

Amplifying Circuit Lower Bounds Against Polynomial Time With Applications

Richard J. Lipton*
College of Computing
Georgia Institute of Technology
Atlanta, GA
rjl@cc.gatech.edu

Ryan Williams†
Computer Science Department
Stanford University
Stanford, CA
rrw@cs.stanford.edu

Abstract—We give a self-reduction for the Circuit Evaluation problem, and prove the following consequences.

- **Amplifying Size-Depth Lower Bounds.** If $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^k, n^{1-\delta}]$ for some k and δ , then for every $\varepsilon > 0$, there is a $\delta' > 0$ such that $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^{1+\varepsilon}, n^{1-\delta'}]$. Moreover, the resulting circuits require only $\tilde{O}(n^\varepsilon)$ bits of non-uniformity. As a consequence, strong enough depth lower bounds for Circuit Evaluation imply a full separation of P and NC (even with a weak size lower bound).
- **Lower Bounds for Quantified Boolean Formulas.** Let $c, d > 1$ and $e < 1$ satisfy $c < (1 - e + d)/d$. Either the problem of recognizing valid quantified Boolean formulas (QBF) is not solvable in $\text{TIME}[n^c]$, or the Circuit Evaluation problem cannot be solved with circuits of n^d size and n^e depth. This implies unconditional polynomial-time uniform circuit lower bounds for solving QBF.

I. INTRODUCTION

Recently, Allender and Koucký [AK10] have proved that if certain weak lower bounds hold at all, they can be *amplified*, in the sense that lower bounds of the form $n^{1+\varepsilon}$ can be extended to arbitrary n^k lower bounds for every k . For example, Allender and Koucký show that if the NC^1 -complete problem Balanced Formula Evaluation (BFE) is contained in TC^0 , then BFE has TC^0 circuits of size $n^{1+\varepsilon}$, for every $\varepsilon > 0$. It follows that even (seemingly) modest TC^0 lower bounds on BFE would imply that $\text{TC}^0 \neq \text{NC}^1$. Similar results are proved for other problems within NC^1 and NL. All of them have the form: “an $n^{1+\varepsilon}$ lower bound against constant-depth class \mathcal{C} implies arbitrary polynomial lower bounds against the same constant-depth class \mathcal{C} .”

These results suggest an intriguing approach to separating complexity classes: find a problem such that we can prove small concrete lower bounds, then find an “amplifying” result that extends the bound to arbitrary polynomials. For instance, if we could prove that

$$\text{SAT} \in \text{TIMESPACE}[n^{O(1)}, \log n] \text{ implies}$$

† Supported in part by a David Morgenthaler II Faculty Fellowship.

$$\text{SAT} \in \text{TIMESPACE}[n^{1.1}, n^7],$$

then we would separate NP from LOGSPACE, using time-space tradeoff lower bounds [For00], [FLvMV05], [BW12]. It is unclear whether we should expect to prove such an amplification result for SAT. Allender and Koucký show that any problem with their “oracle self-reduction” property must already be in NC. (However, this does not rule out the possibility of *using the assumption* that SAT is contained in logspace, to obtain an amplifying reduction for SAT.)

Main Results: First, we prove an amplification lemma for a P-complete problem, the Circuit Evaluation problem (CIRCEVAL), relative to low-depth circuit lower bounds. (The Circuit Evaluation problem is: given a circuit C and input x , determine if $C(x) = 1$.)

Let $\text{SIZEDEPTH}[s(n), d(n)]$ be the class of languages recognized by $s(n)$ -size $d(n)$ -depth circuits.

Theorem 1.1: The following are equivalent:

- There is a $k \geq 1$ and $\delta > 0$ such that $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^k, n^{1-\delta}]$.
- For all $\varepsilon > 0$, there is a $\delta' > 0$ such that $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^{1+\varepsilon}, n^{1-\delta'}]$. Moreover, these circuits need at most $\tilde{O}(n^\varepsilon)$ bits of non-uniformity; that is, the circuits can be generated in polynomial time by an algorithm that is given only $n^\varepsilon \cdot \text{poly}(\log n)$ bits of advice.

It follows that, in order to separate P from polynomial-size, $n^{o(1)}$ -depth circuits, it suffices to prove that CIRCEVAL is not in $\text{SIZEDEPTH}[n^{1.001}, n^{1-o(1)}]$ for circuits with $n^{.001}$ -bit descriptions. That is, any nontrivial size lower bound for sufficiently large depth implies $\text{NC} \neq \text{P}$. This is counterintuitive: we believe that considering $n^{1.001}$ size lower bounds should be an easier task than considering arbitrary polynomial-size lower bounds. Theorem 1.1 says that they are equivalent tasks for P problems in $n^{1-\delta}$ depth.

The amplifying reduction of Theorem 1.1 has applications to proving *unconditional* lower bounds. As the resulting

small circuits have low non-uniformity (even if the original polynomial size circuits did not), this reduction can be combined with other ingredients to prove unconditional time lower bounds for constructing low-depth circuits.

It has been known since Kannan [Kan82] that for every k , there is an $L_k \in \text{NP}$ that does not have P-uniform circuits of size $O(n^k)$. (That is, no polynomial time algorithm can construct $O(n^k)$ -size circuits computing L_k .) However, it is an open problem to produce a *natural* problem of interest (such as SAT) that does not have such circuits (for some $k \geq 1$). Even producing a natural problem in PSPACE with this property is open. Perhaps the most natural candidate is the canonical PSPACE-complete problem of Quantified Boolean Formulas (QBF). Although we do not believe QBF has *non-uniform* NC circuits (i.e., we believe $\text{PSPACE} \not\subseteq \text{NC/poly}$), and we know that QBF does not have LOGSPACE-uniform NC circuits (by the space hierarchy theorem), it is open whether PSPACE has P-uniform NC circuits [All89]. Indeed, it was even open if QBF had $n^{1+o(1)}$ -time uniform NC circuits (Eric Allender, personal communication).

