Amplifying Lower Bounds by Means of Self-Reducibility

ERIC ALLENDER
Rutgers University
and
MICHAL KOUCKÝ
Academy of Sciences of the Czech Republic

We observe that many important computational problems in NC^1 share a simple self-reducibility property. We then show that, for any problem A having this self-reducibility property, A has polynomial-size TC^0 circuits if and only if it has TC^0 circuits of size $n^{1+\epsilon}$ for every $\epsilon > 0$ (counting the number of wires in a circuit as the size of the circuit). As an example of what this observation yields, consider the Boolean Formula Evaluation problem (BFE), which is complete for NC^1 and has the self-reducibility property. It follows from a lower bound of Impagliazzo, Paturi, and Saks, that BFE requires depth d TC^0 circuits of size $n^{1+\epsilon_d}$. If one were able to improve this lower bound to show that there is some constant $\epsilon > 0$ (independent of the depth d) such that every TC^0 circuit family recognizing BFE has size at least $n^{1+\epsilon}$, then it would follow that $\mathrm{TC}^0 \neq \mathrm{NC}^1$. We show that proving lower bounds of the form $n^{1+\epsilon}$ is not ruled out by the Natural Proof framework of Razborov and Rudich and hence there is currently no known barrier for separating classes such as ACC^0, TC^0 and NC^1 via existing "natural" approaches to proving circuit lower bounds.

We also show that problems with small uniform constant-depth circuits have algorithms that simultaneously have small space and time bounds. We then make use of known time-space tradeoff lower bounds to show that SAT requires uniform depth d TC^0 and $\mathrm{AC}^0[6]$ circuits of size n^{1+c} for some constant c depending on d.

Categories and Subject Descriptors: F.1.3 [**Complexity Measures and Classes**]: Reducibility and completeness; F.1.3 [**Complexity Measures and Classes**]: Relations among complexity classes

General Terms: Theory

Additional Key Words and Phrases: Circuit Complexity, Lower Bounds, Natural Proofs, Self-Reducibility, Time-Space Tradeoffs

The work of E. Allender was supported in part by National Science Foundation (NSF) grants CCF-0514155, DMS-0652582, CCF-0830133, and CCF-0832787.

The work of M. Koucký was done in part while visiting DIMACS and Rutgers University partially supported by NSF grant CCF-0514703. Supported in part by grant GA ČR 201/07/P276, project No. 1M0021620808 of MŠMT ČR, Institutional Research Plan No. AV0Z10190503 and grant IAA100190902 of GA AV ČR.

A preliminary version of these results appeared in the Proceedings of the 23rd IEEE Conference on Computational Complexity.

Authors' addresses: E. Allender, Department of Computer Science, Rutgers University, New Brunswick, NJ 08855, e-mail: allender@cs.rutgers.edu; M. Koucký, Institute of Mathematics of the Academy of Sciences of the Czech Republic, Prague, Czech Republic, e-mail: koucky@math.cas.cz

1. INTRODUCTION

There is consensus in the research community that one of the most challenging and important open problems in computer science is to prove that various computational problems require large circuits in order to be computed. However, there is also a great deal of pessimism in the community, regarding the likelihood of proving such lower bounds on circuit size anytime in the near future. One goal of this paper is to suggest that there might be some reason to be more optimistic about prospects for circuit size lower bounds; we show that in certain settings, superpolynomial bounds would follow as a consequence of some very modest-sounding lower bound results (such as a lower bound of size $n^{1.0001}$). Of course, a confirmed pessimist would say that this is merely evidence that even these modest-sounding lower bounds are likely to remain beyond our reach.

1.1 The Quest for Circuit Lower Bounds

This paper focuses primarily on the task of proving superpolynomial lower bounds for various well-studied *restricted classes* of circuits, such as NC^1, TC^0, and CC$^0[6]$. The reader can find definitions of these classes in Section 2, along with a brief discussion of their importance and significance. Here, we recall just a few salient facts:

—Although it seems at first to be an absurdly weak class, CC$^0[6]$ (the class of problems that can be solved by constant-depth polynomial-size circuit families of MOD-6 gates) has not yet been shown to have less computational power than NP. Some theoreticians suspect that CC$^0[6]$ cannot even compute the AND function [Barrington et al. 1990; Hansen and Koucký]. Showing that AND (or any other problem in NP) lies outside of CC$^0[6]$ would constitute a significant advance in complexity theory.

—The "majority function" MAJ, which determines if more than half of the input bits are 1, is the canonical representative of the complexity class TC^0, consisting of the problems computed by constant-depth polynomial-size threshold circuits. Separating the complexity classes TC^0 and CC$^0[6]$ is equivalent to proving a superpolynomial lower bound on the size of CC$^0[6]$ circuits computing MAJ.

—NC^1 is the class of Boolean functions that can be represented by Boolean formulae of polynomial size. NC^1 has several natural problems that are complete under very restrictive notions of reducibility; we mention in particular the problem of evaluating a Boolean formula, which we denote by BFE. Separating the complexity classes NC^1 and TC^0 is equivalent to proving a superpolynomial lower bound on the size of constant-depth threshold circuits computing BFE.

The problem of separating these and other circuit complexity classes has remained open for more than two decades. This in itself would be cause for some discouragement about the prospects for progress. Additional grounds for despair were provided by Razborov and Rudich [Razborov and Rudich 1997], who showed that, if a class of circuits $\mathcal{C}$ is strong enough to compute pseudorandom function generators, then a wide variety of proof techniques are *incapable* of proving a given problem is too difficult to be computed by circuits in $\mathcal{C}$. Since there are constructions of pseudorandom function generators computable in TC^0 that are conjectured to be cryptographically secure [Naor and Reingold 2004], this has been viewed as constituting a significant barrier to progress on proving circuit lower bounds.

Although superpolynomial circuit size lower bounds have proved elusive, there has been significant work proving more modest lower bounds. For example, Håstad presents a nearly-cubic lower bound on the formula size for a certain function [Håstad 1998]. Nonlinear lower bounds on branching program size have been presented [Ajtai 1999; Beame et al. 2003]. The time-space tradeoff results that are surveyed by van Melkebeek [van Melkebeek 2007] give run-time lower bounds of the form n^c for small-space computations.

However, none of these lower bounds has led to superpolynomial lower bounds. More to the point, there was no expectation that a circuit size lower bound of the form n^c could *possibly* yield superpolynomial circuit bounds. In this paper, we show that there are several settings where precisely this sort of "amplification" *can* occur.

Moreover, in Section 8 we show that the work of Razborov and Rudich on "Natural Proofs" [Razborov and Rudich 1997] poses no barrier to proving weak lower bounds of the form n^c. This can be viewed as holding out some hope of separating circuit classes by proving circuit lower bounds using "natural" proof techniques.

1.2 Our Contributions

The main tool allowing us to obtain our results is self-reducibility of problems. We show that many problems in and around NC^1 (such as BFE, MAJ, AND, and many others) are *strongly downward self-reducible*. Then we show that, for any strongly downward self-reducible set, a lower bound of size n^c implies a superpolynomial size lower bound.[1]

In particular, we obtain the following corollaries:

Corollary *If there is some $\epsilon > 0$ such that* AND *requires* $\mathrm{CC}^0[6]$ *circuits of size $n^{1+\epsilon}$, then* AND $\notin \mathrm{CC}^0[6]$.

Corollary *If there is some $\epsilon > 0$ such that* MAJ *requires* $\mathrm{CC}^0[6]$ *circuits of size $n^{1+\epsilon}$, then* $\mathrm{CC}^0[6] \neq \mathrm{TC}^0$.

Corollary *If there is some $\epsilon > 0$ such that* BFE *requires* TC^0 *circuits of size $n^{1+\epsilon}$, then* $\mathrm{TC}^0 \neq \mathrm{NC}^1$.

Let us examine this third corollary more closely. It is interesting to recall that some non-linear lower bounds for BFE *are* known. Impagliazzo, Paturi, and Saks showed that any depth d TC^0 circuit for PARITY must have $n^{1+\epsilon_d}$ wires [Impagliazzo et al. 1997] (where $\epsilon_d = \Omega(1/(2.5)^d)$). Since there is a trivial reduction from PARITY to BFE (see the detailed definition of BFE in Section 2), the same size lower bound holds for BFE. In order to separate TC^0 from NC^1, it would suffice to improve this to a lower bound of size $n^{1+\epsilon}$ where ϵ does *not* depend on d.

One might reasonably wonder whether it is overly optimistic to expect to prove constant-depth circuit size lower bounds that do not depend on the depth d. Most circuit size lower bounds in the literature (such as those of [Furst et al. 1984; Yao 1985; Håstad 1988; Razborov 1987; Smolensky 1987]) do degrade with depth. For instance, the parity function requires depth d AC^0 circuits of size $2^{\omega(n^{1/(d-1)})}$, and this is nearly optimal [Håstad 1988]. However, it is important to note that there are exceptions to this trend; Rossman recently proved that, for every constant k, the k-clique problem requires AC^0 circuits with $\omega(n^{k/4})$ gates *independent* of the depth [Rossman 2008].

[1] A special case of this general observation (relating only to regular sets) also appears in a survey article by the second author [Koucký 2009]; the present article expands significantly on the related results of [Koucký 2009].

Clearly, no proof of $\mathrm{TC}^0 \neq \mathrm{NC}^1$ can follow from a PARITY lower bound such as the bound of Impagliazzo, Paturi, and Saks [Impagliazzo et al. 1997], and equally clearly, their argument does not yield a lower bound on the size of depth d $\mathrm{CC}^0[6]$ circuits computing BFE (since $\mathrm{CC}^0[6]$ circuits of linear size compute PARITY). In fact, there seem to be no known superlinear lower bounds for BFE on depth d $\mathrm{CC}^0[q]$ circuits for any q with at least two distinct prime factors. We now turn to the question of obtaining lower bounds for $\mathrm{CC}^0[q]$ and the related class $\mathrm{AC}^0[q]$, in order to discuss some of our other theorems.

Fortnow showed that SAT does not have logspace-uniform NC^1 circuits of size $n^{1+o(1)}$ [Fortnow 2000]. (Several improvements of this result of Fortnow are presented in [van Melkebeek 2004, Theorem 1.5].) Since we are able to show that modest lower bounds for BFE would yield superpolynomial lower bounds, it is natural to wonder if the same situation holds for SAT. That is, if one could build on the Fortnow lower bound, and show that SAT requires $\mathrm{AC}^0[6]$ circuits of size $n^{1.01}$, would it follow that $\mathrm{NP} \neq \mathrm{AC}^0[6]$? We know of no such implication — and in Section 5 we show that the approach that works for BFE cannot transfer directly to SAT. More specifically, in Section 5 we show that all strongly downward self-reducible sets lie in (uniform) NC. Thus, in order to demonstrate that SAT has the sort of self-reducibility properties that would enable us to amplify modest lower bounds to superpolynomial lower bounds, one would have to first prove that P=NP. (It is still conceivable that one could proceed by arguing that *if* $\mathrm{NP} = \mathrm{AC}^0[6]$, *then* SAT has the desired type of self-reduction, but we have not been able to construct such an argument.)

It is interesting to note that Srinivasan has shown [Srinivasan 2003] that an $\Omega(n^{1+\epsilon})$ lower bound on the running time of algorithms that compute weak approximations (of the form $n^{1-o(1)}$) to MAX-CLIQUE would imply $\mathrm{P} \neq \mathrm{NP}$. Using his techniques, we show in Section 6 that if $\mathrm{NP} = \mathrm{AC}^0[6]$, then there are $\mathrm{AC}^0[6]$ circuits of size $n^{1+o(1)}$ that compute $n^{1-o(1)}$-approximations to MAX-CLIQUE. We also use a similar argument to obtain lower bounds on the running time of any algorithm reducing MAX-CLIQUE to the problem of computing approximations of MAX-CLIQUE.

Even though we do not know how to separate NP from $\mathrm{AC}^0[6]$ by presenting a lower bound of the form n^c for the size of $\mathrm{AC}^0[6]$ circuits for SAT, we would nonetheless like to be able to present such a lower bound (as an illustration that current techniques can provide the sort of modest lower bounds that would separate NC^1 from $\mathrm{AC}^0[6]$ if such bounds could be proved for BFE). Although we can not provide such a lower bound, in Section 7 we do provide a lower bound analogous to the Impagliazzo, Paturi, and Saks bound mentioned above; we show that, for every d, there is a constant c_d such that depth d $\mathrm{AC}^0[6]$ circuits for SAT require size n^{1+c_d}.

2. PRELIMINARIES

2.1 Circuit complexity classes

This paper focuses on Boolean circuits and in particular on the circuit class NC^1 and its subclasses. Let us remind the reader of the main definitions, and present some notation. For more background on circuit complexity, the reader is referred to the text by Vollmer [Vollmer 1999].

For a function $f : \{0,1\}^* \rightarrow \{0,1\}$ and an integer $n \geq 1$, $f_n : \{0,1\}^n \rightarrow \{0,1\}$ is the restriction of f to inputs of size n.

We begin our discussion of circuits by considering a special case: formulas. A *Boolean* *formula* in n variables $x_1, x_2, \ldots, x_n$ is a rooted tree where each internal node is labeled by some function such as AND, OR or NOT and each leaf is labeled either by one of the input variables $x_1, \ldots, x_n$ or by a constant zero or one (false or true). Given an input $x \in \{0,1\}^n$, one can inductively assign a value to each node of the formula as follows: each leaf labeled by a variable gets the value of that variable, each leaf labeled by a constant gets the value of that constant, and each internal node gets the value of the function that labels it applied to the values of its children. In case where the function labeling a node is not symmetric the order of the children has to be specified. The value (*output*) of the formula on input x is the value of the root node. Hence a Boolean formula naturally computes a function $f : \{0,1\}^n \to \{0,1\}$. The nodes of the formula are generally referred to as *gates*. The in-degree of a gate is called its *fan-in*. In addition to the elementary functions AND, OR, and NOT, we will also consider gates computing the function MAJ (which evaluates to one if and only if the strict majority of its inputs is one) and the MOD-q function for an integer $q \geq 2$ (which is one if and only if the number of its inputs set to one is not divisible by q). The MOD-2 function will also be referred to as the PARITY function ($\oplus$ function). Sometimes we allow a more complex function to be computed by a gate; a node of a formula can be designated as an *oracle gate*. Typically all the oracle gates in a given formula will compute the same Boolean function $g : \{0,1\}^* \to \{0,1\}$, although we allow a single formula to have oracle gates for g_m and $g_{m'}$ for $m \neq m'$. The oracle should be viewed as a parameter for the formula; for a function g and formula ϕ with oracle gates, the *formula ϕ with oracle for g* is the formula ϕ where each oracle gate computes the function g. For a set A, an *oracle gate for A* is an oracle gate computing the characteristic function of A.

A *Boolean circuit* is a generalization of a formula where instead of a rooted tree we allow an arbitrary directed acyclic multi-graph. (We allow multiple edges (or *wires*) between nodes.) The nodes of out-degree zero are the output nodes. This way a circuit can compute a function $f : \{0,1\}^n \to \{0,1\}^m$, for integers $n, m \geq 1$. In circuits we also allow oracle gates to have several distinct output bits (wires) thus allowing us to have oracle gates for functions $g : \{0,1\}^m \to \{0,1\}^{m'}$ for $m' > 1$. (The tree-like nature of formulas imposes the restriction that $m' = 1$ in a formula.)

The *depth* of a circuit is the length of the longest path from an input node to an output node. The *size* of a circuit is the number of its wires, which is the number of edges in it. We will frequently refer also to the *number of gates* in a circuit,

A circuit computes a function on a fixed number of variables. To compute a function $f : \{0,1\}^* \to \{0,1\}$ by circuits we need an infinite family of circuits $\{C_n\}_{n \geq 1}$, where for each $n \geq 1$, circuit C_n computes f_n. One may abuse notation and say that *f is computable by circuits with property* $\gamma(n)$. Such an expression means that there is a family of circuits $\{C_n\}_{n \geq 1}$ where each C_n has property $\gamma(n)$ and computes f_n. Similarly, asymptotic statements should be interpreted with respect to the input size; e.g., *f is computable by polynomial-size constant-depth circuits* means that there is a circuit family $\{C_n\}_{n \geq 1}$, polynomial $p(n)$ and constant d, such that each C_n computes f_n and has size at most $p(n)$ and depth at most d. Similarly for formulas.

In addition to functions over the binary alphabet {0,1}, we also consider functions over an arbitrary alphabet Σ. In such cases we assume that there is some fixed encoding Bin $: \Sigma \to \{0,1\}^*$ of symbols from Σ into fixed-length binary strings; circuits for a function over the alphabet Σ operate on inputs encoded symbol-by-symbol by Bin. (The string homomorphism Bin extends in the obvious way to a function $\mathrm{Bin} : \Sigma^* \to \{0,1\}^*$.) Furthermore, a circuit for a function with non-Boolean output produces a binary encoding of the output symbol. The definitions of computability by circuits and of all the other terms extend naturally also to this case, however we only require that a circuit computing function f defined on Σ^* operate correctly on binary strings corresponding to binary encodings of strings from Σ^*. Thus, on inputs that do not correspond to binary encoded strings from Σ^*, the circuit may give an arbitrary output. For example a function $f : \Sigma^* \to \{0,1\}$ is computed by a circuit family $\{C_n\}_{n\geq 1}$ if for some $k \geq 1$ there is a binary encoding $\mathrm{Bin} : \Sigma \to \{0,1\}^k$ such that for each $n \geq 1$, and each input $x \in \Sigma^n$, $C_n(\mathrm{Bin}(x))$ outputs $f_n(x)$. In this case the size of the input is considered to be n although its binary encoding has length kn. Oracle gates for a function over an arbitrary alphabet Σ also operate on binary encoded strings from Σ, and on invalid inputs we assume that they output all zeros. (We state this convention only in order to make such oracle gates unambiguous; none of our results depends on it.)

