even when we replace $\mathcal{C}$-CIRCUIT SAT with the problem: *given an* n*-input* $S(n)$*-size* $\mathcal{C}$*-circuit, approximate its probability of acceptance on a uniform random input to within a* $1/6$ *additive factor*. It is widely believed that this problem can be solved in *polynomial time* for any reasonable $\mathcal{C}$.

It should be possible to extend the superpolynomial lower bound for ACC down to the class $\mathsf{QuasiNP} = \mathsf{NTIME}[n^{\log^{O(1)} n}]$. This paper comes fairly close to proving this result. The only step missing is a proof of the implication: "if QuasiNP has polynomial-size ACC circuits, then there are polynomial-size ACC circuits that encode witnesses to QuasiNP languages." A couple of lemmas rely only on P having non-uniform ACC circuits, so they could be potentially applied in proofs of even stronger lower bounds. At any rate, the prospects for future circuit lower bounds look very promising.

Acknowledgments. I am grateful to Virginia Vassilevska Williams for reading initial drafts of this work and reassuring me that they made sense. I also thank Miki Ajtai, Andreas Björklund, Ron Fagin, and Russell Impagliazzo for helpful discussions, and Eric Allender, Sanjeev Arora, Luke Friedman, Or Meir, Rahul Santhanam, Fengming Wang, and anonymous referees for helpful comments on earlier drafts. I especially thank Andreas for suggesting an alternative proof of Lemma 4.2.

REFERENCES

[AW09] S. Aaronson, A. Wigderson. Algebrization: A New Barrier in Complexity Theory. *TOCT* 1(1), 2009.

[AAD00] M. Agrawal, E. Allender, and S. Datta. On TC0, AC0, and arithmetic circuits. *JCSS* 60(2):395–421, 2000.

[Ajt83] M. Ajtai. Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic* 24:1–148, 1983.

[All99] E. Allender. The permanent requires large uniform threshold circuits. *Chicago J. Theor. Comput. Sci.*, 1999.

[AB87] N. Alon and R. B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica* 7(1):1–22, 1987.

[AB09] S. Arora and B. Barak. *Computational Complexity – a modern approach*. Cambridge University Press, 2009.

[AG94] E. Allender and V. Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.* 23(5):1026–1049, 1994.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential simulations unless EXPTIME has publishable proofs. *Comp. Compl.* 3:307–318, 1993.

[BGS75] T. P. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP Question. *SIAM J. Comput.* 4(4):431–442, 1975.

[Bar89] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. *JCSS* 38:150–164, 1989. See also STOC'86.

[BT88] D. A. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC1. *JACM* 35:941–952, 1988.

[BS95] D. A. Mix Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. *JCSS* 50:374–381, 1995.

[BST90] D. A. Mix Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Inf. and Comput.* 89:109–132, 1990.

[BT94] R. Beigel and J. Tarui. On ACC. *Comput. Compl.* 4:350–366, 1994. See also FOCS'91.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *IEEE CCC*, 8–12, 1998.

[CIP06] C. Calabro, R. Impagliazzo, and R. Paturi. A duality between clause width and clause density for SAT. In *IEEE CCC*, 252–260, 2006.

[CIP09] C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *IPEC*, 2009.

[Cau96] H. Caussinus. A note on a theorem of Barrington, Straubing, and Thérien. *IPL* 58(1):31–33, 1996.

[CGPT06] A. Chattopadhyay, N. Goyal, P. Pudlák, and D. Thérien. Lower bounds for circuits with MODm gates. In *IEEE FOCS*, 709–718, 2006.

[CW09] A. Chattopadhyay and A. Wigderson. Linear systems over composite moduli. In *IEEE FOCS*, 43–52, 2009.

[Coo88] S. A. Cook. Short propositional formulas represent nondeterministic computations. *IPL* 26(5):269–270, 1988.

[DH08] E. Dantsin and E. A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. van Maaren and T. Walsh (eds.), 341–362, 2008.

[FLvMV05] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *JACM* 52(6):835–865, 2005.

[FSS81] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Math. Syst. Theory* 17:13–27, 1984. Also in FOCS'81.

[GS89] Y. Gurevich and S. Shelah. Nearly linear time. *Logic at Botik '89*, Springer-Verlag LNCS 363, 108–118, 1989.