Combining the ideas in Theorem 1.1 with other ingredients, we can obtain non-linear time lower bounds for generating low-depth circuits solving QBF:

Theorem 1.2: Let $c, d \geq 1$ and $e < 1$ satisfy the inequality $c < (1 - e + d)/d$. Either

- QBF $\notin \text{TIME}[n^{c+o(1)}]$, or
- CIRCEVAL does not have circuits of $n^{d+o(1)}$ size and $n^{e+o(1)}$ depth.

Three interesting settings to c, d (letting $e = 0$) yield the following corollaries for multitape Turing machines:

Corollary 1.1: QBF does not have $O(n^{1.618})$ -time uniform circuits of depth $n^{o(1)}$.

Corollary 1.2: QBF does not have $O(n^{2-\varepsilon})$ -time circuits of $n^{1+o(1)}$ size and $n^{o(1)}$ depth, for all $\varepsilon > 0$.

Corollary 1.3: Either QBF cannot be solved in $n^{1+\varepsilon}$ time for some $\varepsilon > 0$, or $\text{P} \not\subseteq \text{NC/poly}$.

Notice that the underlying algorithms are *not* space-restricted; they could spend superlinear space constructing the low-depth circuits.

Prior work has shown how slightly-faster-than-brute-force algorithms for hard problems can imply circuit lower bounds for EXP^{NP} and NEXP (e.g., [IKW02], [KI04], [Wil11]). At the other end of the spectrum, one may investigate the consequences of ultra-efficient algorithms for hard problems, such as nearly-linear time algorithms (which may construct low-depth circuits) for QBF, to explore the space of absurdities that result. Such algorithms can sometimes yield circuit

lower bounds so strong that they contradict the existence of the algorithm. This is one way of looking at Theorem 1.2 and its corollaries.

II. PRELIMINARIES

Our standard model of computation in this paper is the multitape Turing machine, although sometimes random-access machines suffice. When we consider *alternating* computation, it is important to keep in mind that multitape Turing machines and random access machines are known to be equivalent in running time (up to polylogarithmic extra factors). That is, every alternating random-access machine running in time t (where $t(n) \geq n$) can be simulated by an alternating multitape TM running in time $t \cdot \text{poly}(\log t)$ (Gurevich and Shelah [GS89] proved this result for nondeterministic computation, but their proof extends to alternating computation straightforwardly).

Moreover, recall that, via the Hennie-Stearns simulation [HS66], multitape TMs and two-tape TMs are time-equivalent up to polylog factors, and this simulation holds equally well for alternating computation (the two-tape TM may write a record of all the alternating “guesses” at the beginning of the computation, then simulate the multitape TM on those guesses). Hence we may assume the two-tape Turing machine model in our simulations. In any case, for simplicity we often say “algorithm” instead of the specific computational model.

In this paper, any bound of the form $n \cdot \text{poly}(\log n)$ is called *quasi-linear*. We define DQL (Deterministic Quasilinear Time) be the class of languages solvable by a multitape Turing machine in $O(n \text{poly}(\log n))$ time [NRS95].

We say that a language L^* is *DQL-hard* if for all $L \in \text{DQL}$, there is a *polylog-time* reduction R from L to L^* . More precisely, there is a constant c and an algorithm A that is given random access to (x, i) where i ranges from 1 to $n(\log n)^c$, such that A outputs the i th bit of $R(x)$ in $O((\log n)^c)$ time. (An alternative definition is that A can be implemented by a multitape Turing machine in $\text{poly}(\log n)$ time, given $O(n)$ initial steps to first position the tape heads, during which period the TM cannot write. In this paper, the reductions can be revised to fit both definitions.)

Our specific formulation of the quantified Boolean formula problem (QBF) is: given a sentence of the form

$$\psi = (\exists x_1)(\forall x_2) \cdots (\exists x_n)\phi(x_1, \dots, x_n)$$

where ϕ is a 3-CNF formula over Boolean variables x_1, \dots, x_n , determine if ψ is true. It is well known that QBF is PSPACE-complete. One property we use of QBF can be derived from the DQL-hardness of CIRCEVAL. Define $\text{ATIME}[t(n)]$ to be the class of problems solvable

in $t(n)$ time on an alternating Turing machine. Define AQL (Alternating Quasilinear Time) as the class

$$\text{AQL} := \bigcup_{k \geq 1} \text{ATIME}[n \cdot (\log^k n)].$$

QBF is known to be complete for AQL under a strong reducibility notion. The following theorem is implicit in work of Gurevich and Shelah [GS89]; see also Fortnow [For00].

Theorem 2.1: QBF is AQL-complete under polylog-time reductions.

Some Circuit Definitions: Let $\text{SIZEDEPTH}[s(n), d(n)]$ be the class of languages recognized by a $s(n)$ -size $d(n)$ -depth circuit family.

For a machine-based complexity class \mathcal{C} , a circuit family $\{C_n\}$ is \mathcal{C} -uniform if there is a multitape Turing machine M obeying the constraints of class \mathcal{C} which produces a description of C_n on the input 1^n . Note that, if $\mathcal{C} \subseteq \text{TIME}[t(n)]$ for some t , then the number of gates in such a C_n can be no more than $t(n)$.

We say that a circuit family $\{C_n\}$ has $b(n)$ bits of uniformity if there is an infinite family of strings $\{s_n\}$, where $|s_n| \leq b(n)$, and a polynomial time algorithm A , such that $A(1^n, s_n)$ outputs C_n . That is, the n^{th} circuit in the family can be efficiently described with only $b(n)$ bits.

By a counting argument, most $s(n)$ -size circuit families do not have less than $O(s(n) \log s(n))$ bits of non-uniformity, so the set of circuit families with low uniformity (say, $n^{0.01}$ bits) is quite restricted.

Some Related Prior Work: Lipton and Viglas [LV03] proved that if $\text{P} \subset \text{NC}/\text{poly}$ then $\text{TIME}[t] \subseteq \text{SPACE}[t^{1-\varepsilon}]$ for some $\varepsilon > 0$. (Compare with Corollary 4.4.) Their proof also applies block-respecting Turing machines in a similar but not identical way.