A *language* A is a subset of Σ^* for some finite alphabet Σ. Every language naturally corresponds to its characteristic function $\chi_A : \Sigma^* \to \{0,1\}$ defined by $\chi_A(x) = 1$ if and only if $x \in A$. Vice versa, every function into $\{0,1\}$ corresponds to a language. We will identify languages with their characteristic functions. We say that *A is recognized* by $\{C_n\}_{n\geq 1}$ if its characteristic function is computable by $\{C_n\}_{n\geq 1}$.

This allows us to define the following classes of functions.

—NC^0 is the class of functions computable by polynomial-size constant-depth circuits built using fan-in two AND and OR gates and unary NOT gates.

—AC^0 is the class of functions computable by polynomial-size constant-depth circuits built using unbounded fan-in AND and OR gates and unary NOT gates.

—CC$^0[q]$ is the class of functions computable by polynomial-size constant-depth circuits having *only* unbounded fan-in MOD-q gates.

—AC$^0[q]$ is the class of functions computable by polynomial-size constant-depth circuits of unbounded fan-in AND and OR gates and unary NOT gates, along with unbounded fan-in MOD-q gates.

—CC^0 is the union of all the classes CC$^0[q]$ (for $q \geq 2$).

—ACC^0 is the union of all the classes AC$^0[q]$ (for $q \geq 2$).

—TC^0 is the class of functions computable by polynomial-size constant-depth circuits of unbounded fan-in MAJ gates and unary NOT gates.

—NC^1 is the class of functions computable by circuits of fan-in two AND and OR gates, and unary NOT gates, having depth $O(\log n)$.

—NC is the class of functions computable by circuits of fan-in two AND and OR gates, and unary NOT gates, having depth $O(\log^k n)$ for a constant k.

Some authors define these classes in terms of *languages* instead of *functions*, and use notation such as FAC^0 or FNC^1, etc., to refer to the associated class of functions. We prefer the simpler notation, and are confident that no confusion will result. We use the names of the function classes to denote also the corresponding circuit families; e.g., we refer to "AC^0 circuit families" or more succinctly to "AC^0 circuits".

As presented, these classes are *nonuniform*, i.e., it is not required that there be an easy way to construct the circuits for inputs of length n. We shall also need to consider logspace-uniform and Dlogtime-uniform versions of these classes [Barrington et al. 1990]. A circuit family $\{C_n\}_{n\geq 1}$ is *logspace-uniform* if there is a procedure that runs in logarithmic space and on input 1^n it outputs the description of C_n. A circuit family $\{C_n\}_{n\geq 1}$ is *Dlogtime-uniform* if there is a procedure that on input (n, i, r, j, s, t), where n, i, j, r, s are integers encoded in binary and t is a gate type (e.g., AND, OR, NOT, *oracle*, *input*, 0,1), runs in time linear in its input size and accepts if and only if the gate of C_n having label i is of type t and its r-th child is the s-th output bit of the gate labeled j. In case of the gate i being an input gate, the procedure accepts if gate i takes the value of the s-th input bit. Furthermore, the procedure accepts $(n, i, j, s, \mathrm{output})$ if and only if the s-th output bit of gate i is the j-th output bit of the circuit C_n. We also require that the procedure accepts the input (n, i, d) if and only if d is equal to the fan-in of the gate of C_n having label i; without this condition it is not always clear that Turing reducibilities defined in terms of uniform circuit families are closed under composition.[2] Thus, for example, Dlogtime-uniform AC^0 is the class of functions computable by Dlogtime-uniform families of AC^0 circuits, or more precisely, the class of functions computable by some Dlogtime-uniform family of circuits of polynomial-size and constant-depth that are built using unbounded fan-in AND and OR gates and unary NOT gates.

A string $w \in \{0,1\}^*$ of length n is the binary representation of an integer $m = \sum_{i=1}^{n} 2^{n-i} w_i$. The logarithm base two is denoted by $\log$.

We use the following convention throughout the paper. Whenever we refer to some real value a (such as $a = \log n$ or $a = n^\epsilon$) in a context where there should be an integral quantity (for instance: "a string of length a") the reader should read it as $\lceil a \rceil$.

2.2 Reductions and complete problems

The reader is probably familiar with the notion of polynomial-time many-one reducibility $\leq_m^p$. Polynomial-time reducibility is an extremely useful tool for classifying NP-complete problems and more generally for classifying the complexity of problems that are not believed to lie in P. However, it is of no use at all in identifying important distinctions among different problems in P. For that, it is necessary to use a more refined tool, such as AC^0 reductions.

For languages A and B we write $A \leq_m^{\mathrm{AC}^0} B$ if there is a function $r \in$ Dlogtime-uniform AC^0 such that, for all x, $x \in A$ if and only if $r(x) \in B$. More generally, for any class of functions $\mathcal{C}$, we say that $A \leq_m^{\mathcal{C}} B$ if there is a function $r \in \mathcal{C}$ such that for any x, $\chi_A(x) = \chi_B(r(x))$. The function r is called the *many-one reduction of A to B*. We say that A is complete for $\mathcal{D}$ under $\leq_m^{\mathcal{C}}$ reductions if $A \in \mathcal{D}$ and for any language $B \in \mathcal{D}$, $B \leq_m^{\mathcal{C}} A$. (Note, we require only *languages* to reduce to A; otherwise no language could be complete for a class that contains any non-Boolean function.) Although AC^0 reductions may seem to be quite restrictive, most natural examples of NP-complete problems remain complete under $\leq_m^{\mathrm{AC}^0}$ reductions. For any class $\mathcal{C}$ that is closed under $\leq_m^{\mathrm{TC}^0}$ reductions, e.g. NC^1, any language that is complete for $\mathcal{C}$ under $\leq_m^{\mathrm{AC}^0}$ reductions is also complete under $\leq_m^{\mathrm{NC}^0}$ reductions [Agrawal et al. 1998; Agrawal].

Note that we have defined $\leq_m^{\mathrm{AC}^0}$ reducibility for *uniform* reductions. If A is complete for any of the uniform circuit classes $\mathcal{C}$ that we consider under uniform $\leq_m^{\mathrm{AC}^0}$ reductions, then it is also complete for the corresponding *nonuniform* class under nonuniform $\leq_m^{\mathrm{AC}^0}$

[2] There are additional conditions that are required, in order to obtain a satisfactory definition of uniform NC^1; we refer the reader to the work of Ruzzo, who gives a uniformity condition with the desirable property that uniform NC^1 corresponds to logarithmic time on an alternating Turing machine [Ruzzo 1981].

reductions. (See the discussion before Proposition 2.3.)

NC^1 has several natural and important problems that are complete under $\leq_m^{\mathrm{AC}^0}$ reductions. We give detailed definitions of three such problems: the word problem over the permutation group S_5 on five elements [Barrington 1989], the Boolean Formula Evaluation problem [Buss 1987], and s-t-connectivity on directed graphs of width 5.

(1) *The word problem over* S_5. The word problem over the permutation group S_5 is the task of evaluating the product of a sequence of permutations. More generally we define the *word problem over a finite monoid* M first. A monoid M is a finite set with an associative binary operation and identity element 1_M. We denote the operation of M multiplicatively. The word problem over M is the function $W_M : M^* \to M$ that assigns to each sequence $m_1, m_2, \ldots, m_n$ of elements from M their product $m = m_1 m_2 \cdots m_n$ over M. The empty sequence ε is assigned the identity element 1_M. The word problem over S_5 is the word problem W_{S_5} over the permutation group $S_5 = \{$ permutations on five elements $\}$. The binary operation on S_5 is the composition of permutations; the identity element of S_5 is the identity permutation. (The word problem over S_5 can also be presented as a *language* consisting of those pairs $(m, (m_1, \ldots, m_n))$ for which $m = m_1 m_2 \cdots m_n$, and technically it is this language that is complete for NC^1 under $\leq_m^{\mathrm{AC}^0}$ reductions — but this language has the same complexity as the functional version of the problem that we have presented, and that version is more convenient to work with; working with the language would rather obscure things.)

(2) *The s-t-connectivity problem on directed graphs of width 5.* This is an NC^1-complete variant of s-t-connectivity. We say that a directed graph *is of width* k if its vertices can be partitioned into *layers* where each layer is of size at most k, the layers are linearly ordered and every edge goes from vertices of one layer to the vertices of the next layer. Every two consecutive layers of a width 5 directed graph form a bipartite graph and this bipartite graph can be represented by a 5×5 adjacency matrix. Thus a width 5 directed graph with $n + 1$ layers can be described by a sequence of n 5×5 adjacency matrices. The s-t-connectivity problem on directed graphs of width 5 is the problem of deciding whether a given vertex s in the first layer is connected by a path to a vertex t of the last layer in a width 5 directed graph. It is more convenient for us to work with the following functional version of connectivity (which has the same complexity as the decision problem), where we ask about connectivity between all vertices of the first and last layers. Let $\Sigma = \{0,1\}^{5\times 5}$ be the set of binary 5×5 matrices. We define W5-STCONN : $\Sigma^* \to \Sigma$ as follows. For $A_1, A_2, \ldots, A_n \in \Sigma$, W5-STCONN$(A_1 A_2 \ldots A_n) = A$, where $A \in \Sigma$ represents the connectivity between the first and last layer of a width 5 directed graph with $n + 1$ layers with adjacency matrices $A_1, A_2, \ldots, A_n$. It is a standard fact that A is equal to the product $A_1 A_2 \cdots A_n$ over the ring $(\{0,1\}, \mathrm{OR}, \mathrm{AND})$ – and this could also be taken as a formal definition of W5-STCONN. Moreover, one can view W5-STCONN as a word problem over the monoid Σ, where the binary operation is matrix multiplication over the ring $(\{0,1\}, \mathrm{OR}, \mathrm{AND})$ and the identity element of Σ is the identity matrix. This view of W5-STCONN will also be useful for us. Clearly, the word problem over S_5 is a special case of W5-STCONN.

(3) *The Boolean Formula Evaluation problem.* Roughly speaking, the Boolean Formula Evaluation problem is the set of formulas that evaluate to true. We will make use of its variant where we focus only on balanced formulas (that is, formulas whose graph is a

complete binary tree of depth d). Input instances thus consist of a string of 2^d zeros and ones representing the values that label the leaves of the formula, along with a sequence of $2^d - 1$ labels for the internal nodes of the tree. Let $\Sigma = \{0, 1, \wedge, \vee, \oplus\}$. The set BFE consists of all of the "well-formed formulas" over alphabet Σ that evaluate to 1.

In order to simplify the proof that our construction in Proposition 3.9 is Dlogtime-uniform, we choose a particular encoding that will be convenient. The "well-formed formulas" consist of strings of the form vx such that for some d, x is a string of length 2^d in $\{0, 1\}^*$, and v is a string of length $2^d - 1$ in $\{\wedge, \vee, \oplus\}^*$ representing the labels of the internal nodes of the formula, given in the order specified by the following recursive definition.

If $d = 1$, then there is only one internal node, so there is no need to specify the order.

If $d = 2$, then the label of the root is listed first, followed by the label of the left child, and then by the label of the right child.

If $d > 2$ and $d = 2c - 1$, then the $2^c - 1$ labels of the subtree T of depth c containing the root are given first, in the order specified for trees of depth c. This is followed by 2^c encodings of the subtrees of depth $c - 1$ whose values feed into T (starting from the leftmost subtree), in the order specified for trees of depth $c - 1$.

If $d > 2$ and $d = 2c$, then the $2^c - 1$ labels of the subtree T of depth c containing the root are given first, in the order specified for trees of depth c. This is followed by 2^c encodings of the subtrees of depth c whose values feed into T (starting from the leftmost subtree), in the order specified for trees of depth c.

The reader may wonder if it is necessary to be so particular about our encoding of the problem BFE. To some extent, the choice of encoding is crucial. For instance, if a formula were not encoded as a formula, but instead were encoded as an unsorted list of gates and edges, then it is an easy exercise to show that evaluating a formula is complete for L, using the fact that determining whether a vertex u occurs before a vertex v in a directed line graph presented as an unsorted list of edges is complete for L [Etessami 1997]. Thus it is at least important that the formulas in BFE be presented as parenthesized expressions or some similar encoding. The general (not-necessarily balanced) Boolean formula evaluation problem is in NC^1 [Buss 1987], and thus there are "efficient" $\leq_m^{\mathrm{AC}^0}$ reductions from the general formula evaluation problem to the balanced encoding that we have chosen for BFE, but the reductions that one obtains from known NC^1 algorithms (e.g., [Buss 1987; Buss et al. 1992; Buss 1993]) do not appear to be computable by *linear-size* AC^0 circuits. This is one reason why we do not know how to obtain *linear-size* strong downward self-reductions for the general Boolean formula evaluation problem, such as we present for BFE. The reason why we include $\oplus$ as an operation in BFE is so there will be a linear-size reduction from PARITY to BFE, so that the non-linear PARITY lower bounds [Impagliazzo et al. 1997] will immediately carry over to BFE.

Even in this restricted form, BFE is complete for NC^1. (See, for example, the proof of Lemma 7.2 in [Barrington et al. 1990].)

PROPOSITION 2.1. *[Barrington 1989; Buss 1987]* W_{S_5}, BFE, W5-STCONN *are problems complete for* NC^1 *under* $\leq_m^{\mathrm{NC}^0}$ *reductions.*

The problem W5-STCONN remains complete for NC^1 if directed edges are permitted *in both directions* between adjacent layers, as well as in the undirected case. The arguments that we present for W5-STCONN also carry over to these variants, with minor technical modifications.

Although NC^1 has several natural complete problems under $\leq_m^{\mathrm{AC}^0}$ reductions, many of the other complexity classes we consider (such as TC^0, $\mathrm{AC}^0[q]$, and ACC^0) are not believed to have *any* complete problems under $\leq_m^{\mathrm{AC}^0}$ reducibility. Some of them do, however, have complete problems under AC^0-Turing reducibility. Our main theorems rely on different variants of Turing reducibility, and thus we need the following general definition.

For any circuit complexity class $\mathcal{C}$, we define $\mathcal{C}$-Turing reducibility. Let f and g be two functions. We say that $f \leq_T^{\mathcal{C}} g$ if there is a family of circuits of polynomial size computing f, where the circuits have *oracle gates* for the function g in addition to the collection of gates that is provided in the definition of the circuit class $\mathcal{C}$.[3] The family of circuits is called the *reduction of f to g*. We say that f is complete for $\mathcal{D}$ under $\leq_T^{\mathcal{C}}$ reductions if $f \in \mathcal{D}$ and for any function $g \in \mathcal{D}$, $g \leq_T^{\mathcal{C}} f$. It is an easy observation that languages complete for NC^1 under $\leq_m^{\mathrm{AC}^0}$ reductions are also complete for NC^1 under $\leq_T^{\mathrm{AC}^0}$ reductions.

PROPOSITION 2.2. —MAJ *is complete for* TC^0 *under* $\leq_T^{\mathrm{NC}^0}$ *reductions.*

—MOD-q *is complete for* $\mathrm{AC}^0[q]$ *under* $\leq_T^{\mathrm{AC}^0}$ *reductions.*

—AND *is complete for* ACC^0 *under* $\leq_T^{\mathrm{CC}^0}$ *reductions.*

Turing reducibility will be used in the next section, in order to define downward self-reducibility.

Reductions can be either uniform or nonuniform. The reader can verify that all of the examples of reductions that we present in this paper are Dlogtime-uniform. It is worth observing that if A is complete for any of the *uniform* classes that we consider under uniform $\leq_m^{\mathrm{AC}^0}$ or $\leq_{\mathrm{T}}^{\mathcal{C}}$ reductions, then it is also complete for the corresponding *nonuniform* class under nonuniform reductions of the same type. For example, if B is in nonuniform NC^1, then there is a nonuniform family of Boolean formulae $\{\phi_n\}_{n\geq 1}$ accepting B. The set $D = \{(\psi, x)$: a Boolean formula ψ given in infix notation evaluates to 1 on $x\}$ is in uniform NC^1 [Buss 1987; Buss et al. 1992] and thus there is a uniform reduction from D to A. Composing this uniform reduction with the nonuniform reduction of B to D that maps x to $(\phi_{|x|}, x)$ yields the desired nonuniform reduction of B to A. Note that, for this example, it is important that B is presented in terms of Boolean formulae, instead of, say, logarithmic depth Boolean circuits, since it is not known whether logarithmic depth Boolean circuits can be evaluated in NC^1. A similar construction works also for constant-depth circuits. As an example we briefly explain the case of $\mathrm{CC}^0[q]$. If B is in nonuniform $\mathrm{CC}^0[q]$ then it is accepted by a family of $\mathrm{CC}^0[q]$ circuits of depth d and size at most n^k for some $k, d \geq 1$ and any $n \geq 2$. Without loss of generality one may also assume that these circuits are layered and between any two gates there is at most one wire. Consider the set $D = \{(C, x) : C$ is a $\mathrm{CC}^0[q]$ circuit of depth d encoded by a sequence of $|x|^k \times |x|^k$ adjacency matrices, one for each level of the circuit, such that $C(x) = 1\}$. Clearly, D has

[3] In this paper, we do not make use of NC^1-Turing reducibility, and indeed this definition would need to be modified in order to coincide with the definition of NC^1-Turing reducibility as studied by Cook [Cook 1985] and Wilson [Wilson 1990] and others. In defining AC^k reducibility, each oracle gate is considered to have depth 1, as in our definition, but in defining NC^k reducibility, Cook and Wilson felt that it was more in keeping with the flavor of bounded fan-in circuits to define the depth of an oracle gate to be the logarithm of its fan-in. Using their convention, an NC^0-Turing reduction could have oracle gates of only bounded fan-in, which is not a very useful notion. In contrast, our definition yields exactly the type of "NC^0-Turing reducibility" that we need in our definition of "pure self-reducibility".

uniform $\mathrm{CC}^0[q]$ circuits of depth $O(d)$ and size $n^{O(k)}$. Hence it reduces to A. Since B nonuniformly reduces to D by transitivity it also reduces nonuniformly to A.