[Gree95] F. Green. Lower bounds for depth-three circuits with equals and mod-gates. In *STACS*, LNCS 900:71–82, 1995.

[GKRST95] F. Green, J. Köbler, K. W. Regan, T. Schwentick, and J. Torán. The power of the middle bit of a #P function. *JCSS* 50(3):456–467, 1995.

[Gro98] V. Grolmusz. A lower bound for depth-3 circuits with MOD m gates. *IPL* 67(2):87–90, 1998.

[GT00] V. Grolmusz and G. Tardos. Lower bounds for (MODp-MODm) circuits. *SIAM J. Comput.* 29(4):1209–1222, 2000.

[Han06] K. A. Hansen. Constant width planar computation characterizes ACC0. *Theory of Comput. Systems* 39(1):79–92, 2006.

[Hås86] J. Håstad. Almost optimal lower bounds for small depth circuits. *Adv. in Computing Research* 5:143–170, 1989. See also STOC'86.

[IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time versus probabilistic polynomial time. *JCSS* 65(4):672–694, 2002.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Compl.* 13(1-2):1–46, 2004.

[Kan82] R. Kannan. Circuit-size lower bounds and nonreducibility to sparse sets. *Inf. and Control* 55(1-3):40–56, 1982.

[KvM99] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Computing* 31(5):1501–1526, 2002. Also in STOC'99.

[KH09] M. Koucky and K. A. Hansen. A new characterization of ACC0 and probabilistic CC0. *Comput. Compl.* 19(2):211–234, 2010.

[KP94] M. Krause and P. Pudlák. On the computational power of depth 2 circuits with threshold and modulo gates. In *ACM STOC*, 48–57, 1994.

[MVW99] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *COCOON*, Springer LNCS 1627:210–220, 1999.

[PY86] C. H. Papadimitriou and M. Yannakakis. A note on succinct representations of graphs. *Inf. and Control* 71:181–185, 1986.

[PR81] W. J. Paul and R. Reischuk. On Time versus Space II. *JCSS* 22(3):312–327, 1981.

[Raz85] A. A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Doklady Akademii Nauk SSSR* 281(4):798–801, 1985.

[Raz87] A. A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Matematicheskie Zametki* 41(4):598–607, 1987.

[Raz89] A. A. Razborov. On the method of approximations. In *ACM STOC*, 169–176, 1989.

[RR97] A. A. Razborov and S. Rudich. Natural Proofs. *JCSS* 55(1):24–35, 1997.

[Rob91] J. M. Robson. An $O(T \log T)$ Reduction from RAM Computations to Satisfiability. *Theor. Comput. Sci.* 82(1):141–149, 1991.

[San10] R. Santhanam. Fighting perebor: new and improved algorithms for formula and QBF satisfiability. In *IEEE FOCS*, 183–192, 2010.

[Sch78] C.-P. Schnorr. Satisfiability is quasilinear complete in NQL. *J. ACM* 25(1):136–145, 1978.

[Sch05] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms* 54(1):40–44, 2005.

[SFM78] J. Seiferas, M. J. Fischer, and A. Meyer. Separating nondeterministic time complexity Classes. *JACM* 25:146–167, 1978.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *ACM STOC*, 77–82, 1987.

[Thé94] D. Thérien. Circuits constructed with MODq gates cannot compute AND in sublinear size. *Comput. Compl.* 4:383–388, 1994.

[Tou01] I. Tourlakis. Time-space tradeoffs for SAT on nonuniform machines. *JCSS* 63(2):268–287, 2001.

[Wil10] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. In *ACM STOC*, 231–240, 2010.

[Yao85] A. C.-C. Yao. Separating the polynomial-time hierarchy by oracles. In *IEEE FOCS*, 1–10, 1985.

[Yao90] A. C.-C. Yao. On ACC and threshold circuits. In *IEEE FOCS*, 619–627, 1990.

[YP94] P. Y. Yan and I. Parberry. Exponential size lower bounds for some depth three circuits. *Inf. and Comput.* 112:117–130, 1994.

[Zak83] S. Zak. A Turing machine hierarchy. *TCS* 26:327–333, 1983.