Several uniform circuit lower bounds are known for SAT and the Permanent function. Allender and Koucký [AK10] prove there exists a $\delta > 0$ such that SAT cannot be solved with $n^{1+\delta}$ size LOGTIME-uniform TC^0 circuits. In the pioneering time-space tradeoffs work of Fortnow [For00], he also proved that SAT does not have LOGSPACE-uniform branching programs of $n^{1+o(1)}$ size, and proves a result (due to Buhrman) that SAT does not have LOGSPACE-uniform NC^1 circuits of $n^{1+o(1)}$ size. Allender *et al.* [AKRRV01] extended this to prove that SAT does not have LOGSPACE-uniform SAC^1 circuits of $n^{1+o(1)}$ size.

For the Permanent function, Allender [All99] proved that the Permanent does not have LOGTIME-uniform TC^0 circuits of quasi-polynomial size, which uses the fact that the Permanent is $\#\text{P}$ -complete under efficient reductions. Koiran and Perifel [KP09] showed that the Permanent cannot be computed by LOGTIME-uniform polynomial-size thresh-

old circuits of $o(\log \log n)$ depth.

To compare, we show that QBF does not have n^e depth circuits constructible in n^c time, for various $c > 1$ and $e < 1$. That is, we relax the depth restriction on the circuits, use a different uniformity notion (polynomial-time bounded, rather than logarithmic-space bounded), but prove lower bounds for an apparently harder problem. This requires a different approach from the lower bounds arising out of time-space tradeoffs for SAT, which crucially rely on clever simulations of severely space-restricted machines.

III. AMPLIFYING CIRCEVAL LOWER BOUNDS

We start with a completeness result for CIRCEVAL that is apparently folklore, but not well known:

Theorem 3.1: (Folklore) CIRCEVAL is DQL-complete under polylog-time reductions. That is, for all $L \in \text{DQL}$, there is a quasi-linear size circuit family $\{C_n\}$ such that:

- $x \in L$ if and only if $C_{|x|}(x) = 1$, and
- $\{C_n\}$ is polylog-time uniform: given n and a gate index i of C_n (both encoded in $O(\log n)$ bits), in $\text{poly}(\log n)$ time we can compute the tuple $\langle j_1, j_2, g \rangle$ where j_1 and j_2 are the indices of the two possible gates in C_n which are input to gate i , and g is the gate type of gate i .

Proof: Let M be a quasilinear time machine. We say that M is *oblivious* if for every n and input x of length n , the tape head movements of $M(x)$ depend only on n . That is, M makes precisely the same sequence of head movements on every input x of length n . Hennie and Stearns [HS66] showed that for every multitape Turing machine M running in $O(t(n))$ time there is an equivalent *oblivious* M' that uses only two tapes and runs in $O(t(n) \log t(n))$ time. From their simulation, it is easy to construct (in quasilinear time) a circuit C' on $O(t(n) \log t(n))$ gates that simulates M' on n -bit inputs, completing the reduction from $L(M)$ to CIRCEVAL. (Note that Pippenger and Fischer [PF79] gave a simplified, recursive construction.)

This reduction can be made *polylog-time* uniform: if we want to compute the i th bit of the description of the resulting CIRCEVAL instance, this can be done in $\text{poly}(\log n)$ time (independently of the other bits). The proof that polylog-time uniformity is possible is essentially the same as the proof of Fortnow *et al.* [FLvMV05] that SAT is NP-complete under polylog-time uniform reductions. Their proof observes that the oblivious simulation of Hennie-Stearns/Pippenger-Fischer is extremely regular: given the index i of a given timestep (encoded in $O(\log n)$ bits), we can efficiently compute (in $\text{poly}(\log n)$ time) the head positions of the two tapes at timestep i . (We sketch some details of why this is true in the Appendix.) Therefore, we can efficiently produce the $O(1)$ gates of the circuit that simulate the transition

function of M' at timestep i .

It remains to show that CIRCEVAL itself is in DQL. First, given a circuit C as a list of tuples (i, j_1, j_2, g) , indicating that gate i has gate type g and takes its inputs from the outputs of gates j_1, j_2 , topologically sort the nodes of the directed acyclic graph corresponding to C . Then evaluate each gate of the circuit in increasing topological order, starting with the inputs. On a standard algorithm model, this is a simple dynamic program that can be solved in linear time. For multitape TMs, Fischer ([Fis74], pp.13–14) showed how to solve this problem in quasilinear time on a multitape Turing machine.

In more detail, Fischer solved the *mail-carrier problem*: a mail carrier has a number of pushdown stacks containing letters to be delivered. Each letter has a number $i \in \{1, \dots, n\}$ corresponding to a house number. The mail carrier visits houses $1, \dots, n$ in that order, and must deliver and receive letters online (at house i , all letters numbered i must be delivered, and all letters that i wishes to send must be pushed on some stack before going to house $i + 1$). Fischer shows that this problem can be solved in such a way that each letter is handled $O(\log n)$ times on a multitape Turing machine. To apply this to CIRCEVAL, imagine that we visit each gate of C in increasing topological order. Each gate i receives letters addressed to it (its input bits) and uses that to compute the output bit, which is sent in letters to the gates later in the topological order that i points to. ■

Next, we prove that if CIRCEVAL has low-depth circuits at all, it must have small low-depth circuits. Moreover, these small low-depth circuits have sublinear-size representations. The proof of this implication stems from the fact that CIRCEVAL can be solved efficiently on multitape Turing machines, and that computation is rather local on multitape TMs: the computation can be divided into t/b time blocks of $O(b)$ steps each, where only $O(b)$ bits of input are read in each time block, and $O(b)$ bits of output are generated. By replacing each block with a low-depth circuit, we get a “somewhat low depth” circuit that has small size.