Since completeness results carry over from the uniform setting to the nonuniform setting, we will henceforth slightly abuse notation and simply say that a set A is "complete under $\leq^{\mathcal{C}}_{\mathrm{T}}$ reductions" even when $\mathcal{C}$ is a nonuniform class, without explicitly mentioning that the reductions must be nonuniform in this case.

The following fact about Dlogtime-uniform Turing reductions is not entirely obvious, and thus for completeness we provide a proof. Let circuit family $\{C_n\}$ be a Turing reduction of f to g, and let $\{D_n\}$ be a Turing reduction of g to h. The *composition* of these reductions is the reduction of f to h that results by replacing each oracle gate of C_n having fan-in m by D_m.

PROPOSITION 2.3. *For any of the classes $\mathcal{C}$ defined in this section, the composition of two Dlogtime-uniform $\leq^{\mathcal{C}}_{\mathrm{T}}$ reductions is a Dlogtime-uniform $\leq^{\mathcal{C}}_{\mathrm{T}}$ reduction.*

PROOF. Let $\{C_n\}$ and $\{D_n\}$ be two Dlogtime-uniform families of reductions. Define a new family $\{E_n\}$ where E_n has the following gates:

$$\{i : i \text{ is a non-oracle gate of } C_n\} \cup$$

$$\{(i, m, j) : i \text{ is an oracle gate of } C_n \text{ that has fan-in } m \text{ and } j \text{ is a gate of } D_m\}.$$

Since the definition of Dlogtime-uniformity ensures that it is easy to recognize the fan-in of an oracle gate, it is routine to establish that the family E_n (with the obvious connections among gates to implement the composed reduction) is Dlogtime-uniform. For all of the polynomial-size circuit classes $\mathcal{C}$ defined in this section, it is immediate that the resulting reduction $\{E_n\}$ is also a $\leq^{\mathcal{C}}_{\mathrm{T}}$ reduction. □

3. DOWNWARD SELF-REDUCIBILITY

In this section we define downward self-reducibility and present several examples of downward self-reducible functions. Intuitively, a function is downward self-reducible if it can be efficiently computed from its own values at shorter inputs. We give a formal definition next.

A $\mathcal{C}$ *self-reduction for* f is a family of oracle circuits witnessing that $f \leq^{\mathcal{C}}_{T} f$, where on input x, the oracle circuit does not feed input x into any of its oracle gates.

Self-reducibility sometimes also goes by the name "autoreducibility." The term "self-reducibility" is more common in those settings (as here) where interest centers on routines that enforce the condition that x is not queried, by ensuring that all queries have length *shorter* than the length of x.

Definition 3.1. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, and let $\mathcal{C}$ be a class of circuits. Let $s(n), m(n) : \mathbb{N} \to \mathbb{N}$ be functions such that for all n, $m(n) < n$. We say that f_n *is downward self-reducible to* $f_{m(n)}$ *by a* $\mathcal{C}$ *reduction of size* $s(n)$ if there is a family of $\mathcal{C}$ oracle circuits $\{C_n\}_{n \geq 1}$ computing f such that for each n, C_n uses its oracle gates to query f on inputs of size at most $m(n)$, and has at most $s(n)$ wires.

Most of the self-reductions that we present consist of almost no hardware other than oracle gates. We call such reductions "pure"; a *pure self-reduction for* f is an NC^0 self-reduction for f, i.e., a self-reduction where the *only* gates are oracle gates, as well as bounded fan-in AND and OR gates and unary NOT gates.

Definition 3.2. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. Let $s(n), m(n) : \mathbb{N} \to \mathbb{N}$ be functions such that for all n, $m(n) < n$ and let $d \geq 1$ be an integer. We say that f_n *is downward self-reducible to* $f_{m(n)}$ *by a pure reduction of depth* d *and size* $s(n)$ if there is a circuit family $\{C_n\}_{n\geq 1}$ such that for each n, C_n computes f_n, is of depth at most d, size at most $s(n)$, and consists of fan-in two AND and OR gates, unary NOT gates and oracle gates that compute function f on inputs of size at most $m(n)$.

We use the term "pure" rather than simply calling them NC^0 reductions, since the term "NC^0" usually refers to computation in which the output depends on at most $O(1)$ bits of the input, and pure self-reductions do not share that property.

We will almost exclusively be interested in functions that are downward self-reducible to inputs of size at most $m(n) = n^\epsilon$, for some $\epsilon > 0$. This notion of downward self-reducibility is essentially identical to what Goldwasser *et al.* call "strong downward self-reducibility" [Goldwasser et al. 2007]. Hence, if f is downward self-reducible to f_{n^ϵ} by a pure reduction for some $\epsilon > 0$, we will also call it *strongly downward self-reducible*. (Similarly, if f is downward self-reducible to f_{n^ϵ} by a $\mathcal{C}$ reduction for some class $\mathcal{C}$, we will say that f is $\mathcal{C}$ strongly downward self-reducible.) For our purposes however, it is important to pay close attention to the size and depth of the reduction.

The rest of this section is devoted to showing that the following problems are strongly downward self-reducible: AND, W_M, MOD-q, W5-STCONN, MAJ and BFE. We also present somewhat weaker downward self-reducibility results for various types of iterated matrix multiplication problems.

We start with an example that may seem trivial, but is nonetheless useful.

PROPOSITION 3.3. *For any* $0 < \epsilon < 1$, AND$_n$ *is downward self-reducible to* AND$_{n^\epsilon}$ *by a Dlogtime-uniform pure reduction of depth* $O(1/\epsilon)$ *and size* $O(n)$. *Similarly for* OR.

PROOF. Consider the AND function. The idea of the proof is simple: form a tree of depth $O(1/\epsilon)$ from AND$_{n^\epsilon}$ gates and assign to each leaf one of the variables. However, ϵ and n may be arbitrary so this construction may not be uniform. Thus to provide a Dlogtime-uniform construction one has to be careful about the details. We provide a more detailed construction below to demonstrate the necessary techniques. A reader familiar with the issues of uniformity may want to skip the rest of the proof. Let an integer k satisfy $2^{k-1} < 1/\epsilon \leq 2^k$. If $n < 4^{2^k}$ then a tree of AND$_2$ gates can be used to compute AND$_n$. So assume for the rest of the proof that $n \geq 4^{2^k}$. Pick the largest integer $\ell \geq 1$ such that $2^\ell \leq n^{1/2^k}$ and the smallest integer m such that $n < (2^\ell)^m$. We will use AND$_{2^\ell}$ gates to build the circuit. We will label gates of the circuit by labels from $\{0,1,\ldots,m\} \times (\{0,1\}^\ell)^m$. Not all labels will be valid; some labels will be unused. We describe the valid labels together with the associated gates next. Let $n_1, n_2, \ldots, n_m \in \{0,1\}^\ell$ be such that $n_1 n_2 \cdots n_m$ is the ℓm-bit binary representation of $n-1$ padded with leading zeros if necessary. Consider $w_1, w_2, \ldots, w_m \in \{0,1\}^\ell$, $w = w_1 w_2 \cdots w_m$ and $d \in \{0,1,\ldots,m\}$. (d,w) represents a valid label in any of the following cases:

(1) If $d = 0$ and $w_1 = w_2 = \cdots = w_m = 0^\ell$ then (d,w) is the label of the single output AND$_{2^\ell}$ gate with inputs from gates labeled by $(1, w'_1 w_2 \cdots w_m)$ for any $w'_1 \in \{0,1\}^\ell$.

(2) If $0 < d < m$, $w_1 w_2 \cdots w_d$ precedes or is equal to $n_1 n_2 \cdots n_d$ in the lexicographical order and $w_{d+1} = w_{d+2} = \cdots = w_m = 0^\ell$ then (d,w) labels an AND$_{2^\ell}$ gate with inputs from gates labeled by $(d+1, w_1 w_2 \cdots w_d w'_{d+1} w_{d+2} \cdots w_m)$ for any $w'_{d+1} \in \{0,1\}^\ell$.

(3) If $0 < d \leq m$, $w_1 w_2 \cdots w_{d-1} = n_1 n_2 \cdots n_{d-1}$, n_d precedes w_d in the lexicographical order and $w_{d+1} = w_{d+2} = \cdots = w_m = 0^\ell$ then (d, w) labels a gate with constant one.

(4) If $d = m$, $w_1 w_2 \cdots w_m$ precedes or is equal to $n_1 n_2 \cdots n_m$ in the lexicographical order then (d, w) labels an input gate associated with the i-th input variable, where $i - 1$ is represented by w in binary.

No other label is used. Since ϵ is a constant, k is also a constant. One can verify easily from the description of the gate labeling that the connectivity language for the circuit with respect to this labeling is decidable by a Dlogtime procedure. (Given n in binary, one can find ℓ and m in time linear in the binary representation of n. Incrementing and decrementing a number in binary representation can also be done in time linear in the length of the binary representation. All other operations are clearly in linear time assuming our Dlogtime machine has at least two tapes.) One can also easily verify that the described circuit computes exactly AND_n.

We claim that it contains $O(n)$ wires. Indeed, the number of wires between bottom level of AND gates and inputs is at most $n + 2^\ell$. The layer one up contains at most $n/2^\ell + 2^\ell$ wires, the next one $n/(2^\ell)^2 + 2^\ell$, and so on. Thus the number of wires in the circuit is at most $2n + (m+1)2^\ell$. Since $\epsilon < 1$, we have that $k \geq 1$ and hence $2^\ell \leq \sqrt{n}$. Furthermore, $2^{\ell(m-1)} \leq n$ so, $m \leq 1 + \log n$. Thus the number of wires in the circuit is bounded by $2n + \sqrt{n} \cdot (2 + \log n)$. □

The case of AND and OR can be further generalized to word problems over finite monoids.

PROPOSITION 3.4. *For any finite monoid M and any $0 < \epsilon < 1$, $(W_M)_n$ is downward self-reducible to $(W_M)_{n^\epsilon}$ by a Dlogtime-uniform pure reduction of depth $O(1/\epsilon)$ and size $O(n)$.*

The proof is essentially the same as for AND and OR; one uses gates computing W_M on inputs of size $\leq n^\epsilon$ and constants for the binary encoding of 1_M. If for an integer $q \geq 2$ we consider the monoid $Z_q = (\{0, 1, \ldots, q-1\}, +(\bmod q))$ then we obtain the next corollary.

COROLLARY 3.5. *For any $0 < \epsilon < 1$, $(\text{MOD-}q)_n$ is downward self-reducible to $(\text{MOD-}q)_{n^\epsilon}$ by a Dlogtime-uniform pure reduction of depth $O(1/\epsilon)$ and size $O(n)$.*

PROOF. Clearly, MOD-q can be computed using W_{Z_q}. The other way around is also true: one can compute W_{Z_q} using MOD-q. The proof of the corollary consists of showing how $(W_{Z_q})_\ell$ can be computed using gates for $(\text{MOD-}q)_\ell$ and then applying the previous proposition on W_{Z_q}. A reader familiar with the issue of conversion between $(\text{MOD-}q)_\ell$ and W_{Z_q} may want to skip the rest of the proof.

Let $b \geq 1$ be a constant, Bin : $Z_q \to \{0,1\}^b$ be an arbitrary injective function, and $\ell > 4q$ be an integer. We show how to use $(\text{MOD-}q)_\ell$ gates to compute $(W_{Z_q})_\ell$ encoded by Bin. Let $x_1, x_2, \ldots, x_\ell \in Z_q$ be an input to W_{Z_q} and $y_1, \ldots, y_{\ell b}$ be its encoding by Bin. We will build a circuit that takes $y_1, \ldots, y_{\ell b}$ as its input and outputs $z_1 \cdots z_b$, where $z_1 \cdots z_b$ is the encoding of $\sum_{i=1}^{\ell} x_i (\bmod q)$ by Bin. The circuit will be of constant depth (depending only on Bin and q) and use $O(\ell)$ bounded fan-in AND, OR and NOT gates and q MOD-q gates of fan-in ℓ. The circuit computes as follows.

Let $m = \lceil \ell/q \rceil - 2$. Partition arbitrarily $\{1, \ldots, \ell\}$ into non-empty sets $S_1, S_2, \ldots, S_m$ of size at most $2q$. Since $\ell > 4q$, this is possible. For each $i \in \{1, \ldots, m\}$, let $w_i =$

$1^{\sum_{j\in S_i} x_j (\mathrm{mod}\, q)}0^{q-\sum_{j\in S_i} x_j (\mathrm{mod}\, q)}$. Clearly, $\sum_{i=1}^{m}\sum_{j=1}^{q}(w_i)_j \equiv \sum_{i=1}^{\ell} x_i (\mathrm{mod}\, q)$. As q and b are constant, $w_1 w_2 \cdots w_m$ can be computed from $y_1, \ldots, y_{\ell b}$ by a circuit of constant depth using $O(\ell)$ fan-in two AND and OR gates and unary NOT gates.

For $j = 0, \ldots, q-1$, let g_j be a MOD-q gate of fan-in ℓ that takes as its input the string $w_1 w_2 \cdots w_m 0^j 1^{q-j}$ padded with zeros to the length of ℓ. Clearly, g_j evaluates to zero if and only if $\sum_{i=1}^{\ell} x_i = j (\mathrm{mod}\, q)$. Hence, the output of $g_0, \ldots, g_{q-1}$ uniquely determines $\sum_{i=1}^{\ell} x_i (\mathrm{mod}\, q)$. The output of $g_0, \ldots, g_{q-1}$ can thus be processed by a constant size circuit consisting of bounded fan-in AND, OR and NOT gates to compute Bin$(\sum_{i=1}^{\ell} x_i (\mathrm{mod}\, q))$. This gives the desired circuit for computing $(W_{Z_q})_\ell$ encoded by Bin. (For $\ell < 4q$ one can build a constant-depth circuit computing $(W_{Z_q})_\ell$ using fan-in two AND gates and unary NOT gates.)

By Proposition 3.4, there are constants b, k and a function Bin : $Z_q \rightarrow \{0,1\}^b$ such that for all n large enough, there is a circuit C_n of depth $\leq k/\epsilon$ with $\leq kn$ wires that computes $(W_{Z_q})_n$ encoded by Bin using fan-in two AND and OR gates, unary NOT gates and gates for $(W_{Z_q})_\ell$ encoded by Bin, for $\ell \leq n^\epsilon$. Take C_n and replace each gate for $(W_{Z_q})_\ell$ by the circuit constructed in the preceding paragraph to obtain a circuit C'_n computing $(W_{Z_q})_n$. The circuit C'_n consists of fan-in two AND and OR gates, unary NOT gates and $(\text{MOD-}q)_\ell$ gates, for $\ell \leq n^\epsilon$. Since each $(W_{Z_q})_\ell$ gate of fan-in ℓb is replaced by a constant-depth circuit that uses $O(\ell)$ wires, the depth and the number of wires of C'_n are only a constant factor larger than that of C_n.

If we encode an input $v_1, v_2, \ldots, v_n \in \{0,1\}$ symbol by symbol by Bin and we feed the resulting string into the circuit C'_n we obtain $\sum_{i=1}^{n} v_i = j (\mathrm{mod}\, q)$ encoded by Bin. From this encoded value one can decode the output of MOD-q on input $v_1, \ldots, v_n$. Hence using $O(n)$ additional fan-in two AND and OR gates and unary NOT gates one can convert the circuit C'_n into a constant-depth circuit for $(\text{MOD-}q)_n$. The overall size of the circuit will be linear in n.

One can verify that the construction can be made Dlogtime-uniform. Indeed, the circuit computing W_{Z_q} using MOD-q gates can be made Dlogtime-uniform, and its gate labeling can be concatenated with the labeling of gates in C_n to obtain a gate labeling of C'_n. Additional labels can be used for gates calculating the Bin encoding and decoding of input and output of C'_n. The details of these constructions are rather straightforward and we leave them to the interested reader. □

Because of the connection between W5-STCONN and word problems over monoids we also obtain:

PROPOSITION 3.6. *For any* $0 < \epsilon < 1$, W5-STCONN$_n$ *is downward self-reducible to* W5-STCONN$_{n^\epsilon}$ *by a Dlogtime-uniform pure reduction of depth* $O(1/\epsilon)$ *and size* $O(n)$.

We can prove a similar self-reducibility claim also for MAJ. This time the proof is a little bit more involved and uses the following lemma.

LEMMA 3.7. *There is a constant* d *such that for any* $1 \leq \ell \leq m$ *there is a Dlogtime-uniform depth* d *circuit family with at most* $4\ell m$ *gates (consisting of fan-in two* AND *gates, unary* NOT *gates and oracle gates for* MAJ$_{2m}$*), taking as its input* ℓ*-bit binary representations of integers* $y_1, \ldots, y_m$*, and producing as output a sequence of integers* $z_1, \ldots, z_\ell$ *each represented by* $\ell + \log(m+1)$ *bits such that* $y_1 + y_2 + \cdots + y_m = z_1 + \cdots + z_\ell$.

PROOF. First, observe that we can compute $\textsc{And}_m$ and $\textsc{Or}_m$, using a gate for $\textsc{Maj}_{2m}$ and constants 0 and 1. For $j = 1, \ldots, \ell$, let $z_j = 2^{\ell-j} \sum_{i=1}^{m} (y_i)_j$. It follows from the definition of binary representation that $y_1+y_2+\cdots+y_m = z_1+\cdots+z_\ell$. Thus we only need to show how to compute the $(\ell + \log(m+1))$-bit binary representation of $z_1, \ldots, z_\ell$ from the binary representation of $y_1, \ldots, y_m$. (Note each of the representations of $z_1, \ldots, z_m$ has ℓ of its bits always set to zero.)

Fix $j \in \{1, \ldots, \ell\}$. Let $g_1, \ldots, g_m$ be $\textsc{Maj}_{2m}$ gates where for $k = 1, \ldots, m$, g_k takes as its input $(y_1)_j (y_2)_j \cdots (y_m)_j 1^{m-k+1} 0^{k-1}$. Clearly, the output $g_1 \cdots g_m$ is equal to $1^{\sum_{i=1}^{m} (y_i)_j} 0^{m-\sum_{i=1}^{m} (y_i)_j}$. For $k = 1, \ldots, m-1$, let $g'_k = (g_k\ \textsc{And}\ (\textsc{Not}\ g_{k+1}))$ and $g'_m = g_m$. Then g'_k evaluates to one if and only if $\sum_{i=1}^{m} (y_i)_j$ is k. Hence, the h-th bit of the $\log(m+1)$-bit binary representation of $\sum_{i=1}^{m} (y_i)_j$ is obtained by taking the $\textsc{Or}$ of gates in the set $\{g'_k;\ k \in \{1, \ldots, m\}$ and the h-th bit of the $\log(m+1)$-bit binary representation of k is one$\}$. Thus to obtain the $\log(m+1)$-bit binary representation of $\sum_{i=1}^{m} (y_i)_j$ we need m $\textsc{Maj}_{2m}$ gates, $m-1$ $\textsc{And}_2$ gates, $m-1$ $\textsc{Not}$ gates and $\log(m+1)$ $\textsc{Or}_m$ gates. Hence, to obtain all the $\log(m+1)$-bit representations of $\sum_{i=1}^{m} (y_i)_j$ for different j's we need at most $4\ell m$ (non-input) gates in total. The desired binary representation of z_j is obtained by concatenating the binary representations of $\sum_{i=1}^{m} (y_i)_j$ with 0^j on the left and $0^{\ell-j}$ on the right. Dlogtime-uniformity of the circuit is routine to establish. □

THEOREM 3.8. *For any $0 < \epsilon < 1$, $\textsc{Maj}_n$ is downward self-reducible to $\textsc{Maj}_{n^\epsilon}$ by a Dlogtime-uniform pure reduction of depth $O(1/\epsilon)$ and size $O(n^{1+\epsilon})$ consisting of a linear number of gates.*

PROOF. First, we prove the claim for $\epsilon = 1/2$ to illustrate the technique. For simplicity and clarity we mostly ignore rounding issues. We can view the input as n 1-bit integers $a_1, \ldots, a_n$. To determine the output of $\textsc{Maj}_n$ we will compute the binary representation of the sum of these integers. The total sum will be obtained in several stages. Each stage will take as an input a sequence $a_1, a_2, \ldots, a_m$ of integers and convert it into a shorter sequence of integers $b_1, b_2, \ldots, b_{m'}$ having the same sum, i.e., $m' < m$ and $a_1 + a_2 + \cdots + a_m = b_1 + \cdots + b_{m'}$. The first stage will start with the input as a sequence of 1-bit integers and the last stage will output a single integer representing the total sum of the input bits. As no integer at any stage can attain a value larger than n we can always truncate any number of more than $\log(n+1)$-bits to the $\log(n+1)$ least significant bits. (If convenient we may also pad a binary representation of any number by leading zeros to $\log(n+1)$-bits.)

Stage 1: $n \times 1$**-bit** → $(2\sqrt{n}+6) \times \log(n+1)$**-bits.** This stage takes a sequence of bits $a_1, a_2, \ldots, a_n$ and outputs $b_1, b_2, \ldots, b_{2\sqrt{n}+6}$. Partition $\{1, \ldots, n\}$ into sets S_1, $S_2, \ldots,$ $S_{2\sqrt{n}+6}$ of consecutive integers, each set of size at most $\sqrt{n}/2$. Using Lemma 3.7 compute for each $j = 1, \ldots, 2\sqrt{n}+6$ in parallel the integer $b_j = \sum_{i \in S_j} a_i$. Output $b_1, \ldots, b_{2\sqrt{n}+6}$ padded to $\log(n+1)$-bits. Lemma 3.7 (with $\ell = 1$ and $m = \sqrt{n}/2$) provides a circuit with at most $2\sqrt{n}$ gates for computing each b_j, so in total we need $O(n)$ of $\textsc{And}_2$, $\textsc{Not}$ and $\textsc{Maj}_{\sqrt{n}}$ gates for this stage.

Stage 2: $(2\sqrt{n}+6) \times \log(n+1)$**-bits** → $5\lceil\log(n+1)\rceil \times \log(n+1)$**-bits.** This stage takes a sequence of $\log(n+1)$-bit integers $a_1, a_2, \ldots, a_{2\sqrt{n}+6}$ and outputs $b_1, b_2, \ldots, b_{5\lceil\log(n+1)\rceil}$. Similarly to the previous stage divide the a_i's into five subsequences of size at most $\sqrt{n}/2$ and using Lemma 3.7 (with $\ell = \log(n+1)$ and $m = \sqrt{n}/2$, truncating the outputs to $\log(n+1)$ bits), compute for each of the subsequences $\log(n+1)$-bit integers representing the sum of the a_i's in that subsequence. Output all the $5\lceil\log(n+1)\rceil$ integers that were obtained from the application of the lemma. Since each subsequence contains at most $\sqrt{n}/2$ integers, this stage requires at most $O(\sqrt{n}\log n)$ many AND$_2$, NOT and MAJ$_{\sqrt{n}}$ gates.

Stage 3: $5\lceil\log(n{+}1)\rceil\times\log(n{+}1)$**-bits** $\rightarrow 3\times\log(n{+}1)$**-bits.** This stage gets $\log(n{+}1)$-bit integers $a_1, a_2, \ldots, a_{5\lceil\log(n+1)\rceil}$ and outputs b_1, b_2, b_3. It proceeds as follows. We divide the binary representation of each a_i, $i\in\{1,\ldots,5\lceil\log(n+1)\rceil\}$, into blocks of $\log\log n$ consecutive bits. Each block is regarded as a $\log\log n$-bit integer so we get integers $a_{i,1}, a_{i,2}, \ldots, a_{i,k}$, where $k\approx\log(n+1)/\log\log n$ and $a_i=\sum_{j=1}^{k}2^{(k-j)\lceil\log\log n\rceil}a_{i,j}$.

For $j=1,\ldots,k$, we apply Lemma 3.7 on the sequence $a_{1,j}, a_{2,j}, \ldots, a_{5\lceil\log(n+1)\rceil,j}$ to obtain $d_{j,1}, d_{j,2}, \ldots, d_{j,\log\log n}$, where $\sum_{i=1}^{5\lceil\log(n+1)\rceil}a_{i,j}=\sum_{i=1}^{\log\log n}d_{j,i}$. Denote the value of the sum s_j. The sequence $d_{j,1}, d_{j,2}, \ldots, d_{j,\log\log n}$ represents $\log\log n$ integers, each having $O(\log\log n)$ bits. Hence altogether they constitute only $O(\log\log n)^2$ bits. Hence, using a DNF formula of size $2^{O(\log\log n)^2}\le n^{o(1)}$ built from $n^{o(1)}$ AND$_{O(\log\log n)^2}$ gates and a single OR$_{n^{o(1)}}$ gate we can calculate from the sequence $d_{j,1}, d_{j,2}, \ldots, d_{j,\log\log n}$ each bit of the binary representation of s_j. For all j we do the whole calculation in parallel.

Finally, each s_j represents a sum of at most $5\lceil\log(n{+}1)\rceil$ integers each of $\log\log n$-bits, so it can be represented by $\log(5\lceil\log(n+1)\rceil+1)+\log\log n\le 5+2\log\log n$ bits. We can form three integers b_1, b_2, b_3 from $s_1,\ldots,s_k$ that represent the sum of the a_i's (see Fig. 1). Formally, $b_i=\sum_{j\equiv 4-i \bmod 3}2^{(k-j)\lceil\log\log n\rceil}s_j$, where j ranges from 1 to k.

This stage involves $O(\log n/\log\log n)$ applications of Lemma 3.7 with parameters ℓ and m of order less than $\log n$, and $k(5+2\log\log n)$ DNF formulas of size $n^{o(1)}$. Hence, it can be implemented by a constant-depth circuit consisting of a linear number of AND$_2$, NOT and MAJ$_{n^{o(1)}}$ gates.

Stage 4: $3\times\log(n{+}1)$**-bits** $\rightarrow 1\times\log(n{+}1)$**-bits.** Adding two n-bit integers can be done by AC^0 circuits using $O(n^2)$ many AND$_n$, OR$_n$ and NOT gates (see, e.g. [Vollmer 1999, Theorem 1.15]). Hence, adding three $\log(n{+}1)$-bit integers can be done by constant-depth circuits using $O(\log^2 n)$ many AND$_{\log(n+1)}$, OR$_{\log(n+1)}$ and NOT gates. Thus summing the input a_1, a_2, a_3 of this stage can be done by a constant-depth circuit using $O(\log^2 n)$ many MAJ$_{O(\log n)}$ and NOT gates to obtain the final sum.

The resulting total sum obtained from Stage 4 of the circuit can be compared with the binary representation of $n/2$ by an AC^0 circuit consisting of $O(\log^2 n)$ many AND$_{\log(n+1)}$, OR$_{\log(n+1)}$ and NOT gates or alternatively MAJ$_{O(\log n)}$ and NOT gates. As each stage of the computation can be done by constant-depth circuits consisting of a linear number of AND$_2$, NOT and MAJ$_{\sqrt{n}}$ gates the lemma follows for $\epsilon=1/2$.

For general ϵ the computation proceeds similarly, but the first two stages are replaced by a repeated use of a stage that reduces the input sequence $a_1, a_2, \ldots, a_m$ to the sequence $b_1, b_2, \ldots, b_{m'}$, for $m'=(2m\log(n+1))/n^\epsilon$. The reduction is done using Lemma 3.7 applied on subsequences of a_i's of length $n^\epsilon/2$. Once $m'\le n^\epsilon/2$, a single application of Lemma 3.7 produces $\log(n+1)$ integers that can be passed to the last two stages of the above procedure. Clearly, $2+1/\epsilon$ repetitions will suffice for the first stage, each repetition requiring at most $n^{1-\epsilon}\cdot O(n^\epsilon)=O(n)$ gates for MAJ$_{n^\epsilon}$, AND$_2$ and NOT.

We have established that the self-reductions have a linear number of gates, but it remains for us to prove the size bound of $O(n^{1+\epsilon})$ by counting the number of wires. There are $O(n)$

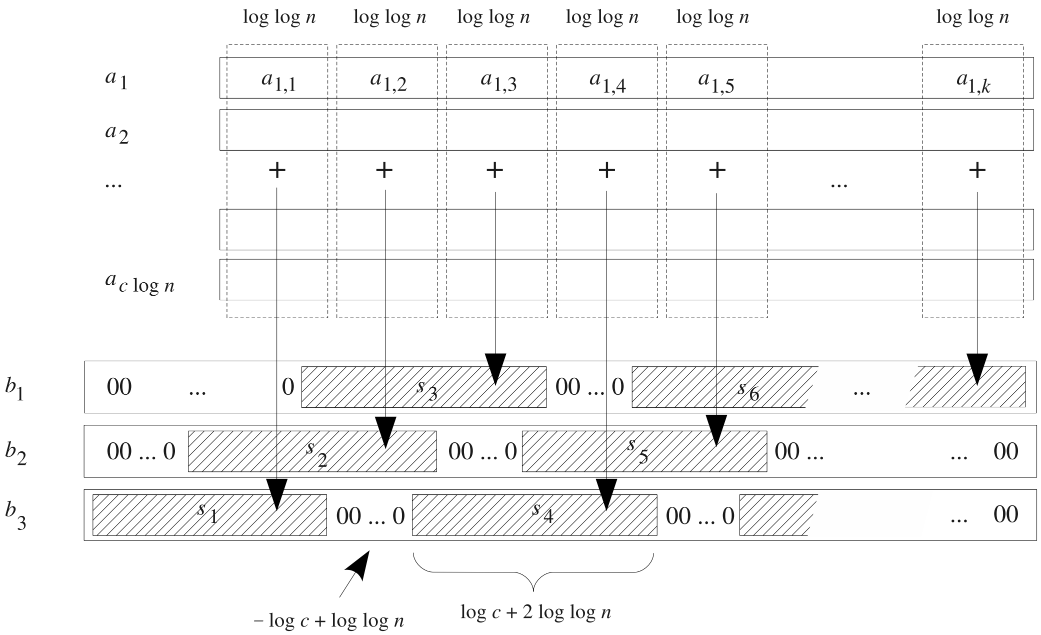

Fig. 1. Adding $(k \log\log n)$-bit integers $a_1, a_2, \ldots, a_{c \log n}$: $b_1 + b_2 + b_3 = a_1 + a_2 + \cdots + a_{c \log n}$.

gates, each having fan-in at most n^ϵ. Thus the total size is $O(n^{1+\epsilon})$. Dlogtime-uniformity of the circuit is routine to establish. □

We have seen that AND, OR, MOD-q, and MAJ are all downward self-reducible. We saw also that downward self-reducibility holds for the word problem over any finite monoid, which yields self-reductions for some of the standard complete problems for NC^1: the word problem over S_5, and W5-STCONN. We thank Mario Szegedy for pointing out that BFE (another standard complete problem for NC^1) is also downward self-reducible:

PROPOSITION 3.9. *For any* $0 < \epsilon < 1$, BFE$_n$ *is downward self-reducible to* BFE$_{n^\epsilon}$ *by a Dlogtime-uniform pure reduction of depth* $O(1/\epsilon)$ *and size* $O(n)$.

PROOF. We will show that there is a constant c and an oracle circuit family $\{C_n\}_{n \geq 1}$ such that C_n is a pure reduction of depth c and size $O(n)$ reducing BFE$_n$ to BFE$_{4n^{1/2}}$, where no path from a leaf to the root of C_n encounters more than two oracle gates.

We first show that this suffices to prove the proposition. Note that if we replace each oracle gate for BFE$_m$ in C_n by the oracle circuit C_m, we obtain a Dlogtime-uniform family of pure reductions of depth $3c$ and size $O(n)$ reducing BFE$_n$ to BFE$_{16n^{1/4}}$, where no path from a leaf to the root of C_n encounters more than four oracle gates. (Notice, each oracle gate for BFE$_m$ uses $O(m)$ wires and is replaced by a circuit having also $O(m)$ wires. Thus, the size of the circuit gets at most multiplied by some constant.) By induction, we obtain, for every k, a Dlogtime-uniform family of pure reductions of depth $(2^k - 1)c$ and size $O(n)$ reducing BFE$_n$ to BFE$_{4^k n^{1/2^k}}$.

Thus, in particular, for ϵ of the form $1/2^{k-1}$ there is a Dlogtime-uniform family of pure reductions of depth $(2^k - 1)c = O(1/\epsilon)$ and size $O(n)$ reducing BFE$_n$ to BFE$_{n^\epsilon}$, since $4^k n^{1/2^k} < n^{1/2^{k-1}}$ for all large n. The theorem follows, since every ϵ is within a factor of 2 of some smaller number of the form $1/2^{k-1}$.

We now proceed to prove the claim, by presenting the circuit family $\{C_n\}$. BFE contains only inputs of length n of the form $n = 2^{d+1} - 1$ for some integer d, so assume n has this form. Assume that d is odd; the construction is simpler if d is even. Let us denote the first $2^d - 1$ input symbols by v, and the last 2^d input symbols by x.

The output gate of C_n will be an AND gate of fan-in two, where one child a checks if the input is a well-formed formula, and the other child b evaluates the formula, assuming that it is well-formed. We consider b first.