Lemma 3.1 (Size Reduction Lemma): If CIRCEVAL can be recognized in $t(n)$ time on a multitape Turing machine and CIRCEVAL has (non-uniform) circuits of size $s(n)$ and depth $d(n)$, then for all time constructible $b(n)$ such that $\Omega(\log n) \leq b(n) \leq t(n)$, there is a $\tilde{b}(n) \leq b(n) \cdot \text{poly}(\log b(n))$ such that CIRCEVAL has circuits of

$$\begin{aligned} &O(t(n) \log t(n) \cdot s(\tilde{b}(n))) \text{ size,} \\ &O\left(\frac{t(n) \cdot \log t(n)}{b(n)} \cdot d(\tilde{b}(n))\right) \text{ depth, and} \\ &O(s(\tilde{b}(n)) \log s(\tilde{b}(n))) \text{ bits of non-uniformity.} \end{aligned}$$

Moreover, assuming we are given these bits of non-uniformity as additional input, the resulting circuits for CIRCEVAL are constructible in $O(t(n) \cdot \text{poly}(\log t) \cdot s(\tilde{b}(n)))$ time.

Proof: Let M be a multitape Turing machine for CIRCEVAL that runs in $O(t(n))$ time. First, we convert M into an oblivious M' , as in Theorem 3.1. M' runs in $t'(n) = O(t(n) \log t(n))$ time. As mentioned in the proof of Theorem 3.1, this oblivious two-tape simulation is polylog-time uniform, in that we can determine the head positions in step i in $\text{poly}(\log t)$ time.

Next, the computation of M' can be partitioned using an old trick of Hopcroft, Paul, and Valiant [HPV77], who call the resulting machines *block-respecting*. The result is a reimplementaion of M' so that its computation of $O(t'(n))$ time can be partitioned into $\frac{t'(n)}{b(n)}$ time blocks of $O(b(n))$ time each, and each tape is partitioned into $O(\frac{t'(n)}{b(n)})$ blocks of $O(b(n))$ cells each. For every time block, the tape heads stay within exactly one tape block, for each tape. So we can think of each time block as running on an input of $O(b(n))$ bits, and each block outputs $O(b(n))$ bits (the new content of those tape blocks). Note that the Hopcroft-Paul-Valiant simulation maintains the obliviousness of M' .

Given that M' is oblivious, the block-respecting simulation is also uniform, in the following sense: for every n -bit input x to M' we can efficiently construct a *computation graph* $G_{M'}$ on $N = \frac{t'(n)}{b(n)}$ nodes, where each node $v \in \{1, \dots, N\}$ represents a time block v , and there is an edge (v, v') if the $O(b(n))$ bits of content output by block v is needed as input to the computation of block v' . More precisely, an edge (v, v') can arise in two ways. Either:

- v immediately precedes v' in the computation, in which case the final state and head positions of the computation in block v is needed to give the initial state and head positions of block v' , or
- block v writes content to some section of tape, and this section of tape is not read again until block v' .

Because M' is oblivious, the graph $G_{M'}$ is the same for all n -bit inputs. Note that $G_{M'}$ has $O(N)$ edges.

Because CIRCEVAL is DQL-hard, the problem of computing the outputs of one block (given the inputs) can be efficiently reduced to $O(b(n))$ instances of CIRCEVAL, each instance having size $\tilde{b}(n) = b(n) \cdot \text{poly}(\log b(n))$. Furthermore, if CIRCEVAL has $s(n)$ -size $d(n)$ -depth circuits, then each of these $O(b(n))$ instances can be computed with an $S = s(\tilde{b}(n))$ size, $D = d(\tilde{b}(n))$ depth circuit, call it C .

Suppose we construct a Boolean circuit C' which replaces all nodes of $G_{M'}$ with copies of C , feeding the $O(b(n))$ -bit outputs of the various copies of C to the appropriate inputs of other copies, as dictated by the edges of $G_{M'}$. Note that by properties of the block-respecting simulation, C' can be uniformly constructed, given access to C . As there are $N = \frac{t'(n)}{b(n)}$ blocks and $O(b(n))$ circuits per block, the final circuit for CIRCEVAL has $O(t(n) \log t(n) \cdot S)$ size and

$O(\frac{t(n) \log t(n)}{b(n)} \cdot D)$ depth, and the amount of non-uniformity needed is just $O(S \log S)$ bits, to encode C . ■

Since every language in $\text{TIME}[n^c]$ has $n^{c+o(1)}$ size circuits, the proof of Lemma 3.1 yields the following important corollary:

Corollary 3.1: Let $T(n)$ be time constructible. Assume CIRCEVAL has (non-uniform) circuits of size $s(n)$ and depth $d(n)$. Then for all $b(n)$ as quantified above, every $L \in \text{TIME}[T(n)]$ has circuits of

$$\begin{aligned} & T(n) \cdot s(b(n)^{1+o(1)}) \text{ size and} \\ & \frac{T(n)^{1+o(1)}}{b(n)} d(b(n)^{1+o(1)}) \text{ depth, with} \\ & s(b(n)^{1+o(1)}) \text{ bits of non-uniformity.} \end{aligned}$$

Proof: (Sketch) Reduce the given L to a CIRCEVAL instance of $s' = T(n)^{1+o(1)}$ size, apply the block-respecting simulation to the s' -size circuit with blocks of size $b(n)$, then apply the small-circuit assumption on CIRCEVAL to convert each of those block computations to circuits of $s(b(n)^{1+o(1)})$ size and $\frac{T(n)^{1+o(1)}}{b(n)} d(b(n)^{1+o(1)})$ depth. ■

The lemma also has the following intriguing consequence which, on the face of it, has nothing to do with Turing machines.

Reminder of Theorem 1.1 The following are equivalent:

- There is a $k \geq 1$ and $\delta > 0$ such that $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^k, n^{1-\delta}]$.
- For every $\varepsilon > 0$, there is a $\delta' > 0$ such that $\text{CIRCEVAL} \in \text{SIZEDEPTH}[n^{1+\varepsilon}, n^{1-\delta'}]$. Moreover, these circuits have only $O(n^\varepsilon)$ bits of non-uniformity.

Proof: One direction is obvious. For the other, we only have to set $b(n) = n^{\frac{\varepsilon}{k}}$ and set $\delta' = \delta \cdot \varepsilon/k$ in the Size Reduction Lemma. We obtain circuits for CIRCEVAL which have size $n^{1+k \cdot (\varepsilon/k) + o(1)} = n^{1+\varepsilon+o(1)}$ and depth $n^{1-\varepsilon/k + (1-\delta) \cdot \varepsilon/k + o(1)} = n^{1-\delta'+o(1)}$, with $n^{\varepsilon+o(1)}$ non-uniformity. The $o(1)$ factors can be neglected by simply adjusting δ' and $b(n)$ to be a little smaller. ■

Corollary 3.2: Suppose CIRCEVAL does not have circuits of $n^{1.1}$ size and $n^{1-\delta}$ depth, for every $\delta > 0$. Then $\text{NC} \neq \text{P}$.

Hence even a modest size lower bound for small depth circuits is already enough to separate P from NC.

IV. LOWER BOUNDS FOR QUANTIFIED BOOLEAN FORMULAS

It is a major open problem to prove that QBF is not solvable in $n^{1+\varepsilon}$ time, for some $\varepsilon > 0$. The only relevant result (to our knowledge) is a barely-superlinear time lower bound for a non-standard encoding of QBF [Will08]. A preliminary step would be to establish that QBF does not have $n^{1+\varepsilon}$

time algorithms under further restrictions on the algorithms. For one-tape Turing machines, it has been known since the 1960's that PALINDROMES requires $\Omega(n^2)$ time, so QBF lower bounds are easy in that setting [Cob66]. For algorithms using only $n^{1+\varepsilon}$ time and $n^{o(1)}$ space simultaneously, it is also easy to prove lower bounds: for example, it is well known that $\text{PSPACE} \not\subseteq \text{LOGSPACE-uniform NC}$, simply because the latter can be simulated in polylogarithmic space.

An interesting extension would be to prove that $\text{PSPACE} \not\subseteq \text{P-uniform NC}$. (The class is also called PUNC in the literature [All89].) This separation problem asks if we can solve QBF efficiently in parallel, provided we are allowed polynomial-time preprocessing (in serial) to prepare the parallel algorithm for its computation. Such a separation would bring us a little closer to proving $\text{P} \neq \text{PSPACE}$. However, currently the separation of PSPACE from P-uniform NC is still open.

The amplification lemma of the previous section can be applied to prove new uniform circuit lower bounds for solving quantified Boolean formulas. At a high level, the general idea is similar to that of time-space tradeoff lower bounds for SAT [For00], [FLvMV05], [vMe06]. First we use the assumption that QBF has nice circuits to “speed up” generic deterministic time-bounded computations (using alternations in our sped-up simulation of deterministic time). Then we use the assumption again to remove these alternations efficiently, resulting in a contradiction to the time hierarchy theorem. However, the actual arguments involved in our lower bounds are ultimately quite different from those of the SAT lower bounds. In particular, the “speed-up” portion of our simulation only works conditioned on the assumption that P has low-depth circuits, unlike in the time-space setting (where the “speed up” simulation used there works unconditionally).

Reminder of Theorem 1.2 Let $c, d \geq 1$ and $e < 1$ satisfy $c < (1 - e + d)/d$. Either

- $\text{QBF} \notin \text{TIME}[n^{c+o(1)}]$, or
- CIRCEVAL does not have circuits of $n^{d+o(1)}$ size and $n^{e+o(1)}$ depth.

Proof: Suppose that both

- (A) $\text{QBF} \in \text{TIME}[n^{c+o(1)}]$ and
- (B) CIRCEVAL has circuits of $n^{d+o(1)}$ size and $n^{e+o(1)}$ depth.

We wish to conclude a contradiction.

Combining assumption (A) with the completeness of CIRCEVAL for quasilinear time (Theorem 3.1), we can infer that every QBF instance of length n can be reduced in $n^{c+o(1)}$ time to a CIRCEVAL instance of size $n^{c+o(1)}$ with n input bits. The completeness of QBF for alternating quasi-

linear time (Theorem 2.1) means that, if QBF is solvable in $n^{c+o(1)}$ time, then every language in $\text{ATIME}[O(n)]$ is in $n^{c+o(1)}$ time. By a standard padding/translation argument, this entails that for all polynomials $t(n)$ we have

$$\text{ATIME}[t] \subseteq \text{TIME}[t^{c+o(1)}]. \quad (1)$$

Assuming (B) is true, we will derive a “fast” simulation of deterministic time:

$$\text{TIME}[t^{c+o(1)}] \subseteq \text{ATIME}[t^{c(1-\varepsilon/d)+e\varepsilon/d+o(1)} + t^{c\varepsilon+o(1)}], \quad (2)$$

for all $\varepsilon \in (0, d)$. To minimize the running time of the simulation, we set

$$c(1 - \varepsilon/d) + e\varepsilon/d = c\varepsilon.$$

and obtain

$$\varepsilon = d/(d + 1 - e).$$

That is, from assumption (B) we conclude that there is a simulation of every t^c time multitape Turing machine that runs in alternating time $t^{cd/(d+1-e)+o(1)}$. But this exponent is less than 1, provided that $c < (1 - e + d)/d$. Hence the combination of (1) and (2) yields a simulation of every language in alternating time $t(n)$ that runs in alternating time $o(t(n))$. This is a contradiction to the alternating time hierarchy, which says that

$$\text{ATIME}[o(t(n))] \subsetneq \text{ATIME}[t(n)].$$

Let us discuss how to prove (2). Because CIRCEVAL is complete for quasi-linear time under $\text{poly}(\log n)$ time reductions (Theorem 3.1), we can think of an arbitrary $\text{TIME}[t^{c+o(1)}]$ machine M on an input of length n as a CIRCEVAL instance C of size $t^{c+o(1)}$, without loss of generality. Furthermore, we may also assume without loss of generality that our alternating simulation of CIRCEVAL runs on an alternating *random-access* machine. (As mentioned in the Preliminaries, alternating RAMs running in time t can be simulated by an alternating 2-tape Turing machine running in $t \cdot \text{poly}(\log t)$ time; this is due to Gurevich and Shelah [GS89]. See also Paul, Prouss, and Reischuk [PPR80] and Fortnow and Lund [FL93].)