The gate b is an oracle gate that has as its input the string $v'x'$, where v' consists of the first $2^{(d+1)/2} - 1$ symbols of v, and x' is a string of $2^{(d+1)/2}$ symbols consisting of the outputs of oracle gates b_i, for $1 \leq i \leq 2^{(d+1)/2}$. If the input string vx is well-formed, the string v' encodes the subformula of the formula v having depth roughly half of the depth of v and containing the output gate of v, and the oracle gates b_i will evaluate the subformulas of v that feed into v'. More precisely, the oracle gate b_i will take as input a string (v_i, x_i), where v_i is the i-th block of length $2^{(d-1)/2} - 1$ after v' in v, and x_i is the i-th block of length $2^{(d-1)/2}$ in x. It is immediate that the gate b produces the desired output, if the input is a well-formed formula. A routine calculation shows that the queries have length bounded by $4n^{1/2}$.

We now turn to the construction of the subcircuit a that tests if the input is well-formed. Recall that the input is well-formed if and only if $v \in \{\wedge, \vee, \oplus\}^*$ and $x \in \{0,1\}^*$. This is simply an AND of n conditions (call them c_i), where each condition c_i can be computed using NC^0 circuitry. We need to evaluate this AND using oracle gates for BFE$_m$ where $m \leq 4n^{1/2}$. To do this, we first use another layer of NC^0 circuitry to halve the fan-in of the unbounded fan-in AND that we need to compute; we compute conditions c'_j defined by $c'_j = c_{2j-1} \wedge c_{2j}$ for $j = 1, \ldots, (n-1)/2$ and $c'_{(n+1)/2} = c_n$. Note that the input is well-formed if and only if BFE$(v''x'')$ evaluates to true, where x'' consists of the bits c'_j and $v'' = \wedge^{(n-1)/2}$. This well-formed instance of BFE can be evaluated using queries to BFE$_m$ for $m \leq 4n^{1/2}$, using the same construction as was used for the gate b, to evaluate a formula.

To complete the proof, we merely observe that the number of wires is easily seen to be linear in n, and we note that Dlogtime-uniformity is routine to establish. □

Indeed, we point out that any problem that is complete for a complexity class that has a strongly downward self-reducible complete problem must be strongly downward self-reducible. See Proposition 5.3.

Another problem for which we can prove downward self-reducibility is *Iterated Matrix Multiplication*. Let IMM$_{n,d,\ell} : \{0,1\}^{nd^2\ell} \to \{0,1\}^{d^2 n(\ell + \log d)}$ be the problem of computing the product of n $d \times d$ matrices, with each entry being a non-negative ℓ-bit integer. Define the *modular* version of the Iterated Matrix Product to be the function mIMM$_{n,d,q} : \{0,1\}^{nd^2 \log q} \to \{0,1\}^{d^2 \log q}$ computing the Iterated Matrix Product modulo some integer $q \geq 2$. Finally, we will also need to consider the *Boolean* Iterated Matrix Product problem BIMM$_{n,d} : \{0,1\}^{nd^2} \to \{0,1\}^{d^2}$ which is the Iterated Matrix Problem over the ring $(\{0,1\}, \text{OR}, \text{AND})$.

The following proposition is immediate using the same technique as in Proposition 3.3:

PROPOSITION 3.10. *For any $0 < \epsilon < 1$ and any $n, d, q \geq 1$, $\mathrm{mIMM}_{n,d,q}$ is downward self-reducible to $\mathrm{mIMM}_{n^\epsilon,d,q}$ by a Dlogtime-uniform pure reduction of depth $O(1/\epsilon)$ and size $O(nd^2 \log q)$ using $O(n)$ oracle gates for $\mathrm{mIMM}_{\ell,d,q}$, $n^{\epsilon/2} \leq \ell \leq n^\epsilon$. Similarly $\mathrm{BIMM}_{n,d}$ is reducible to $\mathrm{BIMM}_{n^\epsilon,d}$ by a reduction of depth $O(1/\epsilon)$ and size $O(nd^2)$ using $O(n)$ oracle gates.*

The following more interesting lemma will be useful in the next section.

LEMMA 3.11. *There is a universal constant c_{CRR} such that for any $0 < \epsilon < 1$ and any $d(n) \leq n$, $\mathrm{IMM}_{n,d(n),n}$ is downward self-reducible to $\mathrm{IMM}_{n^\epsilon,d(n),n^\epsilon}$ by a Dlogtime-uniform TC^0-reduction of depth $O(1/\epsilon)$ and size $O(d(n)^2 \cdot n^{3+2c_{\mathrm{CRR}}})$ using $O(n^3)$ oracle gates.*

Here, c_{CRR} is a specific constant that can be determined from a paper of Hesse et al. [Hesse et al. 2002]. The exact value of c_{CRR} is not important for our purposes, but we estimate that $c_{\mathrm{CRR}} < 10$.

PROOF. Hesse et al. [Hesse et al. 2002] give Dlogtime-uniform TC^0 circuits with $O(n^{c_{\mathrm{CRR}}})$ wires that do the following tasks:

—take as input two n-bit integers a and b, and output $a \bmod b$. (Call this circuit A_n.)

—take as input an n-bit integer a, and output its *Chinese Remainder Representation*, i.e., a sequence of n pairs (a_i, b_i) of $O(\log n)$-bit numbers where $a_i = a \bmod b_i$ and all b_i are distinct primes depending only on n. (Call this circuit B_n.)

—take as input n pairs (a_i, b_i) of $O(\log n)$-bit numbers and output an $O(n \log n)$-bit number a satisfying $a_i = a \bmod b_i$ and $0 \leq a < \prod_i b_i$, if the b_i are distinct primes. (Call this circuit R_n.)

Let n be large enough and set $d = d(n)$. Using these three circuit families we can reduce $\mathrm{IMM}_{n,d,n}$ to the problem of computing $O(n^2)$ instances of mIMM_{n,d,q_i} in parallel for $O(n^2)$ distinct prime $O(\log n)$-bit numbers q_i. Namely to compute the iterated product, we first compute the Chinese Remainder Representation of each input matrix; this gives us $O(n^2)$ instances of mIMM_{n,d,q_i} to solve. Next, we compute the iterated product mod each of the q_i (thereby obtaining the output in Chinese Remainder Representation). Finally, we convert the answer to binary representation. The following three steps describe the computation in more detail.

Step 1. We convert the input matrices $M_1, M_2, \ldots, M_n$ into $2n^2 \geq n(n + \log d)$ sequences of $d \times d$ matrices $M_{1,j}, M_{2,j}, \ldots, M_{n,j}$, $j \in \{1, 2, \ldots, 2n^2\}$ as follows: For each $i \in \{1, \ldots, n\}$ and $k, \ell \in \{1, \ldots, d\}$ we apply the circuit B_{2n^2} on the entry $(M_i)_{k,\ell}$ of M_i padded by leading zeros to $2n^2$ bits, to obtain $((M_{i,1})_{k,\ell}, q_1)$, $((M_{i,2})_{k,\ell}, q_2)$, $\ldots$, $((M_{i,2n^2})_{k,\ell}, q_{2n^2})$. That is, each matrix $M_{i,j}$ consists of the entries of M_i modulo the $O(\log n)$-bit prime q_j. This step consists of $n \cdot d^2$ copies of circuit B_{2n^2} so it can be done by a TC^0 circuit of size $O(d^2 n^{1+2c_{\mathrm{CRR}}})$.

Step 2. For each $j \in \{1, 2, \ldots, 2n^2\}$, we compute the product N_j of matrices $M_{1,j}$, $M_{2,j}$, $\ldots$, $M_{n,j}$ mod q_j. To do so, we use the pure self-reduction of mIMM_{n,d,q_j} to $\mathrm{mIMM}_{n^\epsilon,d,q_j}$ given by Proposition 3.10. However as we do not have oracle access to $\mathrm{mIMM}_{n^\epsilon,d,q_j}$, we replace each oracle gate for $\mathrm{mIMM}_{\ell,d,q_j}$, $n^{\epsilon/2} \leq \ell \leq n^\epsilon$, by a small sub-circuit consisting of an oracle gate $\mathrm{IMM}_{\ell,d,\ell}$ giving an intermediate matrix M followed by an application of the circuit $A_{2\ell^2}$ on each entry of M (padded by leading zeros to $2\ell^2$ bits) to obtain $M \bmod q_j$, i.e., the result expected from the oracle gate $\mathrm{mIMM}_{\ell,d,q_j}$. Thus, to compute N_j, we use a TC^0 circuit with $O(n)$ oracle gates for $\mathrm{IMM}_{\ell,d,\ell}$, $n^{\epsilon/2} \leq \ell \leq n^{\epsilon}$, $O(nd^2)$ copies of $A_{2\ell^2}$, and $O(nd^2 \log n)$ wires for the original self-reduction. In total to implement this step we need $O(n^3)$ oracle gates for $\mathrm{IMM}_{\ell,d,\ell}$ and $n^2 \cdot O(nd^2 \log n + nd^2 n^{2\epsilon c_{\mathrm{CRR}}} + d^2 n^{2\epsilon}) = O(n^{3+2\epsilon c_{\mathrm{CRR}}} d^2)$ wires.

Step 3. From the previous step we obtain matrices $N_1, N_2, \ldots, N_{2n^2}$ which represent the product N of matrices $M_1, \ldots, M_n$. Here, $N_j = N \bmod q_j$. For each $k, \ell \in \{1, \ldots, d\}$ apply the circuit R_{2n^2} on $((N_1)_{k,\ell}, q_1), ((N_2)_{k,\ell}, q_2), \ldots, ((N_{2n^2})_{k,\ell}, q_{2n^2})$ to obtain the entry $N_{k,l}$ of N. This step requires d^2 copies of R_{2n^2}, so in total it uses $O(d^2 n^{2c_{\mathrm{CRR}}})$ wires.

The desired circuit for $\mathrm{IMM}_{n,d,n}$ is obtained by combining the above three steps. Clearly, the circuit will use $O(n^3)$ oracle gates for $\mathrm{IMM}_{\ell,d,\ell}$, $\ell \leq n^{\epsilon}$ and $O(d^2(n^{3+2\epsilon c_{\mathrm{CRR}}} + n^{1+2c_{\mathrm{CRR}}}))$ wires. □

4. AMPLIFYING LOWER BOUNDS

In the previous section we have established several downward self-reducibility results. In this section we show that any problem that is downward self-reducible in this way has circuits of polynomial size of some type if and only if it has very small circuits of that type. Thus, if a small circuit size lower bound can be proved for any such problem, it can be "amplified" into a superpolynomial size lower bound.

The general form of our claims is:

> If a function f is computable by polynomial-size circuits of type $\mathcal{C}$ then for any $\epsilon > 0$, f is computable by circuits of type $\mathcal{C}$ using $O(n^{1+\epsilon})$ gates and wires.

The circuit types we will consider are AC^0, ACC^0, CC^0, TC^0 and NC^1 circuits. The functions f we will consider will typically (but not always) be complete for some complexity class. For example MAJ is complete for TC^0 (under $\leq_T^{\mathrm{NC}^0}$ reductions), and the word problem for S_5 is complete for NC^1, and so on. The consequence of our claim is that establishing a lower bound of $\Omega(n^{1+\epsilon})$ for some $\epsilon > 0$ on the number of wires or gates necessary to compute f would separate some of the circuit classes. The following proposition summarizes known relationships between these circuit classes.

PROPOSITION 4.1.

$$\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$$

$$\mathrm{CC}^0 \subseteq \mathrm{ACC}^0, \;\; \mathrm{CC}^0 \not\subseteq \mathrm{AC}^0$$

Except for the proper inclusion AC^0 $\subsetneq$ ACC^0 [Furst et al. 1984; Yao 1985; Håstad 1988] which also implies CC^0 $\not\subseteq$ AC^0 the precise relationship among ACC^0, CC^0, TC^0 and NC^1 is not known, and any separation or collapse would constitute major progress in theoretical computer science. Separation of, say, TC^0 from NC^1 would typically entail showing that no polynomial-size TC^0 circuit family can compute some chosen function from NC^1. We show that this same separation follows already from a much smaller lower bound.

THEOREM 4.2. *Let $\mathcal{C}$ be one of the circuit classes considered above (and the class of associated functions), and f be a function. If for some ϵ_0 and $s(n) : \mathcal{N} \to \mathcal{N}$, for every $0 < \epsilon \leq \epsilon_0$, f_n is downward self-reducible to f_{n^ϵ} by a pure reduction of depth $O(1/\epsilon)$ and size $O(s(n))$, and f has polynomial-size circuits of type $\mathcal{C}$, then for every $\epsilon' > 0$, f has circuits of type $\mathcal{C}$ of size $O(s(n)n^{\epsilon'})$.*

PROOF. Assume that f_n has circuits of type $\mathcal{C}$ with $n^k + k$ wires. Let $\epsilon = \min(\epsilon'/k, \epsilon_0)$. Consider the reduction of f_n to f_{n^ϵ} that is of size $O(s(n))$ and hence has at most $O(s(n))$ oracle gates. If we replace each oracle gate for f_{n^ϵ} by the circuit of type $\mathcal{C}$ of size $n^{\epsilon k} + k$, we obtain a circuit of type $\mathcal{C}$ for f_n with $O(s(n)n^{\epsilon k}) = O(s(n)n^{\epsilon'})$ wires. The claim follows. (Technically, class $\mathcal{C}$ may not allow for bounded fan-in AND, OR or NOT gates which may appear in the pure reduction hence, one needs to simulate such gates by constant-size circuits of type $\mathcal{C}$. However, this simulation does not affect the size bound by more than a constant factor.) □

By analyzing the depth of the circuits constructed in the proof of Theorem 4.2, one can observe that if $\mathcal{C}$ is a class of *bounded depth* circuits, then f has circuits of type $\mathcal{C}$ having depth $O(1/\epsilon')$ and $O(s(n)n^{\epsilon'})$ wires. For most of our arguments, for any $\epsilon_0 < 1$, either $s(n) = n$ or $s(n) = n^{1+\epsilon_0}$. This yields the following corollary.

COROLLARY 4.3. (*1*) *If for some $\epsilon > 0$,* W5-STCONN *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires, then* $\mathrm{CC}^0 \neq \mathrm{NC}^1$. *The same is true with* ACC^0 *and* TC^0 *in place of* CC^0, *and with* BFE *and* W_{S_5} *in place of* W5-STCONN.

(*2*) *If for some $\epsilon > 0$,* MAJ *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires then* $\mathrm{CC}^0 \neq \mathrm{TC}^0$. *The same is true with* ACC^0 *in place of* CC^0.

(*3*) *If for some $\epsilon > 0$,* AND *requires* CC^0 *circuits with at least $\Omega(n^{1+\epsilon})$ wires then* $\mathrm{CC}^0 \neq \mathrm{ACC}^0$.

Contrast this with the situation for SAT; if SAT is in TC^0, we have no way to bound the number k such that TC^0 size n^k is sufficient to compute SAT. (Although, as we mentioned in Section 1.2, Srinivasan has shown that if P = NP then there are algorithms running in time $n^{1+\epsilon}$ that compute *weak approximations* to MAX-CLIQUE [Srinivasan 2003]. See also our Section 6.)

Although stated as a sequence of implications, the preceding corollary is really a sequence of *equivalences*, since W5-STCONN is complete for NC^1 under $\leq_m^{\mathrm{AC}^0}$ reductions, MAJ is complete for TC^0under $\leq_T^{\mathrm{NC}^0}$ reductions, and AND is complete for ACC^0 under $\leq_T^{\mathrm{CC}^0}$ reductions. Thus, for example, W5-STCONN is in ACC^0 if and only if $\mathrm{NC}^1 = \mathrm{ACC}^0$.

We remark that, since our self-reductions are Dlogtime-uniform, one can compute a constant K such that, for example, if BFE is in Dlogtime-uniform TC^0, then it has TC^0 circuits with $O(n^{1+\epsilon})$ wires where the uniformity machine runs in time $K \log n$. (We have not computed the value of K – and indeed this value may depend on minor details of the particular formulation that is used in defining Dlogtime-uniformity – but we anticipate that $K = 4$ is sufficient; the self-reductions have a *very* regular structure, and the $O(\log n)$ running time of the “original” TC^0 circuit family ends up being simulated only to determine the structure of circuits for inputs of size n^ϵ for small values of ϵ.)

Sometimes concrete lower bounds are easier to prove for specially-constructed sets, rather than for the standard complete sets for a complexity class. The following corollary shows that we can also "amplify" lower bounds for such specially-constructed sets, since if one can show that a specially-constructed set lies in NC^1, then typically one can determine some upper bound on the depth $d(n)$ of the NC^1 circuits computing f.

COROLLARY 4.4. *Let f be computable by NC^1 circuits of depth $d(n)$. If f does not have TC^0 circuits of size $O(3^{d(n)})$ then $\mathrm{TC}^0 \neq \mathrm{NC}^1$. The same is true with ACC^0 and CC^0 in place of TC^0.*

PROOF. If f has NC^1 circuits of depth $d(n)$, then it has a balanced formula of size $2^{d(n)}$, and thus there is a reduction of f to instances of BFE of size $2^{d(n)}$. If $\mathrm{TC}^0 = \mathrm{NC}^1$ then evaluating Boolean formulae of length ℓ can be done by TC^0 circuits of size $O(\ell^{1+\epsilon})$ for any chosen $\epsilon > 0$, by Corollary 4.3. The claim follows. □

The technique is applicable also to other circuit classes, so if we pick a function f from e.g. TC^0 and we know that it is computable by TC^0 circuits of size $O(n^k)$, then if $\mathrm{TC}^0 = \mathrm{ACC}^0$ then for every $\epsilon > 0$, f is computable by ACC^0 circuits using $O(n^{k(1+\epsilon)})$ wires. So proving an $\Omega(n^{k(1+\epsilon)})$ lower bound on the size of ACC^0 circuits for f separates ACC^0 from TC^0.