By Corollary 3.1 of Lemma 3.1 (letting $s(n) = n^{d+o(1)}$, $d(n) = n^{e+o(1)}$, and $b(n) = t^{c\varepsilon/d}$) and assumption (B), there exists a circuit D that can simulate M on inputs of length n (and hence, the circuit C), where D has at most $t^{O(dc)}$ size, $t^{c(1-\varepsilon/d)+e\varepsilon/d+o(1)}$ depth, and $t^{c\varepsilon+o(1)}$ bits of non-uniformity. The bits of non-uniformity encode a small $t^{c\varepsilon+o(1)}$ -size circuit D' for CIRCEVAL on instances of size $t^{c\varepsilon/d}$. Given free access to a copy of D' , we can efficiently construct the circuit D in $t^{O(dc)}$ time, by constructing the computation graph G with $t^{c+o(1)-c\varepsilon/d}$ nodes of Lemma 3.1, and replacing each of its nodes with copies of D' .

We can simulate the evaluation of C on an input x in alternating time $t^{c(1-\varepsilon/d)+e\varepsilon/d+o(1)} + t^{c\varepsilon+o(1)}$, as follows. First, *existentially* guess the computation graph G for M on $t^{c+o(1)-c\varepsilon/d} = t^{c(1-\varepsilon/d)+o(1)}$ nodes, with nodes indexed by time blocks. For each node v of G representing a time block, guess strings $s_{v,1}$ and $s_{v,2}$ of $O(\log t(n))$ bits indicating which $b(n)$ -length block of tape cells (on each of the two tapes of M) is being accessed in time block v . Next, verify that G is consistent with respect to $s_{v,1}$ and $s_{v,2}$, for all v . That is, verify that:

- vertex 1 has $s_{1,1} = s_{1,2} = 1$
(initially, the first block of cells is being read, on both of the tapes)
- for all i and $j = 1, 2$, $s_{i+1,j} \in \{s_{i,j} - 1, s_{i,j}, s_{i,j} + 1\}$
(from one time block to the next, the index of the tape block can change by at most 1)
- for all $i > 1$, there are edges $(j_1, i), (j_2, i)$ in G where $j_1, j_2 < i$, and $s_{j_1,1} = s_{i,1}$, $s_{j_2,2} = s_{i,2}$
(for each time block i , there two previous time blocks j_1, j_2 , the outputs of which are inputs to i , and time blocks j_1, j_2 wrote to the same two tape blocks that i is reading)

If G does not satisfy these properties, then *reject*.

Then, *existentially* guess a $t^{c\varepsilon+o(1)}$ -bit string that is intended to represent D' which according to assumption (B) may be assumed to have $t^{e\varepsilon/d+o(1)}$ depth. To check that this guess is correct, *universally* guess an $t^{c\varepsilon/d}$ -bit input y to D' , and verify (in $t^{c\varepsilon+o(1)}$ time) that $D'(y) = \text{CIRCEVAL}(y)$ (if not, then *reject*). Now we are assured that the guessed small circuit D' is correct.

The circuit D can be simulated on the input (C, x) with a $t^{c(1-\varepsilon/d)+e\varepsilon/d+o(1)}$ -depth uniform circuit using the small circuit D' , along with the graph G and the sequences $s_{v,j}$ to route the outputs of $O(b(n))$ copies of D' (the output of a time block) to the appropriate inputs of other D' copies (the input to another time block). This low-depth simulation itself can be performed in $t^{c(1-\varepsilon/d)+e\varepsilon/d+o(1)}$ alternating time on a random-access machine, via the standard simulation of uniform size- $t^{O(t)}$ depth- d circuits in alternating time $O(d + \log t)$ (cf. Borodin [Bor77]).

Note that, during the course of the low-depth simulation, it is possible that the guessed sequence $s_{i,j}$ is found to be incorrect for some i and j (e.g., the tape head of a block moves off the right at the end of the block, yet the corresponding index does not increase by 1). In such a case, the simulation rejects. ■

Merging the two lower bounds of Theorem 1.2 into one, we obtain a non-linear time lower bound for constructing

low-depth circuits solving quantified Boolean formulas:

Corollary 4.1: For all $\varepsilon > 0$, QBF cannot be solved by circuits of n^d size and n^e depth constructible in n^c time, where $c, d \geq 1$ and $e < 1$ satisfy $c < (1 - e + d)/d$.

Notice that we must have $d \leq c$, as no circuit of size greater than n^c can be constructed in n^c time.

Proof: The assumption that QBF has such circuits implies that QBF is in $n^{c+o(1)}$ time and that QBF has n^d size, n^e depth circuits. Since CIRCEVAL can be reduced in quasilinear time (and polylogarithmic time per bit) to QBF (Theorem 2.1), there is a polylogarithmic depth, quasilinear size circuit C which takes instances of CIRCEVAL and outputs equivalent instances of QBF. Composing this circuit with a circuit of n^d size and n^e depth for QBF, we conclude that CIRCEVAL has $n^{d+o(1)}$ size, $n^{e+o(1)}$ depth circuits. Hence the lower bound of Theorem 1.2 applies. ■

Of course, we believe that for all c there is no n^c time algorithm for QBF (i.e., $P \neq PSPACE$), and that for all d , there are no n^d size, $n^{o(1)}$ depth circuits for QBF (i.e., $PSPACE \not\subseteq NC/poly$).

Let us instantiate some interesting values of c, d, e in Corollary 4.1. If $d = 1$ and $e = o(1)$, then $c = 2 - \varepsilon$ suffices to obtain a lower bound.

Corollary 4.2: For all $\varepsilon > 0$, QBF cannot be solved by circuits of $n^{1+o(1)}$ size and $n^{o(1)}$ depth constructible in $n^{2-\varepsilon}$ time.