This technique is applicable, to a certain extent, also to classes larger than NC^1. First, let us consider NL. Boolean iterated matrix product $\mathrm{BIMM}_{n,n}$ is complete for NL. We do not know how to work directly with $\mathrm{BIMM}_{n,n}$, and thus we work with slightly smaller matrices instead.

THEOREM 4.5. *If $\mathrm{NL} \subseteq \mathrm{NC}^1$ then for any $\epsilon > 0$, $\mathrm{BIMM}_{n,2^{\sqrt{\log n}}}$ is computable by NC^1 circuits of size $O(n^{1+\epsilon})$. The same is true with CC^0, ACC^0, and TC^0 in place of NC^1.*

Note, one can replace the dimension bound $2^{\sqrt{\log n}}$ in the theorem by any other function from $n^{o(1)}$. The contrapositive may be more informative; if one can show for some $\epsilon > 0$ that $\mathrm{BIMM}_{n,2^{\sqrt{\log n}}}$ requires NC^1 circuits of size $\Omega(n^{1+\epsilon})$ then one has shown that $\mathrm{NC}^1 \neq$ NL. Unlike the earlier theorems in this section, we obtain only an implication, and not an equivalence – since $\mathrm{BIMM}_{n,2^{\sqrt{\log n}}}$ is not known (or believed) to be complete for NL. Note that this result is for NC^1 *circuit* size; it does not seem to translate into a useful statement about *formula* size.

PROOF. Since $\mathrm{BIMM}_{n,n}$ is in NL, our assumption implies that $\mathrm{BIMM}_{n,n}$ is computable by NC^1 circuits of size $O(n^k)$ for some $k > 0$. Let $\epsilon > 0$ and set $\epsilon' = \epsilon/k$. Then $\mathrm{BIMM}_{n^{\epsilon'},n^{\epsilon'}}$ is computable by NC^1 circuits of size $O(n^{\epsilon' k}) = O(n^\epsilon)$ and hence we conclude that $\mathrm{BIMM}_{\ell,2^{\sqrt{\log n}}}$ is computable by NC^1 circuits of size $O(n^\epsilon)$ for any $\ell \leq n^{\epsilon'}$. (Here, we are taking advantage of the fact that $2^{\sqrt{\log n}}$ grows more slowly than $n^{\epsilon'}$ for any $\epsilon' > 0$.) By Proposition 3.10, $\mathrm{BIMM}_{n,2^{\sqrt{\log n}}}$ is downward self-reducible to $\mathrm{BIMM}_{n^{\epsilon'},2^{\sqrt{\log n}}}$ by a pure reduction of size $O(n2^{2\sqrt{\log n}})$ with $O(n)$ oracle gates for $\mathrm{BIMM}_{\ell,2^{\sqrt{\log n}}}$, $\ell \leq n^{\epsilon'}$. We can replace each oracle gate by an NC^1 circuit of size $O(n^\epsilon)$, yielding an NC^1 circuit of size $O(n2^{2\sqrt{\log n}} + n \cdot n^\epsilon) = O(n^{1+\epsilon})$. □

We now turn to the complexity class #L (the class of functions that count the number of accepting paths of NL machines). This is the largest complexity class that we know how to address using these techniques. Iterated Matrix Multiplication $\mathrm{IMM}_{n,n,n}$ is a problem complete for #L (see [Allender and Ogihara 1996]). $\mathrm{IMM}_{n,2^{\sqrt{\log n}},n}$ is a subproblem not known (or expected) to be complete for #L, but also not known to lie in any smaller complexity class.

THEOREM 4.6. *Let* c_{CRR} *be the constant from Lemma 3.11. If* $\#\mathrm{L} \subseteq \mathrm{TC}^0$ *then it follows that* $\mathrm{IMM}_{n,2^{\sqrt{\log n}},n}$ *is computable by* TC^0 *circuits of size* $O(n^{2c_{\mathrm{CRR}}+4})$*. The same is true with* NC^1 *in place of* TC^0*.*

Thus to separate #L from TC^0 it suffices to show a lower bound of $\omega(n^{2c_{\mathrm{CRR}}+4})$ on the size of TC^0 circuits computing $\mathrm{IMM}_{n,2^{\sqrt{\log n}},n}$. Similarly for NC^1.

PROOF. Since $\mathrm{IMM}_{n,n,n}$ is in #L, by our assumption, $\mathrm{IMM}_{n,n,n}$ is computable by TC^0 circuits of size $O(n^k)$ for some $k > 0$. Choose $\epsilon = 1/k$. Then $\mathrm{IMM}_{n^\epsilon,n^\epsilon,n^\epsilon}$ is computable by TC^0 circuits of size $O(n^{\epsilon k}) = O(n)$ and hence $\mathrm{IMM}_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ is computable by TC^0 circuits of size $O(n)$.

By Lemma 3.11, $\mathrm{IMM}_{n,2^{\sqrt{\log n}},n}$ is downward self-reducible to $\mathrm{IMM}_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ by TC^0 circuits of size $O(2^{2^{\sqrt{\log n}}} \cdot n^{2c_{\mathrm{CRR}}+3}) \leq O(n^{2c_{\mathrm{CRR}}+4})$. There are $O(n^3)$ oracle gates in this reduction, and each gate for $\mathrm{IMM}_{n^\epsilon,2^{\sqrt{\log n}},n^\epsilon}$ can be replaced by circuits with $O(n)$ wires, yielding TC^0 circuits of size $O(n^{2c_{\mathrm{CRR}}+4} + n^4) = O(n^{2c_{\mathrm{CRR}}+4})$. This yields the bound for TC^0 circuits in the statement of the lemma.

To prove the claim regarding NC^1 circuits, it suffices to remark that each MAJ_ℓ gate in the TC^0 reduction above can be replaced by NC^1 circuitry of size $O(\ell)$. (This follows from the fact that MAJ is computable by NC^1 circuits of linear size [Vollmer 1999, Lemma 2.11.1].) The depth of the circuit increases by a factor of at most $O(\log n)$ and the size by at most a constant factor. □

The preceding two theorems do not make use of problems that are known to be *complete* for well-known complexity classes, and thus we obtain only *implications* regarding NL and #L, instead of *equivalent* statements concerning whether these classes collapse with NC^1. However, it is worthwhile noting that $\mathrm{IMM}_{n,3,n}$ is complete for GapNC^1 [Caussinus et al. 1998] (the class of functions over the integers, computable by polynomial-size arithmetic formulae). All functions in NC^1 are in GapNC^1, and it has been conjectured that GapNC^1 coincides with NC^1 [Allender 2004]. GapNC^1 is the only well-studied complexity class not known to be contained in NC^1, for which we can present a complete problem that is strongly downward self-reducible.

THEOREM 4.7. $\mathrm{GapNC}^1 \subseteq \mathrm{TC}^0$ *if and only if* $\mathrm{IMM}_{n,3,n}$ *has* TC^0 *circuits of size* $O(n^{2c_{\mathrm{CRR}}+3})$*. The same is true with* NC^1 *in place of* TC^0*.*

PROOF. Let us prove the first equivalence. Assume that $\mathrm{GapNC}^1 \subseteq \mathrm{TC}^0$. Since $\mathrm{IMM}_{n,3,n}$ is in GapNC^1, there is some $k > 0$ such that $\mathrm{IMM}_{n,3,n}$ has TC^0 circuits of size $O(n^k)$. Let $\epsilon = 2c_{\mathrm{CRR}}/k$. By Lemma 3.11, $\mathrm{IMM}_{n,3,n}$ is downward self-reducible to $\mathrm{IMM}_{n^\epsilon,3,n^\epsilon}$ by TC^0 circuits of size $O(d^2 n^{3+2c_{\mathrm{CRR}}})$ with $O(n^3)$ oracle gates. Replace each oracle gate in the reduction by the TC^0 circuit for $\mathrm{IMM}_{n^\epsilon,3,n^\epsilon}$ of size $O(n^{\epsilon k}) = O(n^{2c_{\mathrm{CRR}}})$ to obtain a TC^0 circuit of size $O(9 \cdot n^{3+2c_{\mathrm{CRR}}} + n^3 n^{2c_{\mathrm{CRR}}})$ computing $\mathrm{IMM}_{n,3,n}$. This shows one implication. The other implication follows from the fact that $\mathrm{IMM}_{n,3,n}$ is complete for GapNC^1 under $\leq_m^{\mathrm{AC}^0}$ reductions.

The equivalence for NC^1 follows from the first one by an argument similar to the proof of the previous theorem. □

5. LIMITS ON DOWNWARD SELF-REDUCIBILITY

In the previous section we have seen that downward self-reducibility provides us with an interesting tool for the study of circuit classes. We have shown that in order to separate circuit classes such as ACC^0 and NC^1, quadratic lower bounds for the circuit complexity of certain NC^1-complete problems would suffice. What about separating ACC^0 from, say NP? That should in principle be a much easier task. Can we use the technique of downward self-reducibility to establish an analog of Corollary 4.3 for ACC^0 versus NP?

The following theorem shows that there are significant obstacles to overcome before such an approach can work. Namely, in order to establish that a problem is strongly downward self-reducible, one must already have an efficient algorithm for the problem.

THEOREM 5.1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function and $\delta > 0$.*

(1) *If f is strongly downward self-reducible, then $f \in \mathrm{NC}$ and f has $\mathrm{CC}^0[q]$ circuits of size 2^{n^δ}, whenever $q \geq 2$ is an integer with at least two distinct prime factors.*

(2) *If f is TC^0 strongly downward self-reducible, then $f \in \mathrm{NC}$ and f has TC^0 circuits of size 2^{n^δ}.*

(3) *If f is strongly downward self-reducible via polynomial time Turing reductions then f is in* P.

PROOF. We prove the second claim first.

2) Let $n \geq 2$. In order to build a circuit for f_n, start with the TC^0 circuit of depth d and size n^k that reduces f_n to f_{n^ϵ}, for some $\epsilon < 1$. If we replace each oracle gate in this circuit with the circuit that reduces f_{n^ϵ} to $f_{(n^\epsilon)^\epsilon}$, the depth of the new circuit is d^2 and the size is at most $n^k + n^k \cdot n^{\epsilon k}$. We repeat the process until the oracle gates are of size $O(1)$, at which point we replace the oracle gates by circuitry of size $O(1)$ computing f on small inputs. The number of stages is $O(\log\log n)$; thus the depth is $d^{O(\log\log n)} = \log^{O(1)} n$. The size of the circuit is polynomially bounded by $n^k \cdot n^{\epsilon k} \cdot n^{\epsilon^2 k} \cdots \leq n^{k/(1-\epsilon)}$. Finally, replace each MAJ gate by an NC^1 circuit. It is easy to verify that the resulting circuit is logspace-uniform if the self-reduction circuits are. This establishes that $f \in \mathrm{NC}$. In order to see that f has TC^0 circuits of size 2^{n^δ}, merely follow the same iteration process as above, but continue for only $O(1)$ stages instead of $O(\log\log n)$ stages. This results in a TC^0 oracle circuit with oracle gates for f_m with $m < n^\delta$. Now replace each oracle gate with a DNF expression for f_m. (Clearly, if the self-reduction is an AC^0 circuit instead of a TC^0 circuit, then f has AC^0 circuits of size 2^{n^δ}.)

1) The hypothesis for this implication is stronger than the hypothesis in part 2; we are assuming a $\leq_T^{\mathrm{NC}^0}$ downward self reduction, rather than a $\leq_T^{\mathrm{TC}^0}$ reduction. Hence again we can conclude that $f \in \mathrm{NC}$. We obtain the upper bound of 2^{n^δ} on $\mathrm{CC}^0[q]$ circuit size in a similar way. We use the same iterative process for $O(1)$ steps and obtain an oracle circuit with oracle gates for f_m with $m < n^\delta$, where all of the other gates have fan-in $O(1)$. The DNF expression for f_m can be computed by $\mathrm{CC}^0[q]$ circuits of size $O(2^m)$, by using $\mathrm{CC}^0[q]$ circuits of size 2^m to compute each AND of fan-in m [Barrington et al. 1990]. At most one of these AND gates will evaluate to 1, and hence taking the MOD-q of these AND gates computes the DNF for f_m.

3) Again we use the obvious recursive algorithm. We run the Turing reduction and whenever it asks an oracle query about a smaller instance of f we recursively invoke the reduction on the smaller instance. If the reduction runs in time $O(n^k)$ then the total running

time of the algorithm will be bounded by $n^k \cdot n^{\epsilon k} \cdot n^{\epsilon^2 k} \cdots \leq n^{k/(1-\epsilon)}$. Since ϵ is constant, the time is polynomial. □

As a corollary, using the fact that every problem in NC^1 is reducible via $\leq_m^{\mathrm{NC}^0}$ reductions to BFE, we obtain the following upper bound on the complexity of problems in NC^1, which appears to be a new observation.

COROLLARY 5.2. *For every $\delta > 0$ and every integer $q \geq 2$ with at least two distinct prime factors, every problem in NC^1 has $\mathrm{CC}^0[q]$ circuits of size 2^{n^δ} .*

Speculation: Although Theorem 5.1 suggests that we abandon any attempt to show that SAT has the downward self-reducibility property, it does not exclude the following approach for trying to prove an analog of Corollary 4.3 for NP. (Such an analog might, for instance, state that if NP $=$ TC^0 then SAT has TC^0 circuits of size n^2.) Rather than trying to present a self-reduction for SAT *unconditionally*, perhaps one can start with the *assumption* that NP $\subseteq$ TC^0 and construct a downward self-reduction of SAT (or some other specially-constructed set in NP) and conclude that under this assumption SAT has almost linear size TC^0 circuits.

This is the appropriate time to observe that if NP $\subseteq$ TC^0, then SAT certainly does have the strong downward self-reducibility property; this follows from Proposition 5.3 below. However, since one can say nothing about the size of this self-reduction (other than that it is computed by an NC^0 oracle circuit of polynomial size), this does not seem to allow us to conclude that SAT has TC^0 circuits of, say, quadratic size.

PROPOSITION 5.3. *Let $\mathcal{C}$ be one of our constant-depth circuit classes. Let f and g be functions that are equivalent under $\leq_{\mathrm{T}}^{\mathcal{C}}$ reductions (i.e., $f \leq_{\mathrm{T}}^{\mathcal{C}} g \leq_{\mathrm{T}}^{\mathcal{C}} f$), and let f be $\mathcal{C}$ strongly downward self-reducible. Then g is also $\mathcal{C}$ strongly downward self-reducible. Moreover, the size of the self-reduction of f can be determined from the sizes of the reductions between f and g and the size of the strong downward self-reduction of f.*

PROOF. The polynomial-size reductions between f and g each ask queries of size at most n^k for some k, for all $n \geq 2$. The strong downward self-reduction of f reduces f_n to f_{n^ϵ} for some $\epsilon > 0$. Let ℓ be such that $\epsilon^\ell < 1/(2k^2)$. Let $\{C_n\}$ be the circuit family that is the ℓ-fold composition of the downward self reduction of f. By Proposition 2.3, $\{C_n\}$ is a $\mathcal{C}$ downward self-reduction that, on inputs of length n^k makes no query of length greater than $(n^k)^{\epsilon^\ell} < n^{1/(2k)}$. If we compose the reduction from g to f with the reduction computed by $\{C_n\}$, we obtain a reduction of g_n to $f_{n^{1/(2k)}}$. Composing this reduction with the reduction from f to g, we obtain a reduction of g_n to $g_{n^{1/2}}$. This establishes that g is $\mathcal{C}$ strongly downward self-reducible. □

6. INAPPROXIMABILITY OF MAX-CLIQUE

In this section we adapt the technique of Srinivasan [Srinivasan 2003] to the setting of constant-depth circuit classes, and also obtain a lower bound on the complexity of any polynomial-time reduction of MAX-CLIQUE to the problem of computing approximations to MAX-CLIQUE.

For functions $f : \{0,1\}^* \to \mathbb{N}$ and $\alpha : \mathbb{N} \to \mathbb{N}$, *a function* $g : \{0,1\}^* \to \mathbb{N}$ α-*approximates* f if $g(x) \leq f(x)$ and $f(x) \leq \alpha(|x|)g(x)$ for all $x \in \{0,1\}^*$. MAX-CLIQUE is the following computational problem: given an undirected graph G determine the size of the largest clique in G. For simplicity we assume that G is given by its adjacency matrix. We say that the *size* of G is the number of vertices in G. It is known [Zuckerman 2007] (see also [Håstad 1999; Feige and Kilian 1998; Khot 2001; Khot and Ponnuswami 2006]) that if for some $\epsilon > 0$ there is a $n^{1-\epsilon}$-approximation to MAX-CLIQUE computable in P then P $=$ NP.

We use the technique of Srinivasan [Srinivasan 2003] to show the following statement:

THEOREM 6.1. *Let $k > 1$ and let $\epsilon = \epsilon(n) < 1$ be such that $\epsilon(n) = \omega(\log\log n/\log n)$. If* MAX-CLIQUE *is computable by* $\mathrm{AC}^0[q]$ *circuits of size $O(n^k)$ then it follows that a $n^{1-\epsilon(n)}$-approximation to* MAX-CLIQUE *is computable by* $\mathrm{AC}^0[q]$ *circuits having size $O(n^{1+(k-1)\epsilon(n)})$. The same is true with* TC^0 *and* NC^1 *in place of* $\mathrm{AC}^0[q]$.

It is interesting to note that the depth of the $O(n^{1+(k-1)\epsilon(n)})$-size circuits does not increase while decreasing $\epsilon(n)$. As stated, the theorem holds only for nonuniform circuits, but a uniform version holds for any function $\epsilon(n)$ that is sufficiently easy to compute. To prove the theorem we need the following simple lemma.

LEMMA 6.2. *There is a constant $c > 0$ such that for any $\ell, m \geq 1$ there is a Dlogtime-uniform constant-depth circuit of size at most $c2^\ell \ell m$ consisting of unbounded fan-in* AND *and* OR *gates, and unary* NOT *gates, that takes as its input ℓ-bit binary representations of integers $y_1, \ldots, y_m$, and produces as its output the binary representation of $z = \max\{y_1, y_2, \ldots, y_m\}$.*

PROOF. The computation of the circuit proceeds in three steps. We identify integers $0, \ldots, 2^\ell - 1$ with their ℓ-bit binary representations.

Step 1: compute bits $b_0, b_1, \ldots, b_{2^\ell-1}$ where $b_i = 1$ iff $i \in \{y_1, \ldots, y_m\}$. For $i = 0, \ldots, 2^\ell - 1$ and $w \in \{0,1\}^\ell$, let $g_i(w)$ be a circuit that evaluates to one if w is the binary representation of i. Clearly, $g_i(w)$ can be constructed from a single AND_ℓ gate and at most ℓ NOT gates. Bit b_i is obtained by taking OR of $g_i(y_j)$ for $j = 1, \ldots, m$. Thus to compute $b_0, b_1, \ldots, b_{2^\ell-1}$ we need 2^ℓ OR_m gates, $2^\ell m$ AND_ℓ gates and ℓm NOT gates (as the NOT gates may be reused by different gates $g_i(y_j)$ and $g_{i'}(y_j)$). Hence, we need at most $3 \cdot 2^\ell \ell m$ wires for this step.

Step 2: compute the "suffix-OR**" of $b_0, b_1, \ldots, b_{2^\ell-1}$.** The suffix-OR is a vector of bits $d_0, d_1, \ldots, d_{2^\ell-1}$ with the property that $d_i = 1$ if and only if for some $i' \in \{i, i+1, \ldots, 2^\ell - 1\}$, $b_{i'} = 1$. It can be computed by Dlogtime-uniform constant-depth circuits of size $O(2^\ell)$ consisting of AND, OR and NOT gates of fan-in at most 2^ℓ, as was shown by Chandra, Fortune and Lipton [Chandra et al. 1985].

Step 3: compute the output z. For $i = 0, \ldots, 2^\ell - 2$ let $e_i = (d_i \text{ AND } (\text{NOT } d_{i+1}))$ and $e_{2^\ell-1} = d_{2^\ell-1}$. Hence, the k-th bit of the ℓ-bit binary representation of z is obtained by taking the OR of gates computing e_i for all those i such that the k-th bit of the ℓ-bit binary representation of i is one. This step requires ℓ OR_{2^ℓ} gates, and $2^\ell - 1$ AND_2 and NOT gates.

Clearly, the combination of the above three steps gives a constant-depth $c2^\ell \ell m$ size circuit that correctly computes z. Dlogtime-uniformity of the circuit is routine to establish. □

We continue with the proof of Theorem 6.1.

PROOF. Let us assume that we have an $\mathrm{AC}^0[q]$ circuit family of size $O(n^k)$ that computes MAX-CLIQUE. We will build an $\mathrm{AC}^0[q]$ circuit family of size $O(n^{1+(k-1)\epsilon})$ computing a $n^{1-\epsilon}$-approximation of MAX-CLIQUE. The computation of the approximation proceeds as follows: we partition the vertices of the graph G into $n^{1-\epsilon}$ parts $V_1, \dots, V_{n^{1-\epsilon}}$ of size at most $\lceil n^\epsilon \rceil$. For $i = 1, \dots, n^{1-\epsilon}$ we compute in parallel MAX-CLIQUE of G restricted to V_i. Then we output the largest of these partial results. The correctness of the algorithm follows from the simple observation that if G contains a clique of size $f(G)$ then for some i, V_i contains at least $f(G)/\lceil n^{1-\epsilon} \rceil$ vertices of that clique and hence MAX-CLIQUE of G restricted to V_i is at least $f(G)/\lceil n^{1-\epsilon} \rceil$.

The size of a circuit carrying out the computation can be bounded as follows. We use $n^{1-\epsilon}$ circuits of size $O(n^{\epsilon k})$ to compute the value of the $n^{1-\epsilon}$ MAX-CLIQUE subproblems. This requires size $O(n^{1+(k-1)\epsilon})$ in total. By Lemma 6.2 we can find the maximum of the $n^{1-\epsilon}$ values in the range $\{0, \dots, n^\epsilon\}$ by an AC^0 circuit of size $O(n^{1-\epsilon} \cdot n^\epsilon \cdot \log n^\epsilon) = O(n \log n)$. Thus the size of the circuits is bounded by $O(n^{1+(k-1)\epsilon})$. Dlogtime-uniformity of the circuit is routine to establish. The case of TC^0 and NC^1 is proven by essentially the same argument. □

The technique from the previous proof can be also used to establish the following claim.

THEOREM 6.3. *Let $0 < \epsilon < 1$ and $k < 1/\epsilon$ be constants. If there is a polynomial time algorithm that solves* MAX-CLIQUE_n *using an oracle for $m^{1-\epsilon}$-approximation of* MAX-CLIQUE_m*, where $m \le n^k$, then* MAX-CLIQUE_n *is downward self-reducible to* $\text{MAX-CLIQUE}_{n^{\epsilon k}}$.

PROOF. In the proof of Theorem 6.1 we have seen how to compute a $m^{1-\epsilon}$-approximation of MAX-CLIQUE_m by asking queries to $\text{MAX-CLIQUE}_{m^\epsilon}$. If there is a polynomial time algorithm that solves MAX-CLIQUE_n using an oracle for $m^{1-\epsilon}$-approximation of MAX-CLIQUE_m where $m \le n^k$, then we can combine it with the above reduction to obtain the desired self-reduction. □

This gives rise to what is perhaps the first example of a lower bound showing that there is no "quick" reduction between two natural NP-optimization problems. For many natural NP-complete problems A and B, very efficient reductions between A and B are known. (For example, for any problem $A \in \mathrm{NTIME}(n \log^{O(1)} n)$, there is a many-one reduction from A to SAT that is computable in time $O(n \log^{O(1)} n)$ [Cook 1988].) It is easy to show that if $B \notin \mathrm{NTIME}(n^k)$, then any reduction from B to SAT requires time $n^k / \log^{O(1)} n$ – but this does not provide any useful lower bound on the complexity of reducing natural problems to SAT, since no natural NP-complete problem is known to lie outside of $\mathrm{NTIME}(n)$. There seems to be no pair of natural NP-complete problems A and B known, where a reduction from A to B is known to require more than linear time (even under the assumption that $\mathrm{P} \neq \mathrm{NP}$).

In contrast to this, consider the problem of computing a $\sqrt{n}$-approximation to MAX-CLIQUE. Zuckerman presents a deterministic polynomial-time Turing reduction from MAX-CLIQUE to this approximation problem [Zuckerman 2007]. (More precisely, Zuckerman shows that distinguishing graphs having only *small* cliques from graphs with *large* cliques is complete for NP under many-one reductions, i.e., that one can decide the membership of a formula in SAT from the answer to an instance of an arbitrary $n^{1-\epsilon}$-approximation of MAX-CLIQUE_n. The polynomial-time Turing reduction from MAX-CLIQUE follows from the trivial observation that MAX-CLIQUE is computable in $\mathrm{P}^{\mathrm{SAT}} = \mathrm{P}^{\mathrm{NP}}$.) How long must the queries in this reduction be? Assuming that $\mathrm{P} \neq \mathrm{NP}$, Theorems 6.3 and 5.1 tell us that the queries in this reduction must ask about graphs with at least n^2 vertices. We can state the following claim

COROLLARY 6.4. $\mathrm{P} = \mathrm{NP}$ *if and only if there is an* $\alpha < 2$ *and a deterministic polynomial-time Turing reduction from* MAX-CLIQUE *to the problem of computing a* $\sqrt{n}$*-approximation to* MAX-CLIQUE *that asks queries of size no greater than* n^{α}.

PROOF. One direction follows from the observation that if $\mathrm{P} = \mathrm{NP}$ then there is a polynomial-time Turing reduction for this problem that asks queries of size $O(1)$ (or asks no queries at all).

For the other direction: If there is a reduction from MAX-CLIQUE to the problem of computing a $\sqrt{n}$-approximation to MAX-CLIQUE that asks queries only to graphs of size n^{α} for some $\alpha < 2$, then by Theorem 6.3, MAX-CLIQUE$_n$ is downward self-reducible to MAX-CLIQUE$_{n^{\alpha/2}}$. By Theorem 5.1, this implies that MAX-CLIQUE is computable in polynomial time, and hence $\mathrm{NP} = \mathrm{P}$. □

Clearly, analogous statements can be proved for n^{ϵ}-approximation for any value of ϵ such that $0 < \epsilon < 1$ and $\alpha < 1/(1-\epsilon)$; the case $\epsilon = 1/2$ is likely to be of greatest interest. Similar claims can be also proved for probabilistic reductions instead of deterministic ones, under the assumption that SAT does not have probabilistic polynomial-time algorithms.

It is worthwhile mentioning that, in some sense, decreasing the size of the query length in Zuckerman's reduction [Zuckerman 2007] from MAX-CLIQUE to computing a $n^{1/2}$-approximation to MAX-CLIQUE is a *universal* approach to proving $\mathrm{P} = \mathrm{NP}$. If any approach will work, then this approach will.

7. CIRCUIT LOWER BOUNDS

We observed in Section 1.2 that, although BFE requires size $n^{1+\epsilon_d}$ on depth d TC^0 circuits [Impagliazzo et al. 1997], no similar bound for ACC^0 or even $\mathrm{CC}^0[q]$ circuits is known. Here, we present lower bounds of this sort for SAT.

We begin this section by showing that problems with small constant-depth circuits have algorithms that run quickly and have small space bounds. Let TISP($t(n), s(n)$) denote the class of problems that are computable by machines running in time $O(t(n)$ that use space at most $O(s(n))$. (This definition is somewhat sensitive to the underlying model of computation. We shall always refer explicitly to either the Turing machine model or the random access machine model, to clarify which class is meant.)

A technical matter that must be dealt with in stating the following theorem, is that Dlogtime-uniformity does not seem to guarantee that there is a quick way to enumerate, for a given gate h, the list of gates g for which there is a wire from g to h. There are some standard techniques for ensuring that this property holds (see, e.g., [Allender and Gore 1994]), but we note that these techniques seem to involve a polynomial blow-up in the circuit size, which we would prefer to avoid. We believe that, for most uniform families of circuits that are constructed, a quick enumeration of the inputs to a given gate will nonetheless be possible. Rather than alter the definition of Dlogtime-uniformity, in this section we simply say that a circuit family is *strongly uniform* if it is Dlogtime-uniform, and in addition, on input (n, i, h), the name of the gate g that is the i-th input to the gate in C_n having label h can be computed in time $\log^{O(1)} n$.

THEOREM 7.1. *If A has strongly-uniform* TC^0*circuits of depth d with $O(n^{1+\epsilon})$ wires then for every $0 < \delta < 1+\epsilon$, $A \in TISP((n^{1+\epsilon} + n^{\delta d}) \log^{O(1)} n, n^{1+\epsilon-\delta} \log^{O(1)} n)$ on random access machines and $A \in TISP((n^{1+\epsilon+\delta d} \log^{O(1)} n, n^{1+\epsilon-\delta} \log^{O(1)} n)$ on Turing machines. The same is true with* $\mathrm{AC}^0[q]$ *or* $\mathrm{CC}^0[q]$ *in place of* TC^0.

PROOF. A naïve recursive way to evaluate the circuit in space $O(\log n)$ would require time $O(n^{d(1+\epsilon)} \log n)$. Since we can use more space we will use it to remember the computed values of gates that have fan-in larger than n^δ. The faster algorithm then will also recursively evaluate the circuit but whenever it computes the value of a gate with fan-in larger than n^δ it records the value so such a gate will be evaluated at most once. On a random access machine we will store the values in a binary search tree, on a Turing machine we will store them in a simple list. Since there are at most $O(n^{1+\epsilon}/n^\delta)$ gates with fan-in larger than n^δ we will need space only $O(n^{1+\epsilon-\delta} \log^{O(1)} n)$. Finding the value of a gate and whether it has already been computed will take $O(\log^{O(1)} n)$ time on a random access machine and $O(n^{1+\epsilon-\delta} \log^{O(1)} n)$ on a Turing machine. To bound the total time needed to evaluate the circuit notice that we will have to recursively evaluate a tree of fan-in at most n^δ and depth d. To traverse the tree we will need to make $n^{\delta d}$ visits to the nodes. Beside that we will have to evaluate the gates with large fan-in. Since there are at most $O(n^{1+\epsilon})$ wires leading into them these gates will additionally cost at most $O(n^{1+\epsilon})$ node visits. This yields the claimed time bound. □

We need to make use of known time-space tradeoffs for SAT. The following theorem is a special case of Theorem 1.3 in the excellent survey article by van Melkebeek [van Melkebeek 2007]:

THEOREM 7.2. *For every real c such that $1 < c < 5/3$, there exists a positive real e such that SAT cannot be solved by both*

(*1*) *a co-nondeterministic machine with random access that runs in time $O(n^c)$ and*

(*2*) *a deterministic random-access machine that runs in time $O(n^{1.5})$ and space $O(n^e)$.*

Moreover, the constant e approaches 1 from below when c approaches 1 from above.

THEOREM 7.3. *For every $d \geq 1$ there is a constant $\epsilon_d > 0$ such that SAT does not have strongly-uniform depth d* TC^0 *circuits of size $O(n^{1+\epsilon_d})$. The same is true with* $\mathrm{AC}^0[q]$ *or* $\mathrm{CC}^0[q]$ *in place of* TC^0.

PROOF. Assume that the claim fails for some depth $d \geq 2$; thus for every $\epsilon > 0$, SAT has strongly-uniform depth d TC^0 circuits with $O(n^{1+\epsilon})$ wires.

Pick any $\epsilon < 1/(3d-1)$ and $\delta = 3\epsilon$. By Theorem 7.1, SAT is solvable on random access machines in TISP($(n^{1+\epsilon} + n^{\delta d}) \log^{O(1)} n, n^{1+\epsilon-\delta} \log^{O(1)} n) \subseteq$ TISP($n^{1+2\epsilon} + n^{d\delta+\epsilon}, n^{1+2\epsilon-\delta}) \subseteq$ TISP($n^{1+2\epsilon}, n^{1-\epsilon}$). Since this is true for all $\epsilon < 1/(3d-1)$, we have in particular that SAT is in DTIME(n^c) on random access machines for all $c > 1$.

For the rest of the proof fix some $\epsilon < 1/(3d-1)$. In particular, we have SAT is in TISP($n^{1.5}, n^{1-\epsilon}$) on random access machines.

By Theorem 7.2, if we let c approach 1 from above, the value of e (in Theorem 7.2) approaches 1 from below. Thus there is some value of $c > 1$ for which $e > 1-\epsilon$ (in the statement of Theorem 7.2). Fix these values of c and e. Thus, we now have that SAT is in TISP($n^{1.5}, n^e$) on random access machines.

At this point, by Theorem 7.2, we know that SAT is not both solvable by co-nondeterministic random access machine in time $O(n^c)$, and in TISP($n^{1.5}, n^e$) on random access machines. But we have already observed (three paragraphs ago) that SAT is in DTIME(n^c) and thus it is solvable in co-nondeterministic time $O(n^c)$. Thus we must conclude that SAT is not in TISP($n^{1.5}, n^e$). But this contradicts the conclusion of the preceding paragraph. Case $d = 1$ follows from the case of $d = 2$. □

8. THE NATURAL PROOFS BARRIER

Razborov and Rudich [Razborov and Rudich 1997] identified a significant obstacle to further progress in proving lower bounds on circuit size, by observing that existing lower bound arguments rely on the existence of an easy-to-recognize combinatorial property of a function f that (a) is shared by a large fraction of all functions, and (b) is shared by no function that has small circuits of a given type. Razborov and Rudich showed that any "Natural Proof" that follows this paradigm and shows that a function cannot be computed by circuits of a class $\mathcal{C}$ constitutes a proof that $\mathcal{C}$ cannot compute pseudorandom function generators. It is not clear how significant an obstacle this poses for proving lower bounds against ACC^0, since there is not much evidence that ACC^0 circuit families can compute pseudorandom function generators. However, for TC^0 this is a serious impediment, since Naor and Reingold have presented a good candidate pseudorandom function generator that is computable in TC^0 [Naor and Reingold 2004]. (The reader should keep in mind the distinction between pseudorandom *function* generators and pseudorandom *bit* generators. It is known that there are no pseudorandom *function* generators computable in AC^0 [Linial et al. 1993]; in contrast, if the Naor-Reingold generator is secure, then there are pseudorandom *bit* generators computable in NC^0 [Applebaum et al. 2006].)