When $c = d$ and $e = o(1)$, then it suffices to set $c < \phi$, where $\phi = 1.618 \dots$ is the golden ratio.

Corollary 4.3: For all $c < \phi$, QBF cannot be solved by circuits of n^c size and $n^{o(1)}$ depth constructible in n^c time.

That is, there is no $n^{1.618}$ time algorithm which, given the input 1^n , prints an $O(n^{1.618})$ size, $n^{o(1)}$ depth circuit, which can then be used to solve QBF instances of size n .

Suppose $c = 1$ and $e = o(1)$ in Theorem 1.2. Then d can be any constant, and we have:

Corollary 4.4: Either QBF cannot be solved in $n^{1+\varepsilon}$ time for some $\varepsilon > 0$, or $P \not\subseteq NC/poly$.

Finally, any non-trivial depth reduction in circuits for solving QBF results in a non-trivial size lower bound:

Corollary 4.5: For every $\varepsilon > 0$, there is a $c > 1$ such that QBF cannot be solved by circuits of n^c size and $n^{1-\varepsilon}$ depth, constructible in n^c time.

Proof: Let $e = 1 - \varepsilon$ and $c = d$. To obtain the lower bound from Corollary 4.1, we need $c < (\varepsilon + c)/c$, which holds for all sufficiently small c . ■

V. DISCUSSION

In this paper, we have applied some fairly old techniques to prove new lower bounds. Every component we use was essentially known by the 1980's. Perhaps the major difference between then and now is our perspective. What else might we have missed? For example, could a clever recursive argument finally show that PSPACE is not contained in P-uniform NC? Could the techniques here be used (along with other ideas) to finally prove that QBF is not in $n^{1.1}$ time?

An interesting consequence of the Size Reduction Lemma is that, to prove a non-uniform size-depth lower bound for CIRCEVAL, it suffices to prove one for circuits with succinct descriptions ($o(n)$ bits of non-uniformity). This is very intriguing, because it looks far easier to prove lower bounds against succinctly described circuits, compared to arbitrary circuits. For example, it is not difficult to show that there is a language in $O(n \log^4 n)$ time that does not have linear-size circuits with n -bit descriptions:

Proposition 1: $\text{TIME}[n \log^4 n] \not\subseteq \text{TIME}[O(n \log^2 n)]/n$.

Proof: (Sketch) The proof idea is identical to that of Turlakis [Tou01], who proved $\text{NTIME}[n^k] \not\subseteq \text{coNTIME}[o(n^k)]/n$. The diagonalizing machine M running in $O(n \log^4 n)$ time simulates the $|x|$ th linear time machine $M_{|x|}$ on input x , treating x also as the advice string, and outputs the opposite result. For every input length n and linear time M_n/y with advice string y , $M(y) \neq (M_n/y)(y)$. ■

However, if the restriction to n -bit descriptions is removed, it is possible that every polynomial-time computable language has linear-size circuits. Therefore this restriction to low-uniform circuits is rather significant, with respect to what we can prove.

One intermediate direction for future work is to try to understand the space of these new lower bound proofs for QBF. For SAT time-space tradeoffs, the known proof methods are now extremely well-understood, in terms of their power and limits [BW12]. The proofs for QBF here have a similar flavor: one component turns alternating time computations into deterministic time computations, and another component simulates deterministic time computations faster using alternations. Can we generalize and rigorously formalize the methods used here, and build an automatic search that can find better lower bounds?

Related to this question, recently we have been able to strengthen our time lower bound against constructing low-depth circuits for QBF, to a nearly-quadratic time lower bound. The details will appear in the journal version.

REFERENCES

- [All89] E. Allender. P-uniform circuit complexity. *J. ACM* 36(4):912–928, 1989.
- [All99] E. Allender. The Permanent Requires Large Uniform Threshold Circuits. *Chicago J. Theor. Comput. Sci.*, 1999.
- [AKRRV01] E. Allender, M. Koucký, D. Ronneburger, S. Roy, and V. Vinay. Time-Space Tradeoffs in the Counting Hierarchy. *IEEE Conference on Computational Complexity* 295–302, 2001.
- [AK10] E. Allender and M. Koucký. Amplifying Lower Bounds by Means of Self-Reducibility. *JACM* 57(3), 2010.
- [BGHSV05] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Short PCPs verifiable in polylogarithmic time. In *IEEE Conference on Computational Complexity*, 120–134, 2005.
- [Bor77] A. Borodin. On Relating Time and Space to Size and Depth. *SIAM J. Comput.* 6(4): 733–744, 1977.
- [BW12] S. R. Buss and R. Williams. Limits on Alternation-Trading Proofs for Time-Space Lower Bounds. To appear in *IEEE Conference on Computational Complexity*, 2012.
- [Cob66] A. Cobham. The recognition problem for the set of perfect squares. In *IEEE Conference on Switching and Automata Theory*, 78–87, 1966.
- [Fis74] M. J. Fischer. Lecture Notes on Network Complexity. Yale Technical Report 1104, June 1974.
- [For00] L. Fortnow. Time-Space Tradeoffs for Satisfiability. *J. Comput. Syst. Sci.* 60(2):337–353, 2000.
- [FLvMV05] L. Fortnow, R. J. Lipton, D. van Melkebeek, and A. Viglas. Time-space lower bounds for satisfiability. *J. ACM* 52(6):835–865, 2005.
- [FL93] L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity. *Theoretical Computer Science A*, 113:55–73, 1993.
- [GS89] Y. Gurevich and S. Shelah. Nearly-linear time. In *Logic at Botik*, Springer LNCS 363:108–118, 1989.
- [HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *JACM* 13(4):533–546, 1966.
- [HPV77] J. Hopcroft, W. Paul, and L. Valiant. On time versus space. *JACM* 24(2):332–337, 1977.
- [IKW02] R. Impagliazzo, V. Kabanets, and A. Wigderson. In Search of an Easy Witness: Exponential Time vs. Probabilistic Polynomial Time. *Journal of Computer and System Sciences* 65(4):672–694, 2002.
- [KI04] V. Kabanets and R. Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity* 13(1-2):1–46, 2004.
- [Kan82] R. Kannan. Circuit-Size Lower Bounds and Non-Reducibility to Sparse Sets. *Information and Control* 55(1-3):40–56, 1982.
- [KP09] P. Koiran and S. Perifel. A Superpolynomial Lower Bound on the Size of Uniform Non-constant-depth Threshold Circuits for the Permanent. *IEEE Conference on Computational Complexity*, 35–40, 2009.
- [LV03] R. J. Lipton and A. Viglas. Non-Uniform Depth of Polynomial Time and Space Simulations. *International Symposium on Fundamentals of Computation Theory*, 311–320, 2003.
- [vMe06] D. van Melkebeek. A Survey of Lower Bounds for Satisfiability and Related Problems. *Foundations and Trends in Theoretical Computer Science* 2(3):197–303, 2006.
- [NRS95] A. V. Naik, K. W. Regan, and D. Sivakumar. On Quasilinear-Time Complexity Theory. *Theor. Comput. Sci.* 148(2):325–349, 1995.
- [PPR80] W. J. Paul, E. J. Prauss, and R. Reischuk. On alternation. *Acta Informatica* 14:243–255, 1980.
- [PF79] N. Pippenger and M. J. Fischer. Relations Among Complexity Measures. *JACM* 26(2):361–381, 1979.
- [Tou01] I. Tourlakis. Time-Space Tradeoffs for SAT on Nonuniform Machines. *J. Comput. Syst. Sci.* 63(2):268–287, 2001.
- [Will08] R. Williams. Non-Linear Time Lower Bound for (Succinct) Quantified Boolean Formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 2008.
- [Will11] R. Williams. Non-Uniform ACC Circuit Lower Bounds. *IEEE Conference on Computational Complexity*, 115–125, 2011.

APPENDIX

Here we give a few more details about why the tape head positions of the Pippenger-Fischer circuit construction (for multitape Turing machines running in time $n \cdot \text{poly}(\log n)$) can be computed in $\text{poly}(\log n)$ time, given the index of a timestep $t = 1, \dots, n \cdot \text{poly}(\log n)$. WLOG, suppose the running time of the machine M is a power of two, 2^k . One tape is for copying and pasting blocks of tape cells; another tape holds the configurations of all tapes (their cell content and “virtual” head positions) on its tracks. On the second tape, there is a specially marked cell called the “home” cell. At the beginning, all “virtual” tape heads are positioned on the home cell.

First, the simulation of one step of M is done by direct simulation, moving the head one cell to the left (marking the tape for those tape heads that are supposed to move to the left) then moving two steps to the right (for those tape heads that move to the right), then moving left once to recenter the tape head of the simulation at the home cell. (Note we are implicitly assuming that the virtual tape heads are within one cell of the home cell: in general, the invariant is that at the beginning of a 2^q -step simulation, all virtual tape heads

are within 2^q cells of the home cell.) Then the simulation of 2 steps is done, followed by 4 steps, 8 steps, and so on, until 2^k steps are reached.

The simulation of 2^q steps of M takes two recursive calls to simulate 2^{q-1} steps, with two $O(2^q)$ -time phases of “shifting” before the calls, in which the tape heads move obliviously. A shifting phase does a cyclic shift of the $2^{q+1} + 1$ cells of the tape that are within 2^q cells of the current tape head position, either to the left or to the right, for 2^{q-1} cells (depending on the content and the state). (Recall the purpose of this shift is to bring the “virtual” heads to within 2^{q-1} cells of the home cell, to facilitate the recursive calls.) This involves sweeping the tape heads 2^q cells to the left of the current head position and 2^q cells to the right of this position, for a small constant number of times. After the second recursive call, two “cleanup” phases occur, where the cyclic shifts from the shifting phases are undone; this requires that the two shifting phases record a few bits for the later cleanup phases.

To produce the head positions for a given timestep t , we only have to calculate where step t lies within the above procedure. The number of steps in the simulation of 2^q steps of M is precisely $c \cdot 2^{q-1}$ (the cost of the shifting phases and cleanup phases) plus twice the number of steps used to simulate for 2^{q-1} steps, for a small integer c . Hence each simulation of 2^q steps of M takes precisely $d \cdot q 2^q$ steps, for a fixed integer d . This 2^q -step simulation is run in phases, for increasing $q = 0, 1, 2, \dots$, until halting is reached. To determine which phase q that step t refers to, we only need $\text{poly}(\log n)$ arithmetic operations on $O(\log n)$ -size integers. (For example, we could compute $t := t - d \cdot i 2^i$ for increasing i , until $t < 0$. This yields the desired q as well as the step number s within the simulation of 2^q steps.) Then, determining where t lies within the simulation of 2^q steps again only requires $\text{poly}(\log n)$ arithmetic operations on $O(\log n)$ -size integers. (If $s < e \cdot 2^{q-1}$ for a certain integer e , then we are in the first shifting phase; if $s \in [e \cdot 2^{q-1}, e \cdot 2^{q-1} + d \cdot (q-1) 2^{q-1}]$ we are in the first simulation of 2^{q-1} steps; if $s \in [e \cdot 2^{q-1} + d \cdot (q-1) 2^{q-1} + 1, e \cdot 2^{q-1} + d \cdot (q-1) 2^{q-1} + e \cdot 2^{q-1}]$ we are in the second shifting phase, *etc.* The point is that all of these checks can be done with a little arithmetic.)

It is worth noting that even stronger uniformity conditions are known to hold for the Pippenger-Fischer simulation. Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [BGHSV05] have observed (for the purpose of providing efficiently described PCPs) that an $O(\log n)$ -space uniform circuit of $O(\text{poly}(n))$ size and $O(\log n)$ depth can implement the above reduction for simulating 2^n steps, by embedding the oblivious simulation of M into a computation on a butterfly graph (or de Bruijn graph) and using the recursive structure of butterfly graphs to ensure an easily-described circuit.