It is premature to argue very strongly that we have identified a path around this obstacle. After all, the only new lower bound that this paper offers is to be found in Section 7, and that bound follows from known time-space tradeoff results. (These time-space tradeoffs, in turn, rely on diagonalization, which lies outside the natural proofs framework, but only gives lower bounds for *uniform* circuit families. The natural proofs framework addresses the problem of finding lower bounds for *nonuniform* circuit complexity.)

However, we contend that it is at least plausible that a natural proof could form the basis for a proof that NC^1 $\neq$ TC^0, even assuming that the Naor-Reingold generator is cryptographically secure.

How?

A proof that NC^1 $\neq$ TC^0 could conceivably consist of two parts:

(1) A proof that BFE requires TC^0 circuits of size $n^{1.5}$, and

(2) Appeal to Corollary 4.3, to conclude that NC^1 $\neq$ TC^0.

Let us assume for the moment that someone hands us a natural proof of the $n^{1.5}$ lower bound that takes care of the first part of this hypothetical argument. The entire two-part argument nonetheless fails to be a "natural" proof, because the proof of Corollary 4.3 centers on strong downward self-reducibility, which is a combinatorial property that is shared by only a *vanishingly small fraction* of all Boolean functions on n variables, contrary to the requirements of a natural proof. (Strictly speaking, the strong downward self-reducibility property is not a "combinatorial property" in the sense of the Natural Proofs framework, as it is a relationship between function values on different input sizes. However, all strongly downward self-reducible functions must have truth-tables of small Kolmogorov complexity (since the truth-table of size 2^n is determined completely by a truth-table of size 2^{n^δ}), and thus they constitute a tiny fraction of all functions.)

So now we are left with the question of whether it is reasonable to hope that a natural proof could possibly show that BFE requires TC^0 circuits of size $n^{1.5}$.

First, we note that there are already examples of natural proofs that yield lower bounds of the form n^k for some fixed k. The parity lower bound of Impagliazzo, Paturi, and Saks gives a lower bound of this form for BFE on TC^0 circuits of depth d [Impagliazzo et al. 1997]. Håstad gives a nearly cubic lower bound on formula size [Håstad 1998]. These are natural proofs.

Next, in order to directly address the question of what obstacles have been identified by Razborov and Rudich that might block a proof showing that BFE requires TC^0 circuits of size $n^{1.5}$, let us examine their framework more closely, by recalling their definitions of "natural" and "useful" combinatorial properties.

Let F_n denote the class of all Boolean functions $f_n : \{0,1\}^n \to \{0,1\}$. A property $\{T_n \subseteq F_n\}_{n\in\mathbb{N}}$ is $QuasiP$*-natural* if there is a sub-property $\{T_n^* \subseteq T_n\}_{n\in\mathbb{N}}$ such that for some $\epsilon, c > 0$

(1) $|T_n^*| \geq |F_n|/2^{\epsilon n}$, and

(2) there is a deterministic algorithm that given a truth-table of a function $f_n : \{0,1\}^n \to \{0,1\}$ decides whether $f_n \in T_n^*$ in time 2^{n^c}.

Furthermore, a property $\{T_n \subseteq F_n\}_{n\in\mathbb{N}}$ is *useful* against a circuit class Λ if no sequence of functions $\{f_n \in T_n\}_{n\in\mathbb{N}}$ is computable by circuits from Λ.

Razborov and Rudich show that any $QuasiP$-natural property that is useful against TC^0 can be used as a subroutine to foil any purported pseudorandom function generator that is computable in TC^0. More generally, they show how to transform any natural lower bound proof into a lower bound on the complexity of computing a pseudorandom function generator. However it is absolutely essential for their argument, that there be a *single* natural property T that is useful against TC^0 circuits of size n^k for *every* k; lower bounds for circuits of size n^k for small fixed k translate into lower bounds for pseudorandom function generators that are so weak as to be uninformative. More to the point, such natural properties can easily be shown to exist.

To be concrete, let us exhibit an example of a property $T = \{T_n\}_{n\in\mathbb{N}}$ that is $QuasiP$-natural and useful against TC^0 circuits of size $O(n^{1.5})$. Our property T is defined as follows:

$$T_n = \{f_n : \{0,1\}^n \to \{0,1\};\ f_n \text{ does not have circuits of depth } \log^* n \text{ and size } n^2 \text{ consisting of MAJ and NOT gates}\}.$$

It is a trivial exercise to verify that T is natural and useful against TC^0 circuits of size $O(n^{1.5})$. Of course, we are not able to establish that BFE has property T; if it does, then by Corollary 4.3 NC$^1 \neq$ TC^0. Clearly, this argument makes use of no special properties of TC^0; one can easily come up with a $QuasiP$-natural property that will be useful against any class of circuits of a fixed polynomial size.

However, the existence of property T does not seem to imply anything very interesting about the nonexistence of pseudorandom function generators (and consequently does not yield interesting upper bounds on the complexity of factoring Blum integers, which would follow if the Naor-Reingold generator is insecure [Naor and Reingold 2004]). Thus it seems to us that it is reasonable to hope for a “natural” proof that BFE satisfies property T, which would then yield an “unnatural” proof of $\mathrm{TC}^0 \neq \mathrm{NC}^1$, by Corollary 4.3.

9. CONCLUSIONS AND OPEN PROBLEMS

So are there reasons to be more optimistic about prospects for lower bounds? We are not sure. The truth is that we do not understand computation. All the known lower bounds essentially rest on information theoretic arguments and none of them really takes *computation* into account. We realize that this is a vague statement; part of the challenge in seeking lower bound proofs is to be able to say something more precise. For example we are unable to handle *recursion*, so our bounds typically deteriorate with depth. Hence, the underlying message of Razborov and Rudich – namely, that we need to go beyond combinatorial arguments – is still a worthwhile message. We identify two still unresolved challenges that we believe would advance our understanding of computation:

- —Prove $\Omega(n^2)$ lower bounds on the length of width 5 branching programs computing an explicit function (by which we mean any problem in NP). It appears that nothing better than $\Omega(n^2/\log n)$ is known [Neciporuk 1966; Razborov 1991].
- —Prove $\Omega(n^{1+1/\sqrt{d}})$ lower bounds on the size of depth d $\mathrm{CC}^0[6]$ or TC^0 circuits computing an explicit function.

Are there perhaps fundamental barriers that remain in our path, as we attempt to prove circuit lower bounds?

One way to explore this question is to follow the lead of Razborov [Razborov 1995b], who showed that (under cryptographic assumptions) the bounded arithmetic proof system S^2_2 cannot prove that SAT requires circuits of superpolynomial size. (In earlier work, Razborov had argued that most existing lower bound arguments can be carried out in even weaker systems [Razborov 1995a].)

Perhaps techniques similar to those of Razborov [Razborov 1995b], combined with our observations, can enable one to prove that S^2_2 (or a similar system) cannot prove that BFE requires TC^0 circuits of size $n^{1+\epsilon}$.

The most important and interesting question raised by this work is the question of whether it can ultimately lead to separations of complexity classes. (This topic is also discussed in a recent survey [Allender 2008].) However, a number of other questions naturally arise. We close by listing two such questions.

- —Are there sets complete for every level of the NC hierarchy that are downward self-reducible to instances of size n^ϵ? Or is there some fundamental reason why we were unable to find a downward self-reduction of this sort for any problem that is complete for NL or L? (In Theorem 4.5 we worked with a restricted version of the NL-complete problem BIMM; the restriction is not believed to be complete for NL.) Showing that a complete set for L is strongly downward self-reducible (via a pure reduction) would show that every problem in L has subexponential-size $\mathrm{CC}^0[q]$ circuits (by Theorem 5.1), which would be a new upper bound. However, we know of no similar obstacle to showing that sets complete for L or NL are AC^0 strongly downward self-reducible, or that functions complete for #L are TC^0 strongly downward self-reducible. Note in this regard that problems in NL have subexponential-size AC^0 circuits [Allender et al. 2008] and functions in #L have subexponential-size TC^0 circuits [Allender et al. 2001].

—If NP $=$ TC^0, does SAT have TC^0 circuits of quadratic size? If NEXP $\subseteq$ nonuniform CC^0[6], does the standard complete set for NEXP have CC^0[6] circuits of quadratic size? (Even if arguments based on downward self-reducibility fail for problems outside of NC, perhaps there is another approach that leads to the same conclusion.)

Acknowledgments

We thank David Mix Barrington, Scott Diehl, Lance Fortnow, Luke Friedman, Stephan Holzer, Alexander Razborov, Mike Saks, Mario Szegedy, Denis Thérien, and Fengming Wang for helpful discussions. We thank Ryan Williams for calling our attention to the work of Srinivasan [Srinivasan 2003]. We thank all the referees and the program committee for helpful suggestions on this and preliminary versions of this paper.

REFERENCES

AGRAWAL, M. The isomorphism conjecture for constant depth reductions. *Journal of Computer and System Sciences*. To appear. A preliminary version appeared as [Agrawal 2001].

AGRAWAL, M. 2001. The first-order isomorphism theorem. In *Proc. Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS).* Lecture Notes in Computer Science, vol. 2245. 70–82.

AGRAWAL, M., ALLENDER, E., AND RUDICH, S. 1998. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *Journal of Computer and System Sciences 57*, 127–143.

AJTAI, M. 1999. A non-linear time lower bound for Boolean branching programs. In *Proc. IEEE Symp. on Found. of Comp. Sci. (FOCS)*. 60–70.

ALLENDER, E. 2004. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs*, J. Krajíček, Ed. Quaderni di Matematica, vol. 13. Seconda Università di Napoli, 33–72.

ALLENDER, E. 2008. Cracks in the defenses: Scouting out approaches on circuit lower bounds. In *Computer Science – Theory and Applications (CSR 2008).* Lecture Notes in Computer Science, vol. 5010. 3–10.

ALLENDER, E. AND GORE, V. 1994. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing 23*, 1026–49.

ALLENDER, E., HELLERSTEIN, L., MCCABE, P., PITASSI, T., AND SAKS, M. E. 2008. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM Journal on Computing 38,* 1, 63–84.

ALLENDER, E., KOUCKÝ, M., RONNEBURGER, D., ROY, S., AND VINAY, V. 2001. Time-space tradeoffs in the counting hierarchy. In *Proc. 16th Ann. IEEE Conf. on Computational Complexity (CCC).* 295–302.

ALLENDER, E. AND OGIHARA, M. 1996. Relationships among PL, #L, and the determinant. *RAIRO - Theoretical Informatics and Applications 30,* 1, 1–21.

APPLEBAUM, B., ISHAI, Y., AND KUSHILEVITZ, E. 2006. Cryptography in NC^0. *SIAM Journal on Computing 36,* 4, 845–888.

BARRINGTON, D. A. 1989. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences 38,* 1 (Feb.), 150–164.

BARRINGTON, D. A. M., STRAUBING, H., AND THÉRIEN, D. 1990. Non-uniform automata over groups. *Inf. Comput. 89,* 2, 109–132.

BARRINGTON, D. M., IMMERMAN, N., AND STRAUBING, H. 1990. On uniformity within NC^1. *Journal of Computer and System Sciences 41,* 3 (Dec.), 274–306.

BEAME, P., SAKS, M., SUN, X., AND VEE, E. 2003. Super-linear time-space tradeoff lower bounds for randomized computation. *Journal of the ACM 50,* 154–195.

BUSS, S. 1993. Algorithms for boolean formula evaluation and for tree contraction. In *Arithmetic, Proof Theory, and Computational Complexity,* P. Clote and J. J. Krajíček, Eds. Oxford Logic Guides, vol. 23. Oxford Science Publications, 96–115.

BUSS, S., COOK, S., GUPTA, A., AND RAMACHANDRAN, V. 1992. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing 21,* 755–780.

BUSS, S. R. 1987. The boolean formula value problem is in ALOGTIME. In *Proc. ACM Symp. on Theory of Computing (STOC).* 123–131.

CAUSSINUS, H., MCKENZIE, P., THÉRIEN, D., AND VOLLMER, H. 1998. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences 57,* 200–212.

CHANDRA, A. K., FORTUNE, S., AND LIPTON, R. J. 1985. Unbounded fan-in circuits and associative functions. *Journal of Computer and System Sciences 30,* 2, 222–234.

COOK, S. A. 1985. A taxonomy of problems with fast parallel algorithms. *Information and Control 64,* 1-3, 2–21.

COOK, S. A. 1988. Short propositional formulas represent nondeterministic computations. *Inf. Process. Lett. 26,* 5, 269–270.

ETESSAMI, K. 1997. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences 54,* 3 (Jun), 400–411.

FEIGE, U. AND KILIAN, J. 1998. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences 57,* 2, 187–199.

FORTNOW, L. 2000. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences 60,* 336–353.

FURST, M., SAXE, J. B., AND SIPSER, M. 1984. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory 17,* 13–27.

GOLDWASSER, S., GUTFREUND, D., HEALY, A., KAUFMAN, T., AND ROTHBLUM, G. N. 2007. Verifying and decoding in constant depth. In *Proc. ACM Symp. on Theory of Computing (STOC).* 440–449.

HÅSTAD, J. 1998. The shrinkage exponent of de Morgan formulas is 2. *SIAM Journal on Computing 27,* 1, 48–64.

HÅSTAD, J. 1999. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica 182,* 105–142.

HANSEN, K. A. AND KOUCKÝ, M. A new characterization of ACC^0 and probabilistic CC^0. *Computational Complexity*. To appear.

HÅSTAD, J. 1988. *Computational Limitations of Small Depth Circuits.* MIT Press, Cambridge.

HESSE, W., ALLENDER, E., AND BARRINGTON, D. A. M. 2002. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences 65,* 695–716.

IMPAGLIAZZO, R., PATURI, R., AND SAKS, M. E. 1997. Size-depth tradeoffs for threshold circuits. *SIAM Journal on Computing 26,* 693–707.

KHOT, S. 2001. Improved inaproximability results for maxclique, chromatic number and approximate graph coloring. In *Proc. IEEE Symp. on Found. of Comp. Sci. (FOCS).* 600–609.

KHOT, S. AND PONNUSWAMI, A. K. 2006. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *Proc. of the 33rd Intl. Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 4051. 226–237.

KOUCKÝ, M. 2009. Circuit complexity of regular languages. *Theory of Computing Systems 45,* 4, 865–879.

LINIAL, N., MANSOUR, Y., AND NISAN, N. 1993. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM 40,* 3, 607–620.

NAOR, M. AND REINGOLD, O. 2004. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM 51,* 2, 231–262.

NECIPORUK, E. I. 1966. On a Boolean function. *Doklady of the Academy of Sciences of the USSR 169,* 4, 765–766. English translation in Soviet Mathematics Doklady 7:4, pages 999-1000.

RAZBOROV, A. AND RUDICH, S. 1997. Natural proofs. *Journal of Computer and System Sciences 55,* 24–35.

RAZBOROV, A. A. 1987. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Mathematicheskie Zametki 41,* 598–607. English translation in Mathematical Notes of the Academy of Sciences of the USSR 41:333-338, 1987.

RAZBOROV, A. A. 1991. Lower bounds for deterministic and nondeterministic branching programs. In *Foundations of Computation Theory*. Lecture Notes in Computer Science, vol. 529. 47–60.

RAZBOROV, A. A. 1995a. Bounded arithmetic and lower bounds. In *Feasible Mathematics II,,* P. Clote and J. Remmel, Eds. Progress in Computer Science and Applied Logic, vol. 13. Birkhäuser, 344–386.

RAZBOROV, A. A. 1995b. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya Math. 59,* 205–227.

ROSSMAN, B. 2008. On the constant-depth complexity of k-clique. In *STOC.* 721–730.

RUZZO, W. L. 1981. On uniform circuit complexity. *Journal of Computer and System Sciences 22,* 3, 365–383.

SMOLENSKY, R. 1987. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. ACM Symp. on Theory of Computing (STOC)*. 77–82.

SRINIVASAN, A. 2003. On the approximability of clique and related maximization problems. *Journal of Computer and System Sciences 67,* 3, 633–651.

VAN MELKEBEEK, D. 2004. Time-space lower bounds for NP-complete problems. In *Current Trends in Theoretical Computer Science,* G. R. G. Paun and A. Salomaa, Eds. World Scientific Press, 265–291.

VAN MELKEBEEK, D. 2007. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science 2*, 197–303.

VOLLMER, H. 1999. *Introduction to Circuit Complexity*. Springer.

WILSON, C. 1990. Decomposability of NC and AC. *SIAM Journal on Computing 19*, 384–396.

YAO, A. C. C. 1985. Separating the polynomial-time hierarchy by oracles. In *Proc. IEEE Symp. on Found. of Comp. Sci. (FOCS)*. IEEE Computer Society Press, 1–10.

ZUCKERMAN, D. 2007. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing 3,* 1, 103–128.

August 19, 2